OGC® DOCUMENT: 21-018

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t17-D027

OGC TESTBED-17: FEATURES AND GEOMETRIES JSON CRS ANALYSIS OF ALTERNATIVES ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2021-11-19
Approval Date: 2021-12-09
Publication Date: 2022-02-08
Editor: Panagiotis (Peter) A. Vretanos

CONTENTS

LIST OF FIGURES

I ABSTRACT

One of the primary requirements for the OGC Testbed-17 Features and Geometries JSON task is to define an extension or profile of GeoJSON that supports encoding spatiotemporal data in Coordinate Reference Systems (CRS) other than the GeoJSON default of the World Geodetic System 1984 (WGS 84) datum, with longitude and latitude units of decimal degrees (CRS84).

This OGC Testbed 17 (TB17) Engineering Report (ER) presents the various alternatives considered for declaring CRS information in a Features and Geometries JSON (JSON-FG) file. JSON-FG is an OGC extension to GeoJSON that, among other things, adds support of coordinate reference systems other than the CRS84 default. One of the alternatives was selected to be the mechanism for declaring CRS information in a JSON-FG document and is fully described in the "OGC Testbed-17: OGC Features and Geometries JSON Engineering Report" (OGC 21-017r1).

This ER was submitted to the OGC Features and Geometries JSON Standards Working Group so that the work in TB17 can inform their task of developing and documenting a Features and Geometries JSON standard.

II EXECUTIVE SUMMARY

GeoJSON is a popular encoding format for geospatial data [1]. GeoJSON is targeting developers who need to integrate geospatial data into their application but are not necessarily experts in GIS and are thus not aware of topics such as CRS, 3-D geometries, feature types, etc. As a result, certain choices were made in the design of GeoJSON in order to simplify the format. One of those choices was to fix the CRS that can be used in a GeoJSON file to CRS84.

For developers who are more familiar with the use of geospatial data and GIS professional this restriction limits the scope of use of GeoJSON. The JSON-FG format extends GeoJSON by dropping this restriction on the CRS that may be used to encoding geospatial data. This ER proposes and analyzes a number of alternative approaches for declaring the CRS being used to encoding spatiotemporal and related data in a JSON-FG document. To cover a variety of requirements the following alternatives were considered:

1. CRS information is asserted in a JSON-FG document by URI reference.
2. CRS information is asserted in a JSON-FG document by value. This entails copying the definition of the CRS into the file.
3. Specifying a dictionary of CRSs used in a JSON-FG document at the top of the document and then locally referencing values from that dictionary to assert the CRS of a spatial value in the document.

4. CRS information is asserted in the JSON-FG document for simple non-spatial values, such as height, that are related to the spatial data in the file but are specified separately from the spatial data.

Of these options, the testbed participants determined that the primary requirement — that is, asserting the CRS being used for spatial data in a JSON-FG file — is satisfied by Option 1 from this list. Option 1 is described in detail in clause in this document titled “Asserting Coordinate Reference Systems in a JSON-FG document”. The remaining options are described in detail in the “Alternatives for asserting CRS in a JSON-FG document” clause.

III KEYWORDS

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

ogcdoc, OGC document, JSON, GeoJSON, Features and Geometries

IV PREFACE

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

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

V SECURITY CONSIDERATIONS

No security considerations have been made for this document.

VI SUBMITTING ORGANIZATIONS

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

- CubeWerx

VII SUBMITTERS

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

NAME	ORGANIZATION	ROLE
Panagiotis (Peter) Vretanos	CubeWerx	Editor
Stefano Bovio	GeoSolutions	Contributor
Felipe Carrillo Romero	Hexagon	Contributor
Ignacio Correas	Skymantics	Contributor
Patrick Dion	Ecere	Contributor
Jérôme Jacovella-St-Louis	Ecere	Contributor
Clemens Portele	interactive instruments	Contributor

1

SCOPE

1 SCOPE

2

TERMS, DEFINITIONS AND ABBREVIATED TERMS

2 TERMS, DEFINITIONS AND ABBREVIATED TERMS

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

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

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

2.1. Terms and definitions

2.1.1. **coordinate reference system**

coordinate system that is related to an object by a datum (source: OGC Topic 2, version 5.0)

2.2. Abbreviated terms

CRS	Coordinate Reference System
JSON	JavaScript Object Notation
JSON-FG	Features and Geometries JSON

3

INTRODUCTION

3 INTRODUCTION

The Asserting Coordinate Reference Systems in a JSON-FG document clause describes the primary means, and the one recommended to the Features and Geometries Standards Working Group (SWG), for asserting CRS information in a JSON-FG file.

The Alternatives for asserting CRS in a JSON-FG document clause describes the other alternatives that were considered for asserting CRS information in a JSON-FG file.

The Implementations clause describes the components developed, deployed and tested in the testbed.

4

ASSERTING COORDINATE REFERENCE SYSTEMS IN A JSON-FG DOCUMENT

4 ASSERTING COORDINATE REFERENCE SYSTEMS IN A JSON-FG DOCUMENT

4.1. Introduction

A primary goal of the OGC Testbed 17 Features and Geometries JSON thread was to investigate and define a solution that enables GeoJSON to support different Coordinate Reference Systems (CRS) as either an extension or within a profile of GeoJSON.

In coordination with the OGC Features and Geometries JSON SWG, the approach selected was to build on the widely used GeoJSON standard and to **extend** it with minimal extensions to support, among other things, the ability to use CRSs other than CRS84(h). A number of approaches were considered for declaring alternative CRSs for spatiotemporal values in a JSON-FG document. One of these approaches was selected to be recommended for the core of JSON-FG and is documented in this clause. The other approaches are documented in the “Alternatives for declaring CRS in a JSON-FG document clause.

4.2. Considerations

The following considerations were taken into account in deciding how to declare a CRS for spatial-temporal values in a JSON-FG document:

1. Reuse GeoJSON CRS mechanism

   - **Consideration**: Should we reuse the `crs` mechanism from the legacy version of GeoJSON?

   - **Decision**: No. It is tempting to reuse the legacy mechanism because over the years a large number of GeoJSON clients have emerged that can handle the `crs` key and make use of the *prior arrangement* provision of the legacy GeoJSON specification. However, reusing this mechanism would introduce a significant amount of ambiguity. For example, whether the presence of the `crs` key was targeting the GeoJSON keys (i.e. `geometry`) or the JSON-FG keys (i.e. `where`) would be unclear. Even though the GeoJSON RFC does not define a CRS mechanism, the mere fact that dependencies have been built on the legacy mechanism disqualifies it from being redefined for JSON-FG use.

2. Scope of applicability

   - **Consideration**: To which values in a JSON-FG file should the specified reference systems apply?

   - **Decision**: The key(s) defined in this proposal only apply to JSON-FG spatiotemporal objects (e.g. `where`, `when`) and any spatiotemporal value in the `properties` section which are not bound by GeoJSON requirements. So, the herein defined key(s) **don't** apply to any defined GeoJSON spatiotemporal element such as `geometry`.

3. Temporal and vertical reference systems.

   - **Consideration**: Should temporal and vertical reference systems be supported?

   - **Decision**: The most common use cases involve geographic or projected coordinate reference systems and this ER focuses on correctly declaring those CRSs in a JSON-FG document. However, temporal and vertical reference systems are also just coordinate reference systems and so the proposed mechanism for declaring a CRS in a JSON-FG file is also applicable to vertical and/or temporal reference systems.

4. Coordinate order

   - **Consideration**: According to the GeoJSON RFC: *"The coordinate reference system for all GeoJSON coordinates is a geographic coordinate reference system, using the World Geodetic System 1984 (WGS 84) [WGS84] datum, with longitude and latitude units of decimal degrees. This is equivalent to the coordinate reference system identified by the Open Geospatial Consortium (OGC) URN urn:ogc:def:crs:OGC::CRS84. An OPTIONAL third-position element SHALL be the height in meters above or below the WGS 84 reference ellipsoid. In the absence of elevation values, applications sensitive to height or depth SHOULD interpret positions as being at local ground or sea level."* There is also the *prior agreement* provision that loosens this requirement if all parties agree. With regard to coordinate order, what should the requirements be for JSON-FG spatial-temporal values?

   - **Decision**: For JSON-FG spatiotemporal values, the coordinate order — following OGC policy — shall be in the order specified by the declared CRS.

5. Scoping rules

   - **Consideration**: At which level in a JSON-FG file should the CRS be declared (i.e. collection, feature and/or value level)?

   - **Decision**: Reference system information for JSON-FG spatiotemporal values will primarily be declared at the top level of a file (collection or feature level) since it is likely that all values in a document will be in the same CRS. However, having defined a structure for declaring a CRS at the top level, reusing the key(s) to declare a CRS at deeper levels is a simple matter (i.e. feature-level or value-level).

6. Compound reference systems

   - **Consideration**: Should compound reference systems be supported?
   - **Decision**: Yes. A mechanism is defined in this clause for specifying a compound CRS.

7. Specifying reference system information by value or by reference

   - **Consideration**: How should CRS information be included in a JSON-FG file? By value or by reference?
   - **Decision**: Handling CRS by reference seems reasonable for JSON-FG core and also aligns with Features Part 2. While being able to specify CRSs by value might be a useful capability, whether this should be in the core of JSON-FG is unclear. First, most clients have connectivity and so a CRS reference is sufficient since the CRS definition can be retrieved as necessary. Second, while many desktop and web clients recognize CRS identifiers, it is unclear whether web and other clients are generally able to easily parse and manipulate, for example, a WKT2 representation of a CRS. For these reasons, specifying the CRS by value in a JSON-FG document was omitted from the core and is documented as an alternative in the Alternatives clause.

8. Dictionary of reference systems

   - **Consideration**: Is there a need to specify the reference systems used in a JSON-FG file up-front in a dictionary?
   - **Decision**: This capability is closely related to the CRS-by-value approach and mirrors what is done in GeoPackage. This approach might be useful for the “offline container” use case described in the OGC Testbed 17 Call For Participation. While the offline use case is a valid use case, participants felt that most clients would have connectivity thus alleviating the need for a local copy of the CRS definition. A CRS reference via URL, it was felt, is sufficient and connected client can simply look up the CRS definition as necessary. For this reason, the up-front dictionary approach was rejected for the core of JSON-FG. However, the specific approach for encoding a dictionary of CRSs in a JSON-FG document considered in the testbed is described in detail in the Alternatives clause. Please note that this approach was used in an offline container experiment described in the *OGC Testbed-17: OGC Features and Geometries JSON Engineering Report* [2].

9. Separate vertical reference system handling

- **Consideration**: Is there a requirement for separate vertical CRS handling?
- **Decision**: Many times the height or depth component of data is handled as a separate property rather than being in the geometry. Such properties are typically defined using a simple type (i.e. double). As such, unlike object values such as GeoJSON's `geometry` element or JSON-FG's `where` element, an additional key conveying vertical CRS information cannot be conveniently and unambiguously added to the structure. While this might be another useful capability, the thread participants deemed that this is not a core requirement for JSON-FG. The specific approach considered for handling separate vertical CRSs in a JSON-FG document is described in the Alternatives clause.

4.3. The coord-ref-sys key

4.3.1. Overview

Spatiotemporal objects are specified relative to some reference system.

GeoJSON (both the current RFC and the legacy version) fixed the reference system for geometric values to the "WGS84 datum, and with longitude and latitude units of decimal degrees". The legacy version included a "prior arrangement" provision to allow other reference systems to be used and also defined the `crs` key for declaring the reference system. This *prior arrangement* mechanism survived into the RFC but the accompanying `crs` key did not. The result is that there is no interoperable way to unambiguously declare a different CRS in GeoJSON files. Therefore the safest thing to do is to use OGC CRS84(h) for GeoJSON members and ignore the *prior arrangement* provision and the legacy `crs` key.

JSON-FG is not bound by these restrictions and so this clause describes a new key named `coord-ref-sys` that can be used to declare the CRS for spatiotemporal values in a JSON-FG document. This new key does not interfere with anything, past or present, defined in any of the GeoJSON specifications. Therefore, GeoJSON elements can continue to operate as always but JSON-FG elements can provide enhanced CRS support.

4.3.2. CRS values

A CRS value can be specified in a JSON-FG document in one of three ways:

1. As a simple reference using a URI,
2. As a simple reference using a URI accompanied by an epoch value,

3. As an array of CRS values denoting an ad hoc compound CRS.

The following JSON Schema fragment defines a JSON-Schema for a CRS value:

Example 1:

```
{
   "$defs": {
      "refsys-simpleref": {
         "type": "string",
         "format": "uri"
      },
      "refsys-byref": {
         "type": "object",
         "required": [ "href" ],
         "properties": {
            "href": {
               "type": "string",
               "format": "uri"
            },
            "epoch": {
               "type": "string"
            }
         }
      },
      "refsys": {
         "oneOf": [
            { "$ref": "#/$defs/refsys-simpleref" },
            { "$ref": "#/$defs/refsys-byref" },
            {
               "type": "array",
               "items": {
                  "oneOf": [
                     { "$ref": "#/$defs/refsys-simpleref" },
                     { "$ref": "#/$defs/refsys-byref" },
                  ]
               }
            }
         ]
      }
   },
   "$ref": "#/$defs/refsys"
}
```

The following are examples of CRS values:

Example 2 — A simple reference system value by reference.:

```
"http://www.opengis.net/def/crs/EPSG/0/3857"
```

Example 3 — A reference system value by reference and with an epoch.:

```
{
  "href": "http://www.opengis.net/def/crs/EPSG/0/3857",
  "epoch": "2016.47"
}
```

Example 4 — A ad hoc compound reference system value:

```
[
  {
    "href": "http://www.opengis.net/def/crs/EPSG/0/25832",
    "epoch": "2016.47"
  },
  "http://www.opengis.net/def/crs/EPSG/0/5783"
]
```

4.3.3. Schema for the `coord-ref-sys` key

The following JSON-Schema defines the `coord-ref-sys` key which can used to declare the CRS of JSON-FG spatial-temporal values in a JSON-FG document:

Example 1:

```
{"$ref": "#/$defs/refsys" }
```

NOTE: Readers of this ER should be aware that the JSON-FG SWG has renamed the `coord-ref-sys` key to be `coordRefSys` following the key-naming conventions adopted by the SWG. This ER continues to use the name `coord-ref-sys` to reflect what components in the JSON-FG thread generated and consumed.

Used at the collection level, the `coord-ref-sys` key declares the CRS for JSON-FG spatiotemporal values found anywhere in the JSON document that are not otherwise tagged with closer-to-scope CRS information.

Example 2 — An example of declaring CRS information at the collection level.:

```
{
  "type": "FeatureCollection",
  "@context": "https://t17.ldproxy.net/airspace/collections/class_all/context",
  "@type": "geojson:FeatureCollection",
  "numberReturned": 10,
  "numberMatched": 5573,
  "timeStamp": "2021-07-18T23:00:25Z",
  "coord-ref-sys":  "http://www.opengis.net/...",
  "features": [ ... ]
}
```

Used at the feature level, the `coord-ref-sys` key declares the CRS for geometric JSON-FG values found anywhere in the feature that are not otherwise tagged with closer-to-scope CRS information.

Example 3 — Reference system specified at the feature-level.:

```
{
   "type": "Feature",
   "id": "DENW19AL0000giv5BL",
   "coord-ref-sys": "http://www.opengis.net/def/crs/EPSG/0/5555",
   "geometry": {
      "type": "Polygon",
      "coordinates": [
         [
            [ 8.709204563652449, 51.50352856284526, 100.0 ],
            [ 8.709312860802727, 51.503457005181794, 100.0 ],
            [ 8.709391968693081, 51.50350306810203, 100.0 ],
            [ 8.709283757429898, 51.503574715968284, 100.0 ],
            [ 8.709204563652449, 51.50352856284526, 100.0 ]
         ]
      ]
   },
   "when": [ "2014-04-24T10:50:18Z", null ],
   "where": {
      "type": "Polyhedron",
      "coordinates": [
         [ [ [ 479816.67, 5705861.672, 100 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479816.67, 5705861.672, 100 ] ]
         ],
         [ [ [ 479816.67, 5705861.672, 110 ],
             [ 479824.155, 5705853.684, 110 ],
             [ 479829.666, 5705858.785, 120 ],
             [ 479822.187, 5705866.783, 120 ],
             [ 479816.67, 5705861.672, 110 ] ]
         ],
         [ [ [ 479816.67, 5705861.672, 110 ],
             [ 479824.155, 5705853.684, 110 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479816.67, 5705861.672, 100 ],
             [ 479816.67, 5705861.672, 110 ] ]
         ],
         [ [ [ 479824.155, 5705853.684, 110 ],
             [ 479829.666, 5705858.785, 120 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479824.155, 5705853.684, 110 ] ]
         ],
         [ [ [ 479829.666, 5705858.785, 120 ],
             [ 479822.187, 5705866.783, 120 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479829.666, 5705858.785, 120 ] ]
         ],
         [ [ [ 479822.187, 5705866.783, 120 ],
             [ 479816.67, 5705861.672, 110 ],
             [ 479816.67, 5705861.672, 100 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479822.187, 5705866.783, 120 ] ]
         ]
      ]
   },
   "properties": {
      "lastChange": "2014-04-24T10:50:18Z",
      "function": "Agricultural building",
      "height_m": 20.0
   }
}
```

Used at the value level, the `coord-ref-sys` key declares the CRS for the JSON-FG value within which the key is contained.

Example 4 — Reference system specified at the value-level.:

```
{
   "type": "Feature",
   "id": "DENW19AL0000giv5BL",
   "geometry": {
      "type": "Polygon",
      "coordinates": [
         [
            [ 8.709204563652449, 51.50352856284526, 100.0 ],
            [ 8.709312860802727, 51.503457005181794, 100.0 ],
            [ 8.709391968693081, 51.50350306810203, 100.0 ],
            [ 8.709283757429898, 51.503574715968284, 100.0 ],
            [ 8.709204563652449, 51.50352856284526, 100.0 ]
         ]
      ]
   },
   "when": [ "2014-04-24T10:50:18Z", null ],
   "where": {
      "type": "Polyhedron",
      "coord-ref-sys": "http://www.opengis.net/def/crs/EPSG/0/5555",
      "coordinates": [
         [ [ [ 479816.67, 5705861.672, 100 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479816.67, 5705861.672, 100 ] ]
         ],
         [ [ [ 479816.67, 5705861.672, 110 ],
             [ 479824.155, 5705853.684, 110 ],
             [ 479829.666, 5705858.785, 120 ],
             [ 479822.187, 5705866.783, 120 ],
             [ 479816.67, 5705861.672, 110 ] ]
         ],
         [ [ [ 479816.67, 5705861.672, 110 ],
             [ 479824.155, 5705853.684, 110 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479816.67, 5705861.672, 100 ],
             [ 479816.67, 5705861.672, 110 ] ]
         ],
         [ [ [ 479824.155, 5705853.684, 110 ],
             [ 479829.666, 5705858.785, 120 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479824.155, 5705853.684, 100 ],
             [ 479824.155, 5705853.684, 110 ] ]
         ],
         [ [ [ 479829.666, 5705858.785, 120 ],
             [ 479822.187, 5705866.783, 120 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479829.666, 5705858.785, 100 ],
             [ 479829.666, 5705858.785, 120 ] ]
         ],
         [ [ [ 479822.187, 5705866.783, 120 ],
             [ 479816.67, 5705861.672, 110 ],
             [ 479816.67, 5705861.672, 100 ],
             [ 479822.187, 5705866.783, 100 ],
             [ 479822.187, 5705866.783, 120 ] ]
         ]
      ]
   },
   "properties": {
      "lastChange": "2014-04-24T10:50:18Z",
      "function": "Agricultural building",
      "height_m": 20.0
   }
}
```

5

ALTERNATIVES FOR DECLARING CRSS IN A JSON-FG DOCUMENT

5 ALTERNATIVES FOR DECLARING CRSS IN A JSON-FG DOCUMENT

5.1. Introduction

Various alternatives were considered for declaring a CRS in a JSON-FG document. These alternatives are not recommended to be part of the JSON-FG core and the reasons are discussed in the Considerations clause. These alternative approaches, however, are documented so that they may become the basis for future extensions to JSON-FG.

5.1.1. Coordinate reference systems by value

5.1.1.1. Overview

This clause specifies how to encode a CRS by value in a JSON-FG document using any well-known encoding such as PROJJSON or OGC-WKT2. Other encodings are allowed (e.g. GML CRS) but are not discussed here.

5.1.1.2. Value schema

The following JSON-Schema fragment defines a reference system value. The fragment extends the schema defined in the CRS values clause:

Example 1:

```
{
   "$defs": {
      "refsys-simple-ref": {
         "type": "string",
         "format": "uri"
      },
      "refsys-byref": {
         "type": "object",
         "required": [ "href" ],
         "properties": {
            "href": {
               "type": "string",
               "format": "uri"
            },
            "epoch": {
               "type": "string"
         }
         }
      },
      "refsys-byvalue": {
         "type": "object",
         "required": [ "valueType", "value" ],
         "properties": {
            "valueType": {
               "type": "string",
               "enum": ["projjson","ogcwkt"]
            },
            "value": {
               "oneOf": [
                  { "type": "string" },
                  { "type": "object" },
               ]
            },
            "epoch": {
               "type": "string"
            }
         }
      },
      "refsys": {
         "oneOf": [
            { "$ref": "#/$defs/refsys-simpleref" },
            { "$ref": "#/$defs/refsys-byref" },
            { "$ref": "#/$defs/refsys-byvalue" },
            {
               "type": "array",
               "items": {
                  "oneOf": [
                     { "$ref": "#/$defs/refsys-simpleref" },
                     { "$ref": "#/$defs/refsys-byref" },
                     { "$ref": "#/$defs/refsys-byvalue" },
                  ]
               }
            }
         ]
      }
   },
   "$ref": "#/$defs/refsys"
}
```

Example 2 — A vertical reference system value specified by value using OGC WKT notation.:

```
{
  "valueType": "ogcwkt",
  "value": "VERTCRS[\"NAVD88\", VDATUM[\"North American Vertical Datum 1988\"],
 CS[vertical,1], AXIS[\"gravity-related height (H)\",up], LENGTHUNIT[\"metre
\",1.0]]"
}
```

Example 3 — A temporal reference system value specified by value using PROJ JSON notation.:

```
{
  "valueType": "projjson",
  "value": {
    "type": "TemporalCRS",
    "name": "Gregorian",
    "datum": {
      "type": "TemporalDatum",
      "name": "Gregorian",
      "calendar": "Gregorian"
    }
  }
}
```

Example 4 — A coordinate reference system value specified by value using OGC WKT notation.:

```
{
  "valueType": "ogcwkt",
  "value": "PROJCRS[\"NAD27 / Texas South Central\", BASEGEOGCRS[\"NAD27\",
 DATUM[\"North American Datum 1927\", ELLIPSOID[\"Clarke 1866\",20925832.
164,294.97869821, LENGTHUNIT[\"US survey foot\",0.304800609601219]] ] ],
 CONVERSION[\"Texas South Central SPCS27\", METHOD[\"Lambert Conic
 Conformal (2SP)\",ID[\"EPSG\",9802]], PARAMETER[\"Latitude of false origin
\",27.83333333333333, ANGLEUNIT[\"degree\",0.0174532925199433],ID[\"EPSG
\",8821]], PARAMETER[\"Longitude of false origin\",-99.0, ANGLEUNIT[\"degree
\",0.0174532925199433],ID[\"EPSG\",8822]], PARAMETER[\"Latitude of
 1st standard parallel\",28.383333333333, ANGLEUNIT[\"degree\",0.
0174532925199433],ID[\"EPSG\",8823]], PARAMETER[\"Latitude of 2nd standard
 parallel\",30.283333333333, ANGLEUNIT[\"degree\",0.0174532925199433],ID[\"EPSG
\",8824]], PARAMETER[\"Easting at false origin\",2000000.0, LENGTHUNIT[\"US
 survey foot\",0.304800609601219],ID[\"EPSG\",8826]], PARAMETER[\"Northing at
 false origin\",0.0, LENGTHUNIT[\"US survey foot\",0.304800609601219],ID[\"EPSG
\",8827]] ], CS[Cartesian,2], AXIS[\"(X)\",east], AXIS[\"(Y)\",north],
 LENGTHUNIT[\"US survey foot\",0.304800609601219], REMARK[\"Fundamental point:
 Meade's Ranch KS, latitude 39°13'26.686\"\"N, longitude 98°32'30.506\"\"W.
\"] ]"
}
```

Example 5 — A coordinate reference system value specified by value using PROJ JSON notation.:

```
{
  "valueType": "projjson",
  "value": {
    "$schema": "https://proj.org/schemas/v0.1/projjson.schema.json",
    "type": "ProjectedCRS",
    "name": "WGS 84 / UTM zone 31N",
    "base_crs": {
      "name": "WGS 84",
      "datum": {
        "type": "GeodeticReferenceFrame",
        "name": "World Geodetic System 1984",
        "ellipsoid": {
          "name": "WGS 84",
          "semi_major_axis": 6378137,
          "inverse_flattening": 298.257223563
        }
      },
      "coordinate_system": {
        "subtype": "ellipsoidal",
        "axis": [
          {
            "name": "Geodetic latitude",
            "abbreviation": "Lat",
            "direction": "north",
            "unit": "degree"
          },
          {
            "name": "Geodetic longitude",
            "abbreviation": "Lon",
            "direction": "east",
            "unit": "degree"
          }
        ]
      },
      "id": {
        "authority": "EPSG",
        "code": 4326
      }
    },
    "conversion": {
      "name": "UTM zone 31N",
      "method": {
        "name": "Transverse Mercator",
        "id": {
          "authority": "EPSG",
          "code": 9807
        }
      },
      "parameters": [
        {
          "name": "Latitude of natural origin",
          "value": 0,
          "unit": "degree",
          "id": {
            "authority": "EPSG",
            "code": 8801
          }
        },
        {
          "name": "Longitude of natural origin",
          "value": 3,
          "unit": "degree",
          "id": {
            "authority": "EPSG",
            "code": 8802
          }
        },
        {
          "name": "Scale factor at natural origin",
          "value": 0.9996,
          "unit": "unity",
          "id": {
            "authority": "EPSG",
            "code": 8805
          }
        },
        {
          "name": "False easting",
          "value": 500000,
          "unit": "metre",
          "id": {
            "authority": "EPSG",
            "code": 8806
          }
        },
        {
          "name": "False northing",
          "value": 0,
          "unit": "metre",
          "id": {
            "authority": "EPSG",
            "code": 8807
          }
        }
      ]
    },
    "coordinate_system": {
      "subtype": "Cartesian",
      "axis": [
        {
          "name": "Easting",
          "abbreviation": "E",
          "direction": "east",
          "unit": "metre"
        },
        {
          "name": "Northing",
          "abbreviation": "N",
          "direction": "north",
          "unit": "metre"
        }
      ]
    },
    "area": "World - N hemisphere - 0°E to 6°E - by country",
    "bbox": {
      "south_latitude": 0,
      "west_longitude": 0,
      "north_latitude": 84,
      "east_longitude": 6
    },
    "id": {
      "authority": "EPSG",
      "code": 32631
    }
  }
}
```

Example 6 — A compound reference system value specified by reference and value.:

```
[
  "http://www.opengis.net/def/crs/EPSG/0/25832",
  {
    "valueType": "ogcwkt",
    "VERTCRS[\"DHHN92 height\",VDATUM[\"Deutsches Haupthoehennetz
 1992\",ID[\"EPSG\",5181]],CS[vertical,1,ID[\"EPSG\",6499]],AXIS[\"Gravity-
related height (H)\",up],LENGTHUNIT[\"metre\",1,ID[\"EPSG\",9001]],ID[\"EPSG
\",5783]]"
  }
]
```

5.1.2. Dictionary of coordinate reference systems

This clause specifies how to encode a dictionary of CRS values used in a JSON-FG document. Values from this dictionary can then be referenced, using the `coord-ref-sys` key, to declare the CRS for JSON-FG values within the enclosing document.

The `ref-systems` key is defined for specifying the dictionary of reference systems used in a JSON-FG document. The following schema fragment defines the schema of the `ref-systems` key:

Example 1 — The schema of the `ref-systems` key:

```
{
   "type": "object",
   "patternProperties": {
      ".*": { "$ref": "#/$defs/refsys" },
   }
}
```

Values from the dictionary can be referenced in the rest of the document using the appropriate key (i.e. `coord-ref-sys`) with a relative URL/JSON pointer referencing a value from this dictionary.

Example 2 — An example of a dictionary of reference systems.:

```
{
  "type": "FeatureCollection",
  "numberReturned": 10,
  "numberMatched": 5573,
  "timeStamp": "2021-07-18T23:00:25Z",
  "ref-systems": {
    "25832": {
      "valueType": "ogcwkt",
      "value": "PROJCRS[\"ETRS89 / UTM zone 32N
\",BASEGEOGCRS[\"ETRS89\",ENSEMBLE[\"European Terrestrial Reference System 1989
 ensemble\", MEMBER[\"European Terrestrial Reference Frame 1989\", ID[\"EPSG
\",1178]], MEMBER[\"European Terrestrial Reference Frame 1990\", ID[\"EPSG
\",1179]], MEMBER[\"European Terrestrial Reference Frame 1991\", ID[\"EPSG
\",1180]], MEMBER[\"European Terrestrial Reference Frame 1992\", ID[\"EPSG
\",1181]], MEMBER[\"European Terrestrial Reference Frame 1993\", ID[\"EPSG
\",1182]], MEMBER[\"European Terrestrial Reference Frame 1994\", ID[\"EPSG
\",1183]], MEMBER[\"European Terrestrial Reference Frame 1996\", ID[\"EPSG
\",1184]], MEMBER[\"European Terrestrial Reference Frame 1997\", ID[\"EPSG
\",1185]], MEMBER[\"European Terrestrial Reference Frame 2000\", ID[\"EPSG
\",1186]], MEMBER[\"European Terrestrial Reference Frame 2005\", ID[\"EPSG
\",1204]], MEMBER[\"European Terrestrial Reference Frame 2014\", ID[\"EPSG
\",1206]], ELLIPSOID[\"GRS 1980\",6378137,298.2572221,LENGTHUNIT[\"metre
\",1,ID[\"EPSG\",9001]],ID[\"EPSG\",7019]], ENSEMBLEACCURACY[0.
1],ID[\"EPSG\",6258]],ID[\"EPSG\",4258]],CONVERSION[\"UTM zone 32N
\",METHOD[\"Transverse Mercator\",ID[\"EPSG\",9807]],PARAMETER[\"Latitude
 of natural origin\",0,ANGLEUNIT[\"degree\",0.0174532925199433,ID[\"EPSG
\",9102]]],PARAMETER[\"Longitude of natural origin\",9,ANGLEUNIT[\"degree
\",0.0174532925199433,ID[\"EPSG\",9102]]],PARAMETER[\"Scale factor at natural
 origin\",0.9996,SCALEUNIT[\"unity\",1,ID[\"EPSG\",9201]]],PARAMETER[\"False
 easting\",500000,LENGTHUNIT[\"metre\",1,ID[\"EPSG\",9001]]],PARAMETER[\"False
 northing\",0,LENGTHUNIT[\"metre\",1,ID[\"EPSG\",9001]]],ID[\"EPSG
\",16032]],CS[Cartesian,2,ID[\"EPSG\",4400]],AXIS[\"Easting
 (E)\",east],AXIS[\"Northing (N)\",north],LENGTHUNIT[\"metre\",1,ID[\"EPSG
\",9001]],ID[\"EPSG\",25832]]"
    },
    "5783": {
      "valueType": "ogcwkt",
      "value": "VERTCRS[\"DHHN92 height\",VDATUM[\"Deutsches Haupthoehennetz
 1992\",ID[\"EPSG\",5181]],CS[vertical,1,ID[\"EPSG\",6499]],AXIS[\"Gravity-
related height (H)\",up],LENGTHUNIT[\"metre\",1,ID[\"EPSG\",9001]],ID[\"EPSG
\",5783]]"
    }
  },
  "coord-ref-sys":  [
    { "href": "#/ref-systems/25832" },
    { "href": "#/ref-systems/5783" }
  ]
  "features": [ ... ]
}
```

5.1.3. Reference systems for simple JSON-FG values

Declaring a reference system for structured JSON-FG values such as `where` is a simple matter because additional properties can be added to the structure that convey the reference system information. For simple JSON-FG values (e.g. `height` defined as a `double`) this is not possible. Consider the following example:

Example 1 — A simple height property without a reference system.:

```
{
   "type": "Feature",
   "id": "DENW19AL0000giv5BL",
   "coord-ref-sys": "http://www.opengis.net/def/crs/EPSG/0/5555",
   "geometry": {
      "type": "Polygon",
      "coordinates": [ ... ]
   },
   "when": [ "2014-04-24T10:50:18Z", null ],
   "where": {
      "type": "Polyhedron",
      "coordinates": [ ... ]
   },
   "properties": {
      "lastChange": "2014-04-24T10:50:18Z",
      "function": "Agricultural building",
      "height": 20.0
   }
}
```

In this example the `height` property does not have a reference system associated with it and declaring one would be awkward since this is a value without structure.

The `value-ref-sys` key is defined for declaring a reference system for simple, unstructured, JSON-FG values with the following schema:

Example 2 — The schema of the `value-ref-sys` key:

```
{
   "type": "object",
   "patternProperties": {
      "(/(([^/~])|(~[01]))*)": { "$ref": "#/$defs/refsys" },
   }
}
```

Used at the collection level, each key of the `value-ref-sys` object is a JSON pointer to a value in the document. The value of the key declares the reference system for the referenced value.

Example 3 — A simple height property with a reference system.:

```
{
   "type": "Feature",
   "id": "DENW19AL0000giv5BL",
   "coord-ref-sys": "http://www.opengis.net/def/crs/EPSG/0/5555",
   "value-ref-sys": {
      "$.properties.height": {
          "valueType": "ogcwkt",
          "value": "VERTCRS[ ... ]"
      }
   },
   "geometry": {
      "type": "Polygon",
      "coordinates": [ ... ]
   },
   "when": [ "2014-04-24T10:50:18Z", null ],
   "where": {
      "type": "Polyhedron",
      "coordinates": [ ... ]
   },
   "properties": {
      "lastChange": "2014-04-24T10:50:18Z",
      "function": "Agricultural building",
      "height": 20.0
   }
}
```

6

IMPLEMENTATIONS

6 IMPLEMENTATIONS

6.1. Servers

6.1.1. interactive instruments (D100)

The interactive instruments deliverable is based on ldproxy. ldproxy is an OGC Reference Implementation for OGC API Features, parts 1 and 2. That is, ldproxy already supported responses in various CRSs before TB17. The version deployed in the testbed uses the GeoTools library for coordinate transformation. In a parallel and independent development to TB17, GeoTools was replaced by the PROJ library. Therefore, ldproxy uses the PROJ library starting with ldproxy version 3.1.

The CRS to be supported by an API can be set as part of the API configuration, for the API or for each collection. All EPSG CRSs supported by the library used for the coordinate conversions can be configured in addition to the default CRSs.

6.1.2. Skymantics (D101)

Skymantics used pygeoapi as a base for its deliverable. pygeoapi in its current version (v 0.11) does not support most requirements in OGC API Features, part 2. Therefore, these had to be coded in order to support various CRSs. In particular, changes were required in the collections configuration file and in the code to generate the collection description, the collection items and the feature item. Moreover, `crs` was added as a reserved fieldname and allowed as a query parameter.

No coordinate transformation was implemented in the server. Instead, a dataset with events in Madrid, that natively stored coordinates in both WGS84 and ETRS89 (EPSG:25830) was used to test the multi-CRS scenario. The GeoJSON `geometry` member was used to publish the WGS84 coordinates, while the JSON-FG `where` member published the EPSG:25830 coordinates. The `coord-ref-sys` member was included at the beginning of each feature announcing the CRS http://www.opengis.net/def/crs/EPSG/9.9.1/25830, which would be used by the JSON-FG `where` member.

6.1.3. CubeWerx (D115)

Deliverable D115 was implemented by CubeWerx Inc. This OGC API — Features server is a component of CubeWerx Suite 9.3.62. CubeWerx Suite is deployed on a bare-metal server running Fedora 31 and using Oracle 11.2 and MariaDB 10.3 as the backend databases to store feature data.

The `cubeserv` module implements OGC API — Features — Part 1: Core and OGC API — Features — Part 2: Coordinate Reference Systems by Reference and was extended in this thread to implement the `coord-ref-sys` (or `coordRefSys`) key and offline dictionary of CRSs in a JSON-FG document. The latter was using in offline container experiment documented in the *OGC Testbed-17: OGC Features and Geometries JSON Engineering Report*.

The implementation of these extensions was straight forward and no issues where encountered.

The following request fetches one feature in the default CRS:

https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1

Figure 1

and generates the following output:

```
{
  "type": "FeatureCollection",
  "timeStamp": "2021-11-02T08:31:26-04:00",
  "numberMatched": 1367145,
  "numberReturned": 1,
  "bbox": [
    -74.004526,
    40.219935,
    -72.033476,
    41.001497
  ],
  "links": [
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1&offset=1",
      "rel": "next",
      "type": "application/vnd.ogc.fg+json"
    },
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1",
      "rel": "self",
      "type": "application/vnd.ogc.fg+json"
    },
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
 Buildings/collections/manhattan_buildings/schemas/collection",
      "rel": "describedby",
      "type": "application/xml"
    },
    .
    .
    .
  ],
  "features": [
    {
      "type": "Feature",
      "featureType": "manhattan_buildings",
      "id": "CWFID.MANHATTAN_BLDGS.0.0",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
              [
                -74.000064,
                40.219953
              ],
              [
                -74.000009,
                40.219935
              ],
              [
                -73.999986,
                40.219977
              ],
              [
                -74.000041,
                40.219995
              ],
              [
                -74.000064,
                40.219953
              ]
            ]
          ]
        },
        "where": null,
        "properties": {
          "bld_height": 36
        }
      }
    ]
  }
```

Requesting a feature in the default CRS.

Since the default CRS (i.e. CRS84) was requested, the value of the `where` key is set to null and the geometry of the feature is specified using the `geometry` key. Since the `where` key is not populated and the spatial information is in the GeoJSON default of CRS, there is no need to declare a CRS and so it is omitted.

The following request makes use of OGC API — Features — Part 2 to request the same response but in a CRS that is not the default of CRS84.

https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1&crs=http://www.opengis.net/def/crs/EPSG/0/3557

Figure 2

Generates the following response:

```
{
  "type": "FeatureCollection",
  "timeStamp": "2021-11-02T08:34:33-04:00",
  "numberMatched": 1367145,
  "numberReturned": 1,
  "coord-ref-sys": "http://www.opengis.net/def/crs/EPSG/0/3557",
  "bbox": [
    -74.004526,
    40.219935,
    -72.033476,
    41.001497
  ],
  "links": [
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1&crs=http://
www.opengis.net/def/crs/EPSG/0/3557&offset=1",
      "rel": "next",
      "type": "application/vnd.ogc.fg+json"
    },
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
%20Buildings/collections/manhattan_buildings/items?f=jsonfg&limit=1&crs=http://
www.opengis.net/def/crs/EPSG/0/3557",
      "rel": "self",
      "type": "application/vnd.ogc.fg+json"
    },
    {
      "href": "https://test.cubewerx.com/cubewerx/cubeserv/tb17/ogcapi/US
 Buildings/collections/manhattan_buildings/schemas/collection",
      "rel": "describedby",
      "type": "application/xml"
    },
    .
    .
    .
  ],
  "features": [
    {
      "type": "Feature",
      "featureType": "manhattan_buildings",
      "id": "CWFID.MANHATTAN_BLDGS.0.0",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -74.000064,
              40.219953
            ],
            [
              -74.000009,
              40.219935
            ],
            [
              -73.999986,
              40.219977
            ],
            [
              -74.000041,
              40.219995
            ],
            [
              -74.000064,
              40.219953
            ]
          ]
        ]
      },
      "where": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -168234.6384,
              -368259.5645
            ],
            [
              -168230.0781,
              -368261.8558
            ],
            [
              -168227.8289,
              -368257.3109
            ],
            [
              -168232.3892,
              -368255.0196
            ],
            [
              -168234.6384,
              -368259.5645
            ]
          ]
        ]
      },
      "properties": {
        "bld_height": 36
      }
    }
  ]
}
```

Requesting a feature in a non-default CRS.

In this case, the value of the `where` key is the feature geometry in the requested CRS. This also triggers the inclusion of the `coord-ref-sys` key in the response to declare that CRS of the `where` key; https://docs.ogc.org/is/18-058/18-058.html in this example. Please note that the `geometry` key is still populated with the feature geometry in CRS84 for backward compatibility with existing GeoJSON client that are unaware of the JSON-FG keys such as `where`.

6.1.4. Ecere (in-kind)

As an in-kind contribution, Ecere implemented support for JSON-FG in its GNOSIS Map Server (demonstration server available), as well as support for *OGC API — Features — Part 2: CRS by reference*, successfully passing the OGC TEAM Engine executable test suite conformance tests. In addition to OGC:CRS84, Ecere implemented support for EPSG:4326 (latitude, longitude), EPSG:3395 (World Mercator) and EPSG:3857 (Web Mercator). Ecere also prototyped an implementation of a CRS extension for *OGC API — Coverages*, *Maps* and *Tiles*, paving the way for a *Common* OGC API CRS building block.

6.2. Clients

6.2.1. Hexagon (D102)

See the clause titled "D102 Features and Geometries JSON Client for Aviation (Hexagon)" in the "OGC Testbed-17: OGC Features and Geometries JSON Engineering Report".

6.2.2. Ecere (D103)

Ecere's GNOSIS Cartographer is a cross-platform 3D visualization client built upon the GNOSIS SDK geospatial visualization and processing library. By extending the library's GeoJSON parsing and loading module, Ecere added support for loading JSON-FG features and feature collections, including its capabilities for describing the CRS in `coord-ref-sys`, and using a CRS other than CRS84 in the `where` property for the geometry.

Because the latest version of the client always renders features in 3D, in practice the client must de-project features coordinates to convert them to CRS84(h) anyways, so there is little value in this case to request the features in an alternate CRS if they are also readily available in CRS84 from the service. However, for the purpose of testing the new capability to load JSON-FG specified in alternate CRS, support for *OGC API — Features — Part 2: CRS by reference* was implemented in the client, and the client was configured to request features in an alternate supported CRS such as Web Mercator (EPSG:3857) or World Mercator (EPSG:3395), even though this requires extra work from the client's perspective.

When a CRS other than CRS84(h) is requested, the coordinates stored in the `where` property are used as geometry instead of the usual GeoJSON `geometry` property.

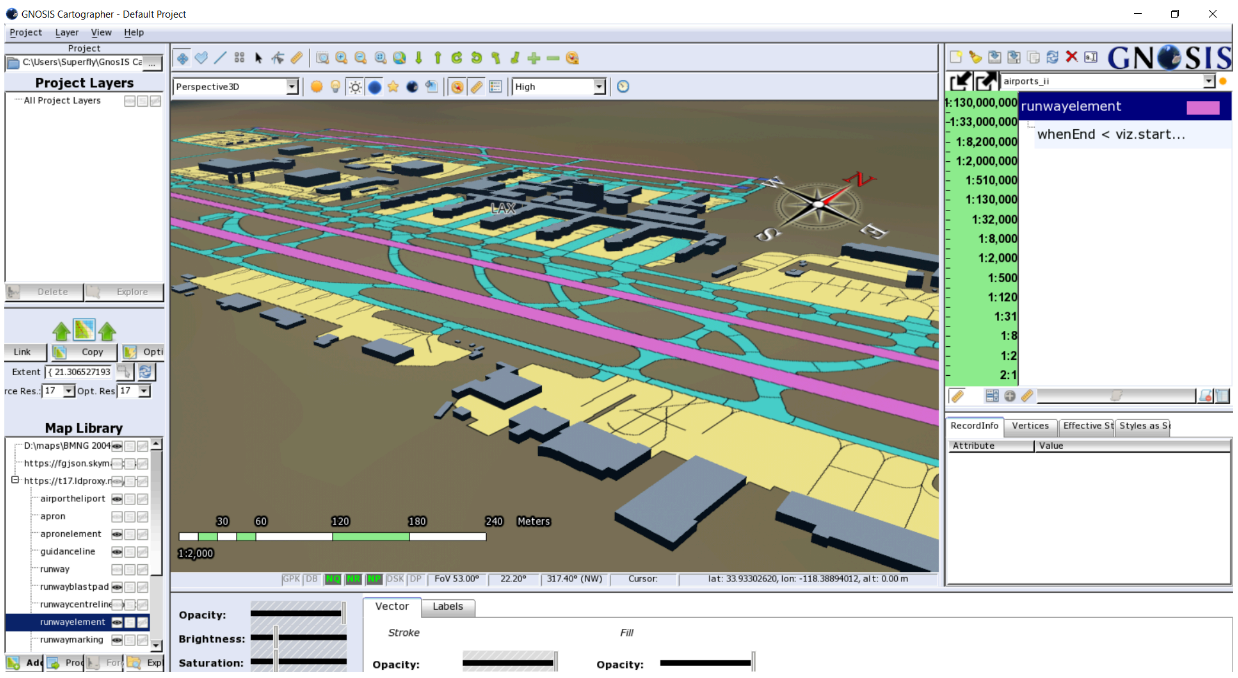


Figure 3 — Ecere's GNOSIS Cartographer client requests features from Interactive Instruments endpoint in CRS 3857 zoomed at airport location, loads and renders with styling

6.2.3. GeoSolutions (D116)

The D116 components implements a web client application that interacts with the OGC API Features endpoints to retrieve, render and reproject the JSON-FG data format. The web application is based on MapStore, a modular open source WebGIS framework written in JavaScript and ReactJS that supports building interactive application around maps and spatial data.

- repository with source code
- live demo

6.2.4. Structure of the web client application

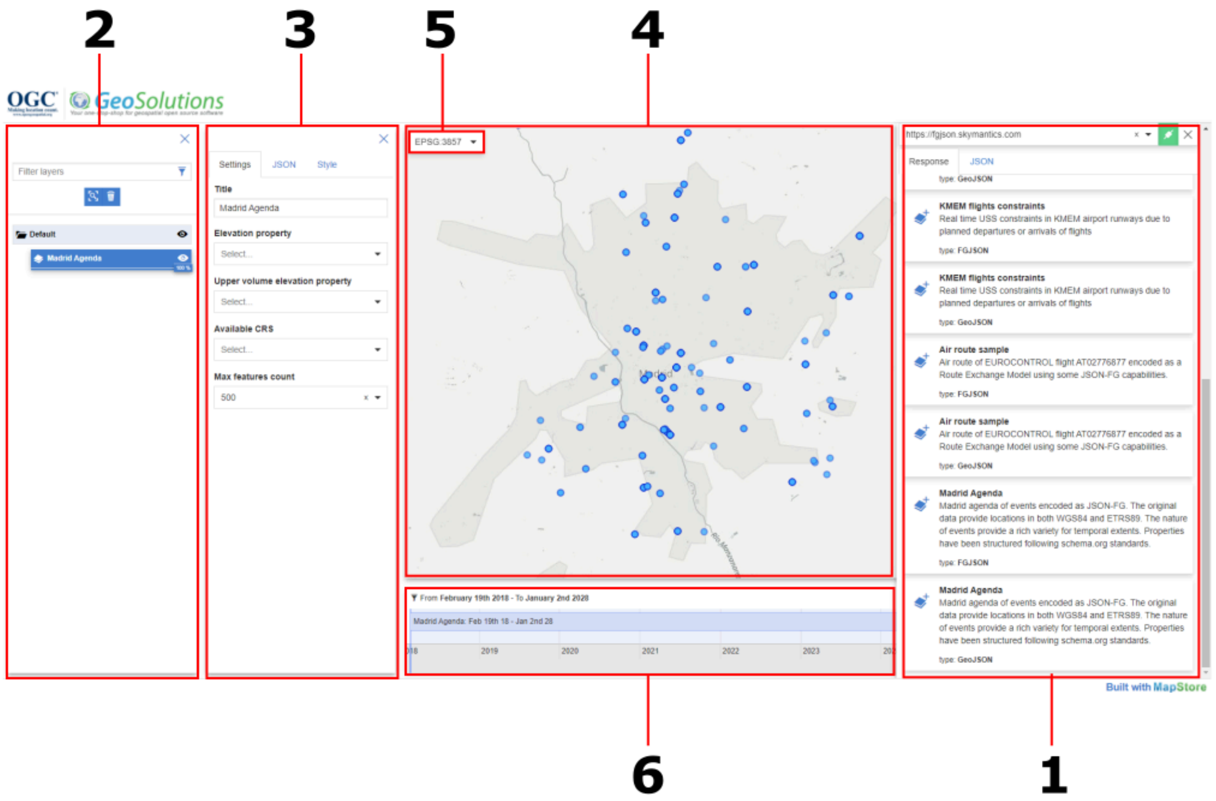
The implemented application was realized from a MapStore custom project and provides the following plugins:

1. Catalog panel to connect to an OGC API endpoint.
2. Layer tree to select and edit imported layers.

3. Layer setting panel to edit parameters:

    - elevation property: if specified uses this property as elevation value
    - upper volume elevation property: if specified uses this property as upper value
    - available CRS: select a different CRS to apply to the features collection
    - max features count: maximum number of feature to request
    - style: a dedicated panel to edit the features style (only type polygon supported)
    - JSON preview: a preview in JSON of the selected collection

4. Map viewer: Render the imported layers and it’s possible to change between 2D (OpenLayers) and 3D (Cesium) view.
5. Projection selector: Available only for the 2D view. Supports changing the rendered projection.
6. Time range selector: Available only for GeoJSON with the experimental property `when` that supports visualizing the time range of the imported collection and filter the features base on their own interval.



Title: Madrid Agenda

Elevation property: Select...

Upper volume elevation property: Select...

Available CRS: Select...

Max features count: 500

From February 19th 2018 - To January 2nd 2028

Madrid Agenda: Feb 19th 18 - Jan 2nd 28

https://fgjson.skymantics.com

type: GeoJSON

KMEM flights constraints
Real time USS constraints in KMEM airport runways due to planned departures or arrivals of flights
type: FGJSON

KMEM flights constraints
Real time USS constraints in KMEM airport runways due to planned departures or arrivals of flights
type: GeoJSON

Air route sample
Air route of EUROCONTROL flight AT02776877 encoded as a Route Exchange Model using some JSON-FG capabilities.
type: FGJSON

Air route sample
Air route of EUROCONTROL flight AT02776877 encoded as a Route Exchange Model using some JSON-FG capabilities.
type: GeoJSON

Madrid Agenda
Madrid agenda of events encoded as JSON-FG. The original data provide locations in both WGS84 and ETRS89. The nature of events provide a rich variety for temporal extents. Properties have been structured following schema.org standards.
type: FGJSON

Madrid Agenda
Madrid agenda of events encoded as JSON-FG. The original data provide locations in both WGS84 and ETRS89. The nature of events provide a rich variety for temporal extents. Properties have been structured following schema.org standards.
type: GeoJSON

Figure 4 — Structure of the web application

6.2.5. Technology interchange experiments (TIEs)

The TIEs documented in this ER focused on testing the following process:

1. Request JSON-FG collection from OGC API Features endpoints.
2. Display information of JSON-FG collection and retrieve all the available crs from the OGI API endpoint.
3. Use the features data projection described by the `coord-ref-sys` property.
4. Reproject the feature coordinates to CRS84 projection used internally by the client to correctly represent 2D/3D feature on the map.
5. Render the JSON-FG collection on a 2D/3D map.

6.2.5.1. Issue encountered and lessons learnt

This client implementation recognizes three different roles of the coordinates system:

1. Native CRS: The original CRS of the JSON-FG collection detected by the `coord-ref-sys` property.
2. Pivot CRS: The CRS84 CRS used internally by the client to manage different the 2D/3D visualizations.
3. Render CRS: the selected projection used by the map context.

The main issues encountered are:

- The client is not able to render the features on the map if the projections codes are not previously defined in the application configuration. The application could be improved to extend programmatically the projection definitions configuration to mitigate this issue.
- The `coord-ref-sys` is nested inside the feature object level instead of collection level could increase the complexity in the parsing phase.

In conclusion the subdivision of coordinate system roles in this implementation allowed the client to use a similar approach and utils related to GeoJSON format.

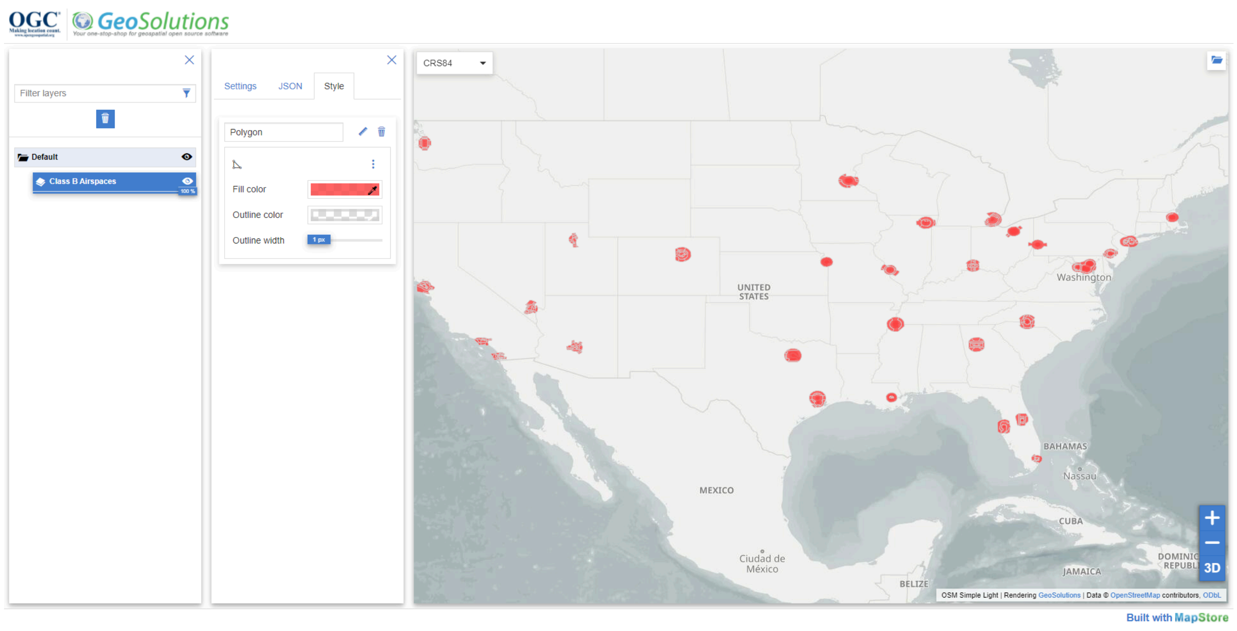
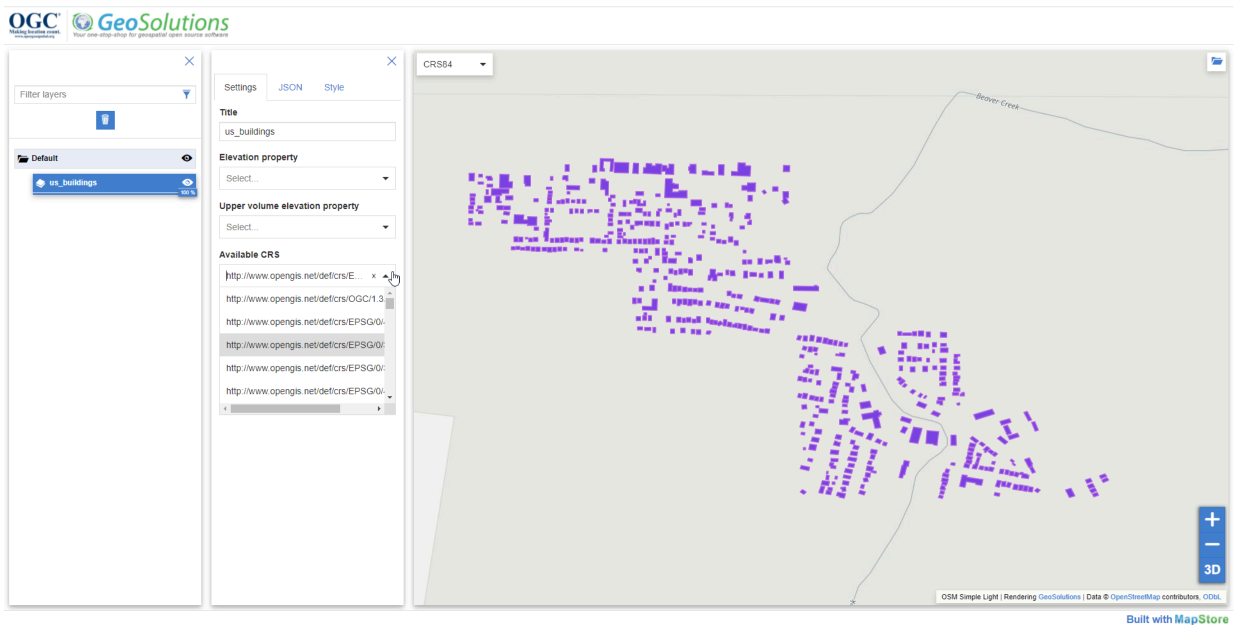


Figure 5 — MapStore client interacts with the interactive instruments service and shows a JSON-FG feature collection requested in EPSG:3857 and then rendered in CRS84 to show the difference between the request projection and the rendered projection



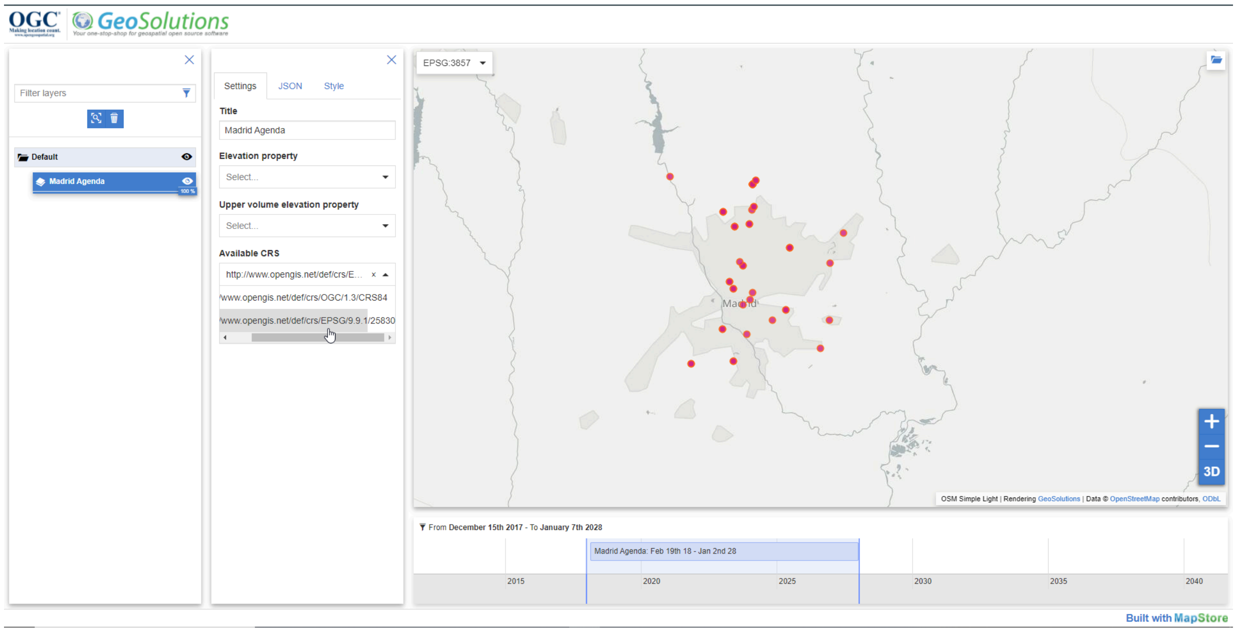
Title
us_buildings

Elevation property
Select...

Upper volume elevation property
Select...

Available CRS
http://www.opengis.net/def/crs/E...
http://www.opengis.net/def/crs/OGC/1.3
http://www.opengis.net/def/crs/EPSG/0/
http://www.opengis.net/def/crs/EPSG/0/
http://www.opengis.net/def/crs/EPSG/0/
http://www.opengis.net/def/crs/EPSG/0/

Figure 6 — MapStore client interacts with the Cubewrx service and shows all the available CRSs from the us_buildings JSON-FG collection


OGC
GeoSolutions
Filter layers
Default
Madrid Agenda
Settings
JSON
Style
EPSG:3857
Madrid
3D
OSM Simple Light | Rendering GeoSolutions | Data © OpenStreetMap contributors, ODbL
From December 15th 2017 - To January 7th 2028
Madrid Agenda: Feb 19th 18 - Jan 2nd 28
2015
2020
2025
2030
2035
2040
Built with MapStore

Title
Madrid Agenda

Elevation property
Select...

Upper volume elevation property
Select...

Available CRS
http://www.opengis.net/def/crs/E...
www.opengis.net/def/crs/OGC/1.3/CRS84
www.opengis.net/def/crs/EPSG/9.9.1/25830

Figure 7 — MapStore client interacts with the Skymantics service it shows all the available CRS from the Madrid Agenda JSON-FG collection

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
May 20, 2021		Panagiotis (Peter) A. Vretanos		initial version
Nov 18, 2021		Panagiotis Vretanos pvretano[at]cubew com		Apply OGC review comments.
Nov 12, 2021		Gobe Hobona		Update bibtex-file. bib
Nov 12, 2021		Panagiotis Vretanos pvretano[at]cubew com		Rewrite bibtex entry for OGC 21-017 since the current entry seems to cause asciidoc compilation problems.
Nov 11, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Tweaks.
Nov 11, 2021		Panagiotis Vretanos pvretano[at]cubew com		Minor tweaks.
Nov 10, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Document tweaks.
Nov 10, 2021		Panagiotis Vretanos		Document tweaks.

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
		pvretano[at]cubew com		
Nov 10, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Spelling.
Nov 10, 2021		Panagiotis Vretanos pvretano[at]cubew com		Fix typos.
Nov 2, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Merge branch 'master' of https://gitlab.ogc.org/ogc/T17-D027-GeoJSON-CRS-Analysis-of-Alternatives-ER
Nov 2, 2021		Panagiotis Vretanos pvretano[at]cubew com		Add contributors.
Nov 2, 2021		Patrick Dion patrick[at]ecere. com		Update 7-implementations. adoc
Nov 2, 2021		Panagiotis Vretanos pvretano[at]cubew com		ER tweaks.
Oct 25, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Reorg and updates to the er.
Oct 14, 2021		Ignacio Correas ignacio. correas[at]skymant com		Update 7-implementations. adoc
Oct 12, 2021		Clemens Portele portele[at]interactive-instruments.de		add d100
Oct 12, 2021		Stefano Bovio stefano.bovio[at]ge solutions.it		Add description of component d116 implementation

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
Oct 5, 2021		youcanmap stefano. bovio[at]geosolutionsgroup. com		add missing links to d116 component
Oct 5, 2021		youcanmap stefano. bovio[at]geosolutic com		add description of component d116
Sep 27, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Checkpoint — checking in more content based on the proposal written for the JSON-FG SWG.
Sep 21, 2021		Panagiotis Vretanos pvretano[at]cubew com		Add document where participants can add their descriptions of their implemenations.
Jun 1, 2021		Panagiotis Vretanos pvretano[at]cubewerx. com		Add a requirement for separate vertical CRS handling.
May 31, 2021		Panagiotis Vretanos pvretano[at]cubew com		Initial checkin.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: RFC 7946, *The GeoJSON Format*. Internet Engineering Task Force, Fremont, CA (2016). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7946.xml

2. Portele, C.: OGC Testbed-17: OGC Features and Geometries JSON Engineering Report. OGC 21-017r1, Open Geospatial Consortium (2022)