OGC Vector Tiles Pilot

WFS 3.0 Vector Tiles Extension Engineering Report

Table of Contents

Publication Date: 2019-02-11

Approval Date: 2018-12-13

Submission Date: 2018-11-09

Reference number of this document: OGC 18-078

Reference URL for this document: http://www.opengis.net/doc/PER/vtp-wfs3

Category: Public Engineering Report

Editor: Panagiotis (Peter) A. Vretanos

Title: OGC Vector Tiles Pilot: WFS 3.0 Vector Tiles Extension Engineering Report

---

OGC Engineering Report

COPYRIGHT

WARNING

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

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Summary

Feature data tiling, colloquially referred to as 'vector tiling', is a data delivery method that allows for large vector feature datasets to be systematically split into subsets or tiles [1]. This engineering report (ER) presents an extension specification for publishing of vector tiles data through an Application Programming Interface (API) that conforms to the emerging version 3.0 of the Web Feature Service (WFS) standard. The core of the WFS 3.0 standard offers direct fine-grained access to geospatial information at the feature level. The WFS standard specifies discovery and query operations for web services that publish feature data. Extensions to the WFS 3.0 Core API offer other capabilities such as transaction operations.

NOTE	This engineering report interchangeably uses both 'tiled feature data' and the colloquial term 'vector tiles'.

1.1. Requirements & Research Motivation

The research presented in this engineering report has been motivated by the increasing adoption of vector tiling within the geospatial industry.

The engineering report addresses deliverable D001 of the Vector Tiles Pilot. The Vector Tiles Pilot Call for Participation (CFP) outlines the deliverable as follows.

"D001: WFS 3.0 Extension Engineering Report - A WFS 3.0 Extension written as a draft OGC standard. The report shall take into account the WFS extension approach documented in Testbed 13. The WFS 3.0 standard is written as a RESTful reusable OpenAPI set of components with responses in JSON and HTML."

1.2. Prior-After Comparison

To date, the only OGC web service standard that offered a tiling approach was the Web Map Tile Service (WMTS) standard. Recently, a tiling storage mechanism has also been incorporated into the OGC GeoPackage standard. In contrast however, the WFS standard and its extensions do not currently offer a tiling mechanism.

Recognizing the increasing adoption of vector tiling across the industry, OGC produced a vector tiling engineering report in the OGC Testbed-13 project [2]. The OGC Testbed-13 Vector Tiles Engineering Report described the evaluation of existing vector tiling solutions and reported on WFS vector tiles experimentation conducted during the testbed. Amongst other recommendations, the engineering report recommended the extension of the WFS standard for support vector tiling.

1.3. Recommendations for Future Work

Future work should explore:

- The potential for asynchronous support for the Vector Tiles extension of WFS 3.0 using the OGC PubSub specification.

- The development of Executable Test Suites for compliance testing of implementations of the Vector Tiles extension of WFS 3.0.

1.4. Document contributor contact points

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

Contacts

Name	Organization
Panagiotis (Peter) A. Vretanos (editor)	CubeWerx Inc.
Clemens Portele	interactive instruments GmbH
Kalimar Maia	Mapbox
Brian Davidson	Mapbox
Luis Bermudez	Open Geospatial Consortium
Gobe Hobona	Open Geospatial Consortium
Andrea Aime	GeoSolutions
Chia-Cheng (Ricky) Lin	GIS.FCU
Jerome St-Louis	Ecere

1.5. Foreword

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

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

Chapter 2. References

The following normative documents are referenced in this document.

- OGC: OGC 06-121r9, OGC Web Services Common Standard, 2010 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 09-025r2, OGC Web Feature Service 2.0 Interface Standard – With Corrigendum, 2014 [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- OGC: OGC 07-057r7, OpenGIS Web Map Tile Service Implementation Standard, 2010 [http://portal.opengeospatial.org/files/?artifact_id=35326]
- OGC: OGC 17-069, OGC Web Feature Service 3.0 - Part 1: Core (Draft Standard), 2018 [https://cdn.rawgit.com/opengeospatial/WFS_FES/3.0.0-draft.1/docs/17-069.html]
- OGC: OGC 15-113r3, Volume 1: OGC® CDB Core Standard: Model and Physical Data Store Structure Standard, 2017 [https://portal.opengeospatial.org/files/?artifact_id=72712]

Chapter 3. Terms and definitions

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

- feature

  ```
  abstraction of real world phenomena [ISO 19101-1:2014]
  ```

- feature collection; collection

  ```
  a set of features from a dataset
  ```

- tile

  ```
  a rectangular representation of geographic data, often part of a set of such
  elements, covering a spatially contiguous extent which can be uniquely defined by a
  pair of indices for the column and row along with an identifier for the tile matrix
  (adapted from OGC 07-057r7)
  ```

- tiling scheme

  ```
  a scheme that defines how space is partitioned into individual tiles. It defines
  the coordinate reference system, the geometric properties of a tile, which space a
  uniquely identified tile occupies, and reversely which unique identifier
  corresponds to a space satisfying the geometric properties to be a tile.
  ```

- tileset

  ```
  a series of tiles of a given tiling scheme containing encoded vector features from
  one or more feature collections
  ```

3.1. Abbreviated terms

- API Application Programming Interface
- MVT Mapbox Vector Tiles
- VTP Vector Tiles Pilot
- WFS Web Feature Service
- WMTS Web Map Tile Service

Chapter 4. Overview

Clause 5 describes two Vector Tile APIs for WFS 3.0; a 'Direct tile access path API' and a 'Feature access path API'. Both are primarily designed to support the portrayal or visualization use case. The first API, like the WMTS, is based on tiled feature data being organized into regularly gridded tiles arranged into a pyramid of tile matrices at a fixed set of resolutions (i.e. tile matrix set). The latter API treats features the way the WFS has always treated features and views vector tiles as simply another output format that a server may support. The feature access path API is designed to leverage the flexible query capabilities of the WFS to generate a vector tile response on the server side leaving the styling to be done on the client side.

In Clause 6 each participant describes the components that they developed and tested for the Vector Tiles Pilot. WFS 3.0 servers implementing one or both of the APIs described in Clause 5 were provided by the following participants:

- CubeWerx
- Ecere
- GeoSolutions
- interactive instruments GmbH
- Mapbox

Clients able to interact with these servers were provided by the following participants:

- Ecere
- GIS FCU
- Mapbox
- GeoSolutions

The following table summarizes the result of the client and server interactions i.e. Technology Integration Experiments (TIE):

Table 1. WFS TIE Matrix

CLIENTS	**SERVERS**				
	CubeWerx	Ecere	GeoSolutions	interactive instruments	Mapbox
Ecere	Connects	Connects	Connects	Connects	Connects
GIS FCU	Connects	Connects	Connects	Connects	
Mapbox	Connects	Connects	Connects	Connects	Connects

Chapter 5. Vector Tiles API for WFS 3.0

5.1. Introduction

OGC Web Feature Service (WFS) is that standard mostly used for disseminating and querying vector feature data on the Web. WFS 3.0 is a complete rewrite of previous versions, focusing on a simple RESTful core specified as reusable OpenAPI components with responses in JavaScript Object Notation (JSON) and Hypertext Markup Language (HTML).

Feature data tiling (colloquially referred to as vector tiling) is a proven mechanism for optimally storing, delivering and visualizing vector data from arbitrary large data sets, used to achieve high performance in Geographic Information Systems (GIS) for several decades. Multi-resolution tile sets leverage the fact that a typical client is either visualizing a large geospatial extent at low resolution, or a limited geospatial extent at high resolution. Furthermore, tiled feature data enables efficient caching of data and provides clients with the option of displaying the best currently available data while awaiting requests for more refined tiles.

The Mapbox Vector Tile (MVT) specifications is a way to encode vector tiles which has garnered widespread support and adoption and has brought vector tiles to mainstream web mapping applications.

Extending WFS 3.0 to support vector tiles, including those based on the MVT specification and GeoJSON Vector Tile (JVT) output formats, is an evolution of the standard that will help deliver geospatial data in a modern optimized mechanism.

This section provides two interfaces that can be used for accessing vector tiles through WFS 3.0 in Mapbox Vector Tile format. The first interface focuses on direct access to the tiles as defined by a tiling scheme, while the second one focuses on features and is more similar to a typical WFS 3.0 request.

Requirements Class: Core	
http://www.opengis.net/spec/wfs/3.0/vt/req/core	
Target type	Token
Requirement 1	http://www.opengis.net/spec/wfs/3.0/vt/req/core/direct-tile-access-path-api
Requirement 2	http://www.opengis.net/spec/wfs/3.0/vt/req/core/feature-access-path-api

The relationship between WFS feature collections and tiling schemes is one-to-many. That is to say that tiles for a specific WFS feature collection can be offered using one or more tiling schemes.

5.2. Direct tile access path API

5.2.1. Introduction

This interface focuses on accessing individual tiles directly. It provides resources describing

supported tiling schemes, as well as templated paths to access individual tiles.

It assumes the following:

- The tiling scheme ('tile matrix set' to use WMTS terminology) defines how space is partitioned into individual tiles.
- In the context of the pilot, as in WMTS, all tiling schemes definitions uniquely identify tiles using a three-part identifier made up of {level} (zoom), {row} (vertical) and {col} (column: horizontal).
- The level parameter references a particular zoom level of a fixed spatial resolution, akin to the typical use of an individual 'tile matrix' in WMTS.
- With the exception of the GNOSISGlobalGrid implemented by Ecere, all other tiling schemes were regular grids of tiles.

Notes:

- An implementation could either generate tiles on-the-fly or serve an existing pre-generated set of tiles.
- A tiling scheme does not need to be a quad-tree tile pyramid, e.g. GlobalCRS84Pixel and GlobalCRS84Scale Well Known Scale Sets do not have a constant factor of 2 between zoom levels.
- In the context of the pilot, only a read-only interface has been considered.

Requirement 1	http://www.opengis.net/spec/wfs/3.0/vt/req/core/direct-tile-access-path-api An implementation asserting support of the Direct Tile Access Path API shall support the behavior described in Clause 5.2 of the Vector Tiles extension of WFS 3.0 .

5.2.2. Preliminaries

OPTIONS & HEAD method

"Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin" [1: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS]. It is recommended that the OPTIONS & HEAD methods be available on each resource in order to support CORS preflight request and to be able to determine the supported representations, methods and response headers.

Content negotiation

According to the WFS specification (clause 7.8): "...any server that supports multiple encodings will have to support a mechanism to mint encoding-specific URIs for resources in order to express links, for example, to alternate representations of the same resource. This document does not mandate

any particular approach for how this is supported by the server." Although it lists f as a parameter that is used on many paths in this email, this is but one possibility. Other methods, such as extensions, may also be used. It is up to the server.

OGC Tile Matrix Set Standard Candidate (17-083)

In general, this is the schema that should be used for the definition of a tiling scheme. However, in this pilot the following changes were made:

- the coordinates for the topLeftCorner, as well as the lowerCorner and upperCorner of the boundingBox, are changed from a string to an array of double-precision floating point numbers
- adding a "links" object to the tilingScheme definition so that templates, for accessing the tiles, can be included

```
   "links" : [
      {
         "rel" : "tiles",
         "type" : "application/geo+json",
         "title" : "Tile in GeoJSON. The link is a URI template where
{level}/{row}/{col} is the tile based on the tiling scheme.",
         "href" : "https://services.interactive-
instruments.de/vtp/daraa/tiles/default/{level}/{row}/{col}?f=json",
         "templated" : "true"
      },
      {
         "rel" : "tiles",
         "type" : "application/vnd.mapbox-vector-tile",
         "title" : "Mapbox Vector Tile. The link is a URI template where
{level}/{row}/{col} is the tile based on the tiling scheme.",
         "href" : "https://services.interactive-
instruments.de/vtp/daraa/tiles/default/{level}/{row}/{col}?f=mvt",
         "templated" : "true"
      }
   ]
```

In the content that follows this clause, any mention of OGC 17-083 should be interpreted to mean OGC 17-083 with the changes proposed above.

The following is a partial example description for the tiling scheme based on the GoogleMapsCompatible Well Known Scale Set:

```
   {
     "identifier": "smerc",
     "title": "Spherical Mercator projection with quad-tree zoom levels, compatible
with Google Maps®, Bing Maps®, etc.",
     "boundingBox": {
       "lowerCorner": [
         -20037508.34278924,
         -20037508.34278924
       ],
       "upperCorner": [
         20037508.34278924,
         20037508.34278924
       ],
       "crs": "http://www.opengis.net/def/crs/EPSG/0/3857"
     },
     "supportedCRS": "http://www.opengis.net/def/crs/EPSG/0/3857",
     "wellKnownScaleSet":
"http://www.opengis.net/def/wkss/OGC/1.0/GoogleMapsCompatible",
     "tileMatrices": [
       {
         "identifier": "0",
         "scaleDenominator": 559082264.028717,
         "tileWidth": 256,
         "tileHeight": 256,
         "matrixWidth": 1,
         "matrixHeight": 1,
         "topLeftCorner": [
           -20037508.3427892,
           20037508.3427892
         ]
       },
       {
         "identifier": "1",
         "scaleDenominator": 279541132.014358,
         "tileWidth": 256,
         "tileHeight": 256,
         "matrixWidth": 2,
         "matrixHeight": 2,
         "topLeftCorner": [
           -20037508.3427892,
           20037508.3427892
         ]
       },
    ...
```


Recommendations for further changes to OGC 17-083

The following were noted during the pilot as further recommendations to the Tile Matrix Set Candidate Standard:

- Supporting tiling schemes with a variable number of columns would enable supporting global grids making adjustment for latitude regions, such as Ecere's GNOSIS Global Grid as well as the CDB tiling grid. This was initially suggested in Testbed-13 (http://ogc.standardstracker.org/show_request.cgi?id=518) and discussed by the Web Map Service (WMS) Standards Working Group (SWG) at Technical Committee (TC) meetings in 2018. An example of how this could be described can be found here[http://maps.ecere.com/hms/tiles/GNOSISGlobalGrid?f=json] with a 'variableWidths' parameter.

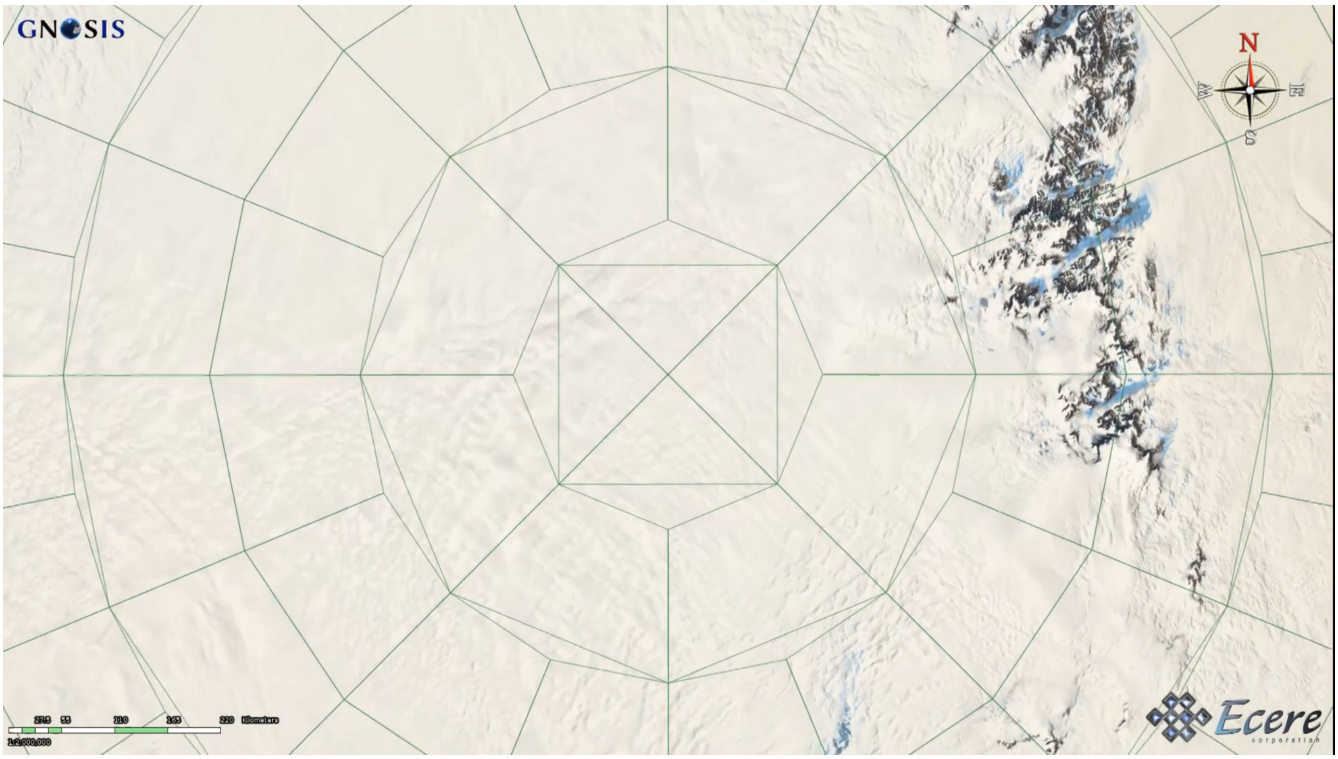

Figure 1. GNOSIS Global Grid at the South Pole

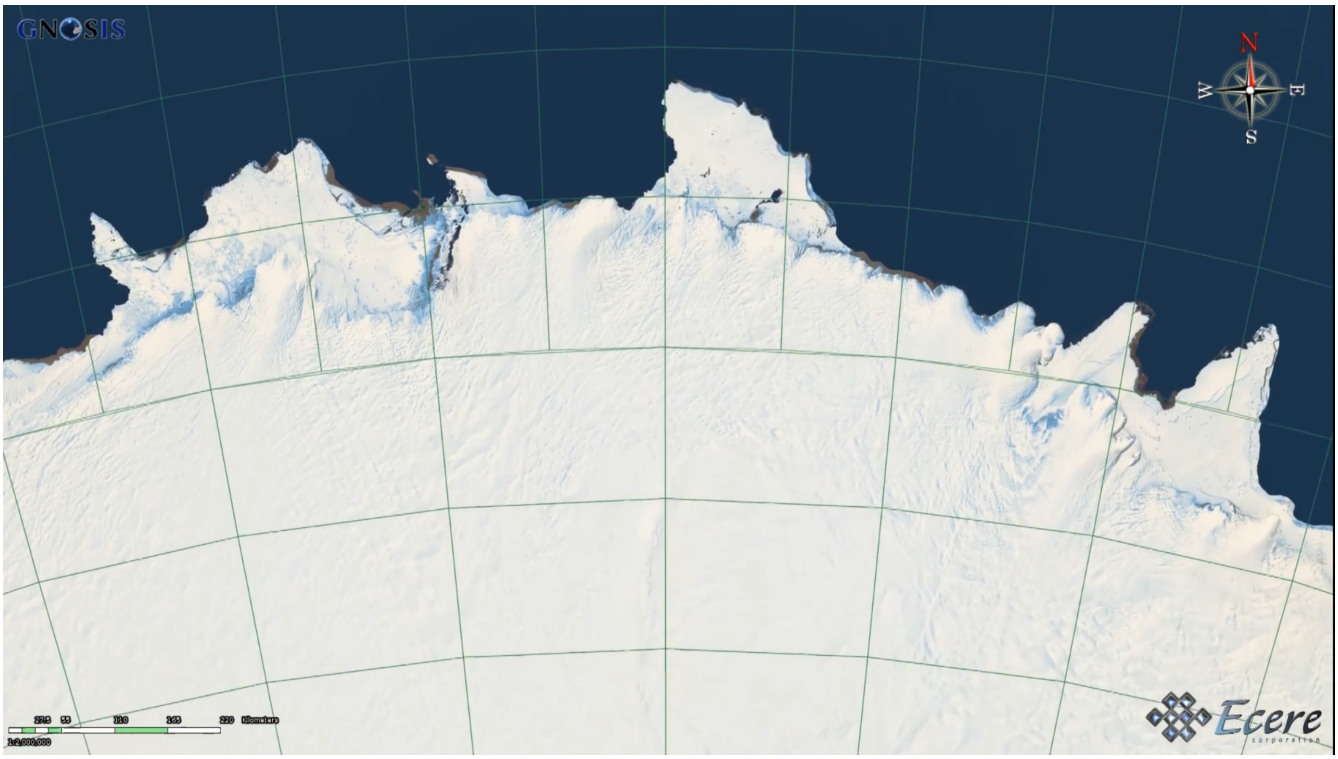

Figure 2. GNOSIS Global Grid at the coast of Antarctica

- The point of origin being fixed to the top-left corner (and the associated 'topLeftCorner' parameter), and always starting counting rows from the top left is unnecessarily restrictive. Using a more generic 'origin' should be considered, with the option to start counting rows south to north.
- There is little value in having identifiers allowed to be textual identifiers, and numeric (integral) zoom levels would be preferable. A potential use of textual identifiers could be to describe completely different data sets stored at different zoom levels, but this is problematic for mixing levels, as is useful in 3D views where tiles further away from the camera can be of lower resolution. Instead, different data sets can be handled by portrayal rules selecting alternate layers.
- The 'TileMatrix' and 'TileMatrixSet' names are a source of confusion, with people unfamiliar with WMTS often even confusing the two. Renaming 'tile matrix' to 'level' or 'ZoomLevel' could make it a lot more intuitive, while opening the possibility of describing something else other than a regular grid e.g. variable number of tiles across, or a Discrete Global Grid System (DGGS) zoom level. Similarly, renaming a 'TileMatrixSet' to 'TilingScheme' would also open up the possibility of describing alternate tiling mechanisms.
- Renaming 'TileMatrixWidth' to 'NumCols' or 'ColsCount' could avoid the dependence on the tile matrix concept, while avoiding a potential confusion with the width of a single tile. Similarly, renaming 'TileMatrixHeight' to 'NumRows' or 'RowsCount' could be more intuitive.

5.2.3. /tiles path

The "/tiles" path may be used to retrieve a list of tiling schemes that are valid for all collections offered by a server.

The response document shall contain a list of tiling scheme identifiers with a "links" object for hypermedia controls. The "links" object should include link(s) (rel="tilingScheme") to the tiling scheme definition (as per OGC 17-083) of the corresponding identifier. There could be more than one link if the tiling scheme definition is available in various formats (JSON, XML, HTML, etc.).

Media types: application/json

NOTE	The specific output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the Vector Tiles Pilot supported the standard HTTP header mechanism for content negotiation and also generally supported the 'f' parameters for embedded links. In the Pilot, the value "json" was used to indicate that a JSON representation should be generated.

The following JSON-schema fragments defined the schema of the resource accessible via the "/tiles" path:

```
  "tilingSchemes": {
     "type": "array",
     "items": {
        "type": "object",
        "required": [ "identifier", "links" ],
        "properties": {
           "identifier": {
              "type": "string"
           }
        "links": {
           "type": "array",
           "items": {
              "$ref": "#/components/schemas/link"
           }
        }
     }
  }

  "link": {
     "type": "object",
     "required": [
        "href"
     ],
     "properties": {
        "href": {
           "type": "string"
        },
        "rel": {
           "type": "string"
        },
        "templated": {
           "type": "boolean"
        },
        "type": {
           "type": "string"
        },
        "hreflang": {
           "type": "string"
        }
     }
  }
```

The following illustrates an example response document:

```
{
   "tilingSchemes": [
      {
         "identifier": "default",
         "links": [
            {
            "href": "https://services.interactive-
instruments.de/vtp/daraa/tiles/default?f=json",
            "rel": "tilingScheme",
            "title": "Google Maps Tiling Scheme",
            "type": "application/json"
            }
         ]
      }
   ]
}
```


5.2.4. /tiles/{tilingSchemeId} path

This resource path accesses the definition of a specific tiling scheme specified by the "{tilingSchemeId}" substitution variable.

The response schemas shall be the standard tile matrix set schema defined in OGC 17-083 with the changes noted at the beginning of this clause.

Media types: application/json

NOTE	The specific output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the Vector Tiles Pilot supported the standard HTTP header (i.e. Accept) mechanism for content negotiation and also generally supported the 'f' parameters for embedded links. In the Pilot, the value "f=json" was used to indicate that a JSON representation should be generated.

5.2.5. Tiles access path

Specific vector tiles may be accessed using the tiles access path. The tiles access path begins with the path elements "/tiles/{tilingSchemeId}" and includes the three substitution variables {level}, {row} and {col}. No specific order or form was assigned to the trailing portions of the direct access path but the general form used in the Vector Tiles Pilot was:

- /tiles/{tilingSchemeId}/{level}/{row}/{col}

Media types: application/vnd.mapbox-vector-tile, application/geo+json

NOTE	The specific vector tile output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the VT pilot support the standard HTTP header mechanism for content negotiation and also generally support the 'f' parameters for embedded links. In the VT pilot the values "f=mvt", "f=json" or "f=jvt" have been used to indicate the Mapbox Vector Tile or GeoJSON vector tile formats should be generated.

Resolving the tiles access path gets the tile at the specified level, row and col for the specified tiling scheme. Within the tile are rendered all the relevant features from all collections offered by the service unless the "collections" parameter is used to limit the scope of what collections are rendered into the tile. The following JSON-Schema fragment defines the "collections" parameters:

```
  "collections_QParam": {
     "name": "collections",
     "in": "query",
     "description": "List of collections to be rendered into a tile",
     "required": false,
     "allowEmptyValue": false,
     "schema": {
        "type": "array",
        "items": {
           "type": "string",
           "enum": [ ... list of valid collection ids ... ]
        }
     },
     "style": "form",
     "explode": false
  }
```

5.2.6. Collection-level metadata

The "/collections/{collectionId}" path gets collection-level metadata about the specified collection.

The "links" object in the metadata may contain link(s) (rel="tilingSchemes") that point to the "/collection/{collectionId}/tiles" resource.

The "links' object in the metadata may also contain template(s) (rel="tiles") for accessing tiles using the substitution variables {tilingSchemeId}, {level}, {row} and {col}. Rendered into the tile shall be all relevant features from the specified collection identified by the "{collectionId}" substitution variable.

Media types: application/json

NOTE	The specific vector tile output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the Vector Tiles Pilot supported the standard HTTP header mechanism for content negotiation and also generally supported the 'f' parameters for embedded links. In the Pilot, the value "f=json" was used to indicate that a JSON representation should be generated.

The following **optional** content may also be included in the metadata to identify the specific tiling schemes available for the corresponding collection.

The collection-level metadata may **optionally** contain a "tilingSchemes" array that lists the valid tiling scheme identifiers for the corresponding collection.

The "links" object within the "tilingSchemes" object shall contain a template (rel="tilingSchemes") for accessing the tiling scheme definition(s). This template shall use the "{tilingSchemeId}" substitution variable whose valid values are taken from the "tilingSchemes" array.

The following JSON-Schema fragment extends the "collectionInfo" definition to include the necessary "tilingSchemes" structure:

```
  "collectionInfo": {
     "type": "object",
     "required": [ "links", "name" ],
     "properties": {
        "name": {
           "type": "string"
        },
        "title": {
           "type": "string"
        },
        "description": {
           "type": "string"
        },
        "links": {
           "type": "array",
           "items": {
              "$ref": "#/components/schemas/link"
           }
        },
        "tilingSchemes: {
           "type": "array",
           "items": {
              "$ref": "#/components/schemas/link/tilingScheme"
           }
        },
        "extent": {
           "$ref": "#/components/schemas/extent"
        },
        "crs": {
          "type": "array",
          "items": {
             "type": "string"
          },
          "default": ["http://www.opengis.net/def/crs/OGC/1.3/CRS84"]
        }
     }
  }
```

5.2.7. /collections/{collectionId}/tiles path

The behavior of this path is identical to the "/tiles" path except that only the tiling schemes available for this collection are listed.

NOTE	A tiling scheme listed at the `/tiles` level (i.e. a tiling scheme that is valid for all collections) should also be repeated at this level. This way a client has a simple and reliable way to list valid tiling schemes regardless of the access path without having to cross-reference information from multiple paths to get a complete list of available tiling schemes.

Media types: application/json

NOTE	The specific output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the Vector Tiles Pilot supported the standard HTTP header mechanism for content negotiation and also generally supported the 'f' parameters for embedded links. In the Pilot, the value "json" was used to indicate that a JSON representation should be generated.

5.2.8. /collections/{collectionId}/tiles/{tilingSchemeId} path

The behavior of this path is identical to the behavior of the "/tiles/{tilingSchemeId}" path.

5.2.9. Tile access path

The behavior of this path is similar to the behavior of the "/tiles{tilingSchemeId}/{level}/{row}/{col}" path except that only features of the corresponding collection identified by the "{collectionId}" substitution variable are rendered into the tile.

NOTE	The behavior is identical to accessing the "/tiles/{tilingSchemeId}/{level}/{row}/{col}?collections={collectionId}" path.

Media types: application/vnd.mapbox-vector-tile, application/geo+json

NOTE	The specific output format may be selected using whatever content negotiation mechanism a server supports. Generally, servers in the Vector Tiles Pilot supported the standard HTTP header mechanism for content negotiation and also generally supported the 'f' parameters for embedded links. In the Pilot, the values "mvt", "jvt"/"json" were used to indicate that Mapbox Vector Tile and GeoJSON representations should be generated.

5.3. Feature access path API

5.3.1. Introduction

The Feature Access Path API, focuses on the feature as the main resource. This API leverages the flexible query capabilities of WFS 3.0 to generate an MVT response. The MVT response is treated

like other output formats (e.g. JSON) supported by a WFS.

Requirement 2	http://www.opengis.net/spec/wfs/3.0/vt/req/core/feature-access-path-api An implementation asserting support of the Feature Access Path API shall support the behavior described in Clause 5.3 of the Vector Tiles extension of WFS 3.0

5.3.2. Path

Using the feature access path, MVT tiles are accessed via the /collection/{collectionId}/items resource already defined in the WFS 3.0 Core specification.

The path retrieves features of a feature collection {collectionId}.

Every feature in a dataset belongs to a collection. A dataset may consist of multiple feature collections. A feature collection is often a collection of features of a similar type, based on a common schema.

5.3.3. Parameters

Standard WFS 3.0 query parameters

The standard WFS 3.0 query parameters (bbox, time, parameters for filtering) may be used to access a vector tile through the feature access path API.

Parameters from the CRS extension

Servers may also choose to offer Coordinate Reference System (CRS) support as defined in the CRS extension for WFS 3.0.

This allows clients to request that features be projected into the requested CRS before being encoded as a vector tile.

The crs parameter may be used to assert the desired target CRS.

Parameter for tile attribution

Mapbox Vector Tiles support attribution of tiles. Selection of attribution, however, is not currently supported in the WFS 3.0 core specification. To enable filtering of attributes, it is suggested a property selection extension named **properties** be offered.

The value of the **properties** parameter is a comma-separated list of feature properties (collections) that should be included in the vector tile.

An MVT-encoded file should only contain features properties that have values. As a result, only a subset of the list of properties specified using the **properties** parameter may appear in an MVT-encoded file.

If not specified, the default value for the **properties** parameter shall be the list of all feature properties.

The following example indicates that the tile should, at most, contain the properties A, C, D and F. From this set of properties only those properties that actually have values should be included in the MVT.

```
.../collections/{collectionId}/items?...&properties=A,C,D,F&...
```

NOTE	The **properties** parameter should also be defined on each direct tile access path to allow a list of tile attributes to be specified. For example:

```
.../collections/{collectionId}/tiles/{tilingSchemeId}/{level}/{row}/{col}?properties=A
,C,D,F
```

clipBox parameter

A parameter behaving like BBOX, with the understanding that the geometry will be clipped.

Resolution parameter

The `resolution` parameter specifies a value(s), in "number of units per pixel" where the units are defined by the CRS. Using the resolution, a specified bounding box and knowing the CRS, a server can compute the size of a vector tile.

The value of the resolution parameter is either a single value representing the same resolution along each axis or a set of values corresponding to the resolution along each axis.

Example:

Consider a client that wishes to generate a 1024x1024 pixel-dimensioned vector tile using the bounding box 32.6173,36.0938,32.6353,36.1154 in WGS84. The client would compute the resolution as:

```
(32.6353-32.6173)/1024 = 0.00002
(36.1154-36.0938)/1024 = 0.00002
```

and could then make a request such as:

```
http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections/agricult
ureSrf/items?f=mvt&bbox=38.7952,-77.1336,39.0062,-76.9102&resolution=0.00002
```

to get the desired tile.

ALTERNATIVE: A more intuitive approach might be to have a `scaleDenominator` parameter. The parameter would take a single value or set of values representing the scale denominator value

along each axis. In this example the value would be `scaleDenominator=50000`.

Summary of query parameters

The following table summarizes the query parameters that may be used.

Table 2. Query Parameters for Accessing Vector Tiles via the Feature Access Path

Parameter Name	Description
bbox	Only features that have a spatial geometry that intersects the bounding box shall be included in the vector tile response.
time	Only features that have a temporal geometry that intersects the timestamp or time period SHALL be included in the vector tile response.
{prop}	Only features that satisfy the specified property filter shall be included in the vector tile response.
Sizing parameters	
resolution	A parameter that, in combination with the bbox parameter and CRS, sets the size of the vector tile.
CRS parameters	
bbox-crs	Asserts the CRS used to specify the
crs	Asserts a specific WFS-supported CRS transformation to be applied to the compatible geometries of the features to be included in the vector tile response.
Tile attribution	
properties	Comma-separated list of feature properties to include in the vector tile response.

NOTE The implied logical operator between query predicates is AND.

The following example illustrates a URL for accessing a vector tile using the feature access path.

```
.../collection/buildings/items?bbox=bbox=41.199844,-83.859060,45.618364,-
75.619314&crs=http://www.opengis.net/def/crs/epsg/0/3257&properties=address,height
```

5.3.4. Requesting vector tiles output

The only requirement for supporting the Feature Access Path API is that the server advertises that it supports a vector tiles encoding (such as Mapbox Vector Tiles or GeoJSON) as an output format in its OpenAPI/Capabilities document.

The MVT output format can be requested as defined in the WFS 3.0 Core specification. That is, by setting the Accept header appropriately or using the URL mechanism for specifying output formats supported by the server and specified in its OpenAPI document (e.g. using the .mvt extension or

setting an "f" parameter to "mvt" or "application/vnd.mapbox-vector-tile").

5.3.5. Multiple collections rendered into a single layer

A common usage pattern with vector tiles is to render features from multiple collections into a single response tile. However, the current WFS 3.0 specification does not define a mechanism for allowing simultaneous access to an enumerated set of feature collections in a single request.

To support this capability, the following extensions to WFS 3.0 Core were defined:

- First, a top level `/items` path was defined (analogous to the `/tiles` path for tile access) to represent the entire set of features from all collections offered by a server.
- Second, a parameter named `collections` was defined to allow a client to limit the scope of a request to the list of collections specified by the parameter

The value of the `collections` the parameter is a comma-separated list of collection identifiers enumerating the set of feature collections (offered by a server) that should be accessed by a request. Only features from the enumerated collections shall appear in the server's response.

If the `collections` parameter is not specified, then the scope of the request shall be all the features from all collections offered by the server.

All the usual predicates defined in WFS 3.0 Core (i.e. `bbox`, `time` and `property` filter elements) may also be specified on the `/items` resource. In the case of property filtering elements, if the specified property is not defined for a specific feature, then that predicate shall be ignored.

Example:

http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/items?f=mvt&collections=agricultureSrf,militarySrf&bbox=32.6173,36.0938,32.6353,36.1154

5.3.6. Response

The response to retrieving features in MVT format shall be a single MVT into which the result feature(s) shall be rendered. Unlike the direct tile access path that assumes tiles of fixed size, for the feature access path the size of the tile is determined by the extent of the request and the requested size/resolution.

Chapter 6. Component Implementations

6.1. Introduction

To support experimentation, an architecture was designed to cover the three main server client relationships that are common in OGC use cases as illustrated in Figure 3.

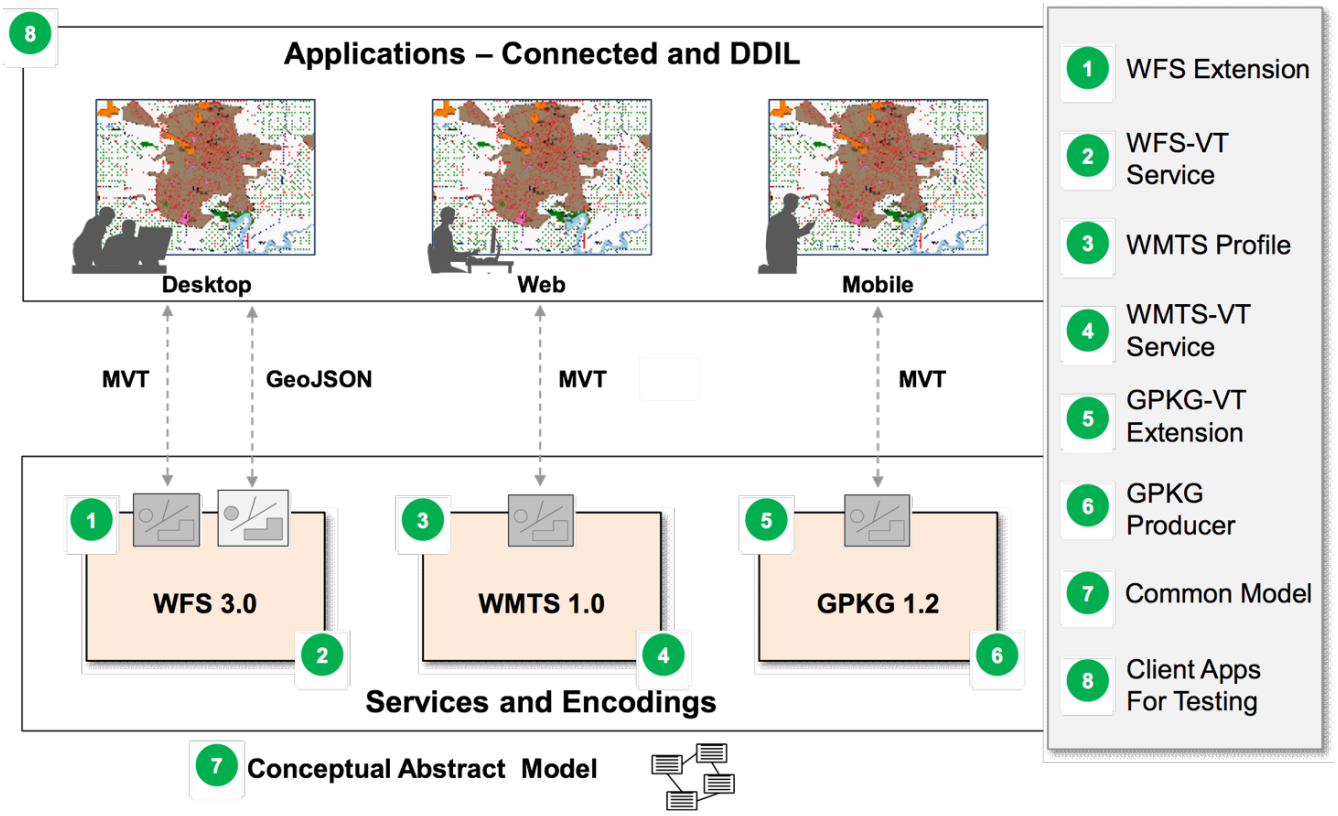

Figure 3. Vector Tiles Pilot Architecture

The architecture addresses the application of tiled feature data in each of the following client-server relationships to simultaneously enable feature data tiling across the relevant suite of OGC standards.

- Desktop Client → Web Feature Service (WFS) 3.0
- Web Client → Web Map Tile Service (WMTS) 1.0
- Mobile Client → GeoPackage 1.2

This clause provides a detailed description of the various server, client and WFS 3.0-related components that were implemented and deployed for the Vectors Tiles Pilot.

The clause also includes a Technology Integration Experiment matrix that outlines the various implementations of servers and clients for the WFS

6.2. WFS Servers

6.2.1. CubeWerx

Servers

CubeWerx deployed two servers for the Vector Tiles Pilot. A development server and a production server. The WFS 3.0 landing pages for the servers can be found at:

- production: http://tb14.cubewerx.com/cubewerx/cubeserv/vt/wfs/3.0.0/vtpilotwfs?f=json
- development (daraa): http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa
- development (zataari): http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/zaatari

Implementation and status

The CubeWerx servers implement both the direct tile access path API and the feature access path API offering both Mapbox Vector Tile and GeoJSON output in both cases.

As recommended in the Mapbox Vector Tile specification, the data is clipped with a small buffer around the tile boundary. Geometries in GeoJSON vector tiles are not clipped.

By default, Mapbox Vector Tile (MVT) and GeoJSON (JVT) outputs include all feature properties. However, the list of feature properties that are included in the tiles can be pruned using the `properties` parameter. The following example, that generates a GeoJSON vector tile, illustrate the use of the `properties` parameter.

Example: http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections/agricultureSrf/items?count=10&outputFormat=jvt&properties=f_code

Besides the standard query parameters defined by the WFS 3.0 specification, the CubeWerx server supports all the query parameters defined in the Vector Tiles API for WFS 3.0 clause except for the `clipBox` parameter.

For the Vector Tiles Pilot, the standard tiling scheme called the "GoogleMapsCompatible" in the WMTS standard was deployed using the identifier `smerc`. The CubeWerx server, however, can deploy any tiling scheme encoding using XML or JSON as described in the draft Tile Matrix Set standard. New tiling schemes can be added to the server by POSTing a representation of tiling scheme to the `/tiles` path.

The OpenAPI definitions generated by the server include all the necessary modifications to describe both vector access paths supported by the servers.

6.2.2. Ecere

Ecere implemented a WFS 3.0 service interface in its GNOSIS Map Server for this pilot. The service implementation focused on the direct access path (tiles), but also implemented some aspects of the feature access path (items). In addition, the service can also provide vector features and tiles using a WFS 1.1 interface (implementing GetCapabilities, DescribeFeatureType and GetFeature requests, currently with the understanding that both BBOX and CLIPBOX will result in geometry getting clipped to the requested extent.

Server deployment

The Ecere WFS/Harmonized Map Service has been deployed at the following endpoint:

- http://maps.ecere.com/hms (JSON [http://maps.ecere.com/hms?f=json])

Tiles for any layers are generated on-the-fly for any supported encoding and tiling scheme from a GNOSIS tiled data store.

It supports the following vector tile formats:

- text/xml;subtype=gml/3.1.1 - GML
- application/vnd.geo+json - GeoJSON [https://tools.ietf.org/html/rfc7946]
- application/vnd.mapbox-vector-tile - Mapbox vector tiles [https://www.mapbox.com/vector-tiles/specification/]
- application/vnd.geo+econ - GeoECON [http://docs.opengeospatial.org/per/17-041.html#GeoECON]
- application/vnd.gnosis-map-tile - GNOSIS Map Tiles [http://ecere.com/gmt.pdf] (initial publication [http://docs.opengeospatial.org/per/17-041.html#_gnosis_compact_vector_tiles_representation])

For raster and coverage layers, both PNG and GNOSIS Map Tiles are supported.

As of the time of writing, the service is still missing an OpenAPI definition (/api) and conformance declaration (/conformance). One reason for not having been able to implement this definition during the short time frame of the pilot is the evolving nature of the vector tiles extensions during the pilot, as well as a great variability between how the different implementations of WFS 3.0 describe its API, and that of the vector tiles extensions. Ecere's point of view is that if WFS 3.0 is defined using OpenAPI, the same core API definition (or a subset of it) should constitute the core of the API definition of all services implementing it. Parameterizations and datasets should be separately available, or at least confined to a specific section of the API definition where parameters are defined. A related proposal [https://github.com/OAI/OpenAPI-Specification/issues/1693] for enhancing OpenAPI to better describe a dynamic API such as OGC Web Services was submitted.

The global list of collections for the server can be found here:

- http://maps.ecere.com/hms/collections (JSON [http://maps.ecere.com/hms/collections?f=json])

The subset of collections specifically added for the Vector Tiles Pilot can be found at:

- http://maps.ecere.com/hms/collections/vtp (JSON [http://maps.ecere.com/hms/collections/vtp?f=json])

The service supports the following tiling schemes [http://maps.ecere.com/hms/tiles] (JSON [http://maps.ecere.com/hms/tiles?f=json]):

- GlobalCRS84Pixel [http://maps.ecere.com/hms/tiles/GlobalCRS84Pixel] - EPSG:4326 (geographic); Based on the WMTS Well Known Scale Set with convenient pixels/degrees values (not a quad tree, resolution does not double at each level)
- GlobalCRS84Scale [http://maps.ecere.com/hms/tiles/GlobalCRS84Scale] - EPSG:4326 (geographic); Based on the WMTS Well Known Scale Set with convenient scale denominator values (not a quad tree, resolution does not double at each level)

- GoogleCRS84Quad [http://maps.ecere.com/hms/tiles/GoogleCRS84Quad] - EPSG:4326 (geographic); Based on the WMTS Well Known Scale Set, a quad tree with convenient scale denominator values
- GoogleMapsCompatible [http://maps.ecere.com/hms/tiles/GoogleMapsCompatible] - EPSG:3857 (spherical pseudo-Mercator projection); Based on the WMTS Well Known Scale Set compatible with GoogleMaps and others (also a quad tree)
- GNOSISGlobalGrid [http://maps.ecere.com/hms/tiles/GNOSISGlobalGrid] - EPSG:4326 (geographic); A global grid with fewer columns of tiles closer to the pole than at the equator, approximating equal area tiles, a quad-tree except for starting with eight 90deg x 90deg tiles at level 0, and tiles touching a pole splitting in 3 rather than 4 [http://docs.opengeospatial.org/per/17-041.html#_global_gnosis_tiling_scheme_adapted_to_polar_regions]

For the definition of the GNOSISGlobalGrid tiling scheme, a 'variableWidths' parameter has been added.

Support for encoding vector features according to the MVT specification, using Google Protocol Buffers, was implemented during the course of this pilot within the Ecere's GNOSIS software libraries and had to be integrated within Ecere's GNOSIS Map Server.

For the purpose of facilitating a large number of collections/layers being served, the Ecere service currently uses multi-segments paths (containing slashes) for {collectionId}, even if this may not technically be valid according to the current OpenAPI specifications. This has been discussed in an issue [https://github.com/opengeospatial/WFS_FES/issues/76], and a related OpenAPI issue [https://github.com/OAI/OpenAPI-Specification/issues/1459] has been commented on. For example, for the collection served at http://maps.ecere.com/hms/collections/vtp/Daraa2/AgricultureSrf, the WFS 3.0 end-point is http://maps.ecere.com/hms and the {collectionId} is 'vtp/Daraa2/AgricultureSrf'. In practice, it is unlikely that this causes issues for a client either accessing tiles directly or parsing the collections resources.

Testing

Mapbox vector tiles generated by Ecere's WFS were tested using 'vtvalidate', which performs the following tests [2: https://github.com/mapbox/vtvalidate]:

- Tile data consistent with the MVT specification - Version 2.
- Read tile layer(s) and feature(s)
- Decode properties
- Decode geometries

Tiles were also tested successfully with OGR, which required a fix [https://github.com/OSGeo/gdal/commit/04952b2a05036e4b2d32d378b67e60c35d13912a] for an extent greater than 16384, as well as with QGIS 3.2.3. Support for visualizing Mapbox vector tiles was implemented in Ecere's visualization client (see section below), which was also used to test the tiles delivered by the WFS service.

The Ecere WFS content was also successfully rendered by Mapbox and GIS-FCU, thereby confirming a successful Technology Integration Experiment (TIE).

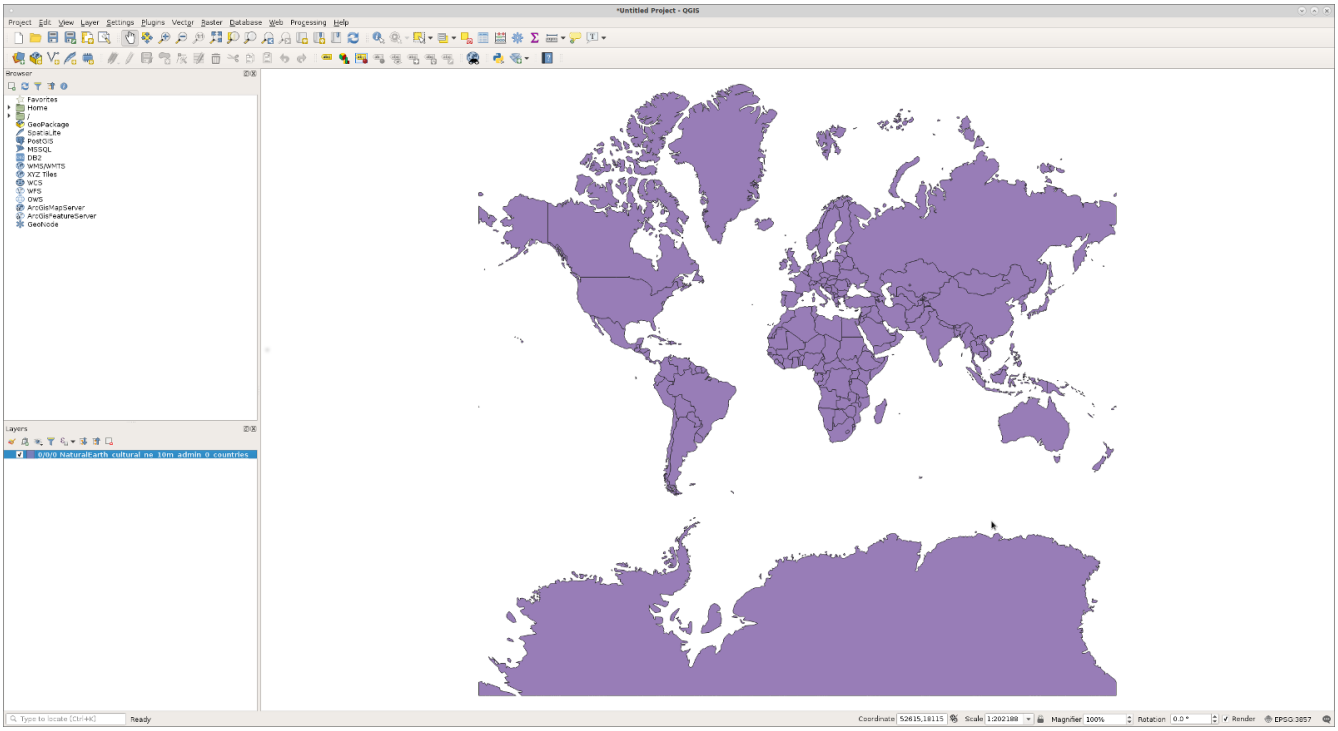

Figure 4. Mapbox vector tile served by Ecere WFS rendered in QGIS 3.2.3

Harmonized Map Service

In addition to serving vector tiles, it is also possible to request imagery and coverage tiles from Ecere's WFS 3 API.

'/layers' can be used as an alternative to '/collections' .

The following showcases how identically shaped requests can serve both vector and imagery data, for the same tile and from the same end-point:

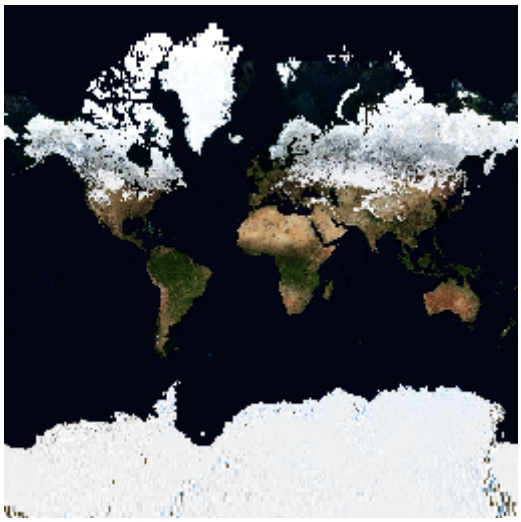

Figure 5. Imagery tile served by Ecere Harmonized Map Service

http://maps.ecere.com/hms/layers/BMNG%202004/tiles/GoogleMapsCompatible/0/0/0.png

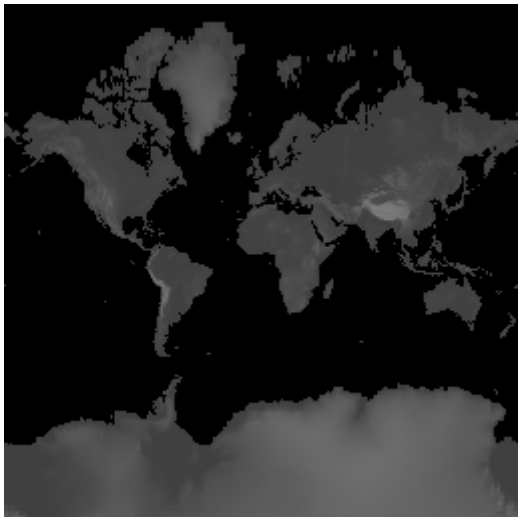

Figure 6. Elevation tile served by Ecere Harmonized Map Service

http://maps.ecere.com/hms/layers/SRTM_ViewFinderPanorama/tiles/GoogleMapsCompatible/0/0/0.png

Figure 7. Vector tile served by Ecere Harmonized Map Service

http://maps.ecere.com/hms/layers/NaturalEarth/cultural/ne_10m_admin_0_countries/tiles/GoogleMapsCompatible/0/0/0.json

Similarly, the 'items' path has been implemented as well for raster features:

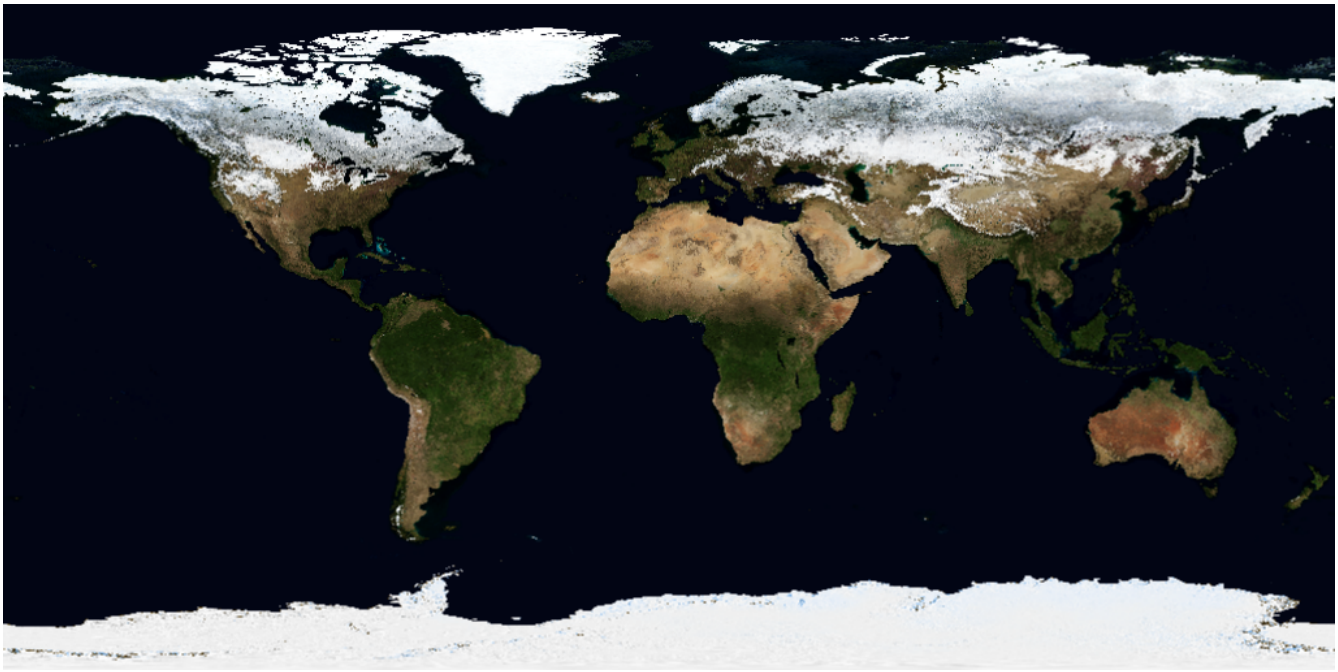

Figure 8. Imagery layer served by Ecere Harmonized Map Service

http://maps.ecere.com/hms/collections/BMNG%202004/items.png

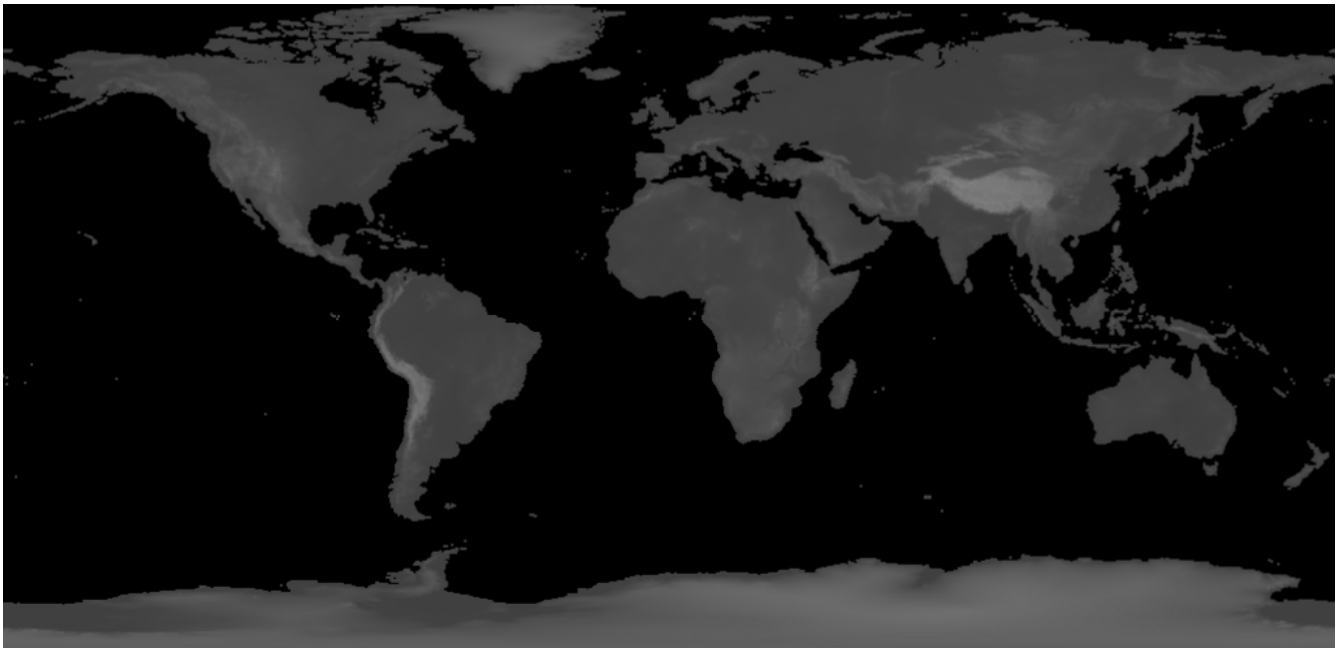

Figure 9. Elevation layer served by Ecere Harmonized Map Service

http://maps.ecere.com/hms/collections/SRTM_ViewFinderPanorama/items.png

Both raster and vector layers from Natural Earth are served from http://maps.ecere.com/hms/layers/NaturalEarth .

Vector tiles from OrdnanceSurvey's OpenMapLocal, used in OGC Testbed-13, are available from http://maps.ecere.com/hms/collections/OpenMapLocal (now as MVT as well).

Although a lot of commonality applies to serving geospatial data of different types, it should also be possible to provide additional parameters pertaining only to a particular type.

6.2.3. GeoSolutions

GeoSolutions implemented the WFS 3.0 tiling extensions as part of the WFS3 community module of GeoServer. The module is available to everybody as part of the nightly builds [https://build.geoserver.org/geoserver/master/community-latest/geoserver-2.15-SNAPSHOT-wfs3-plugin.zip], and the source code is available in the official repository [https://github.com/geoserver/geoserver/tree/master/src/community/wfs3].

The server component implementation provides three separate datasets at the following addresses:

- WFS3 Daraa Dataset http://maps.geo-solutions.it/geoserver/vtp/wfs3/collections
- WFS3 Iraq Dataset http://maps.geo-solutions.it/geoserver/iraq_vtp/wfs3/collections
- WFS3 Syria Dataset http://maps.geo-solutions.it/geoserver/syria_vtp/wfs3/collections

The "direct tile access path API" is implemented with MVT, the results contain data generalized based on the current zoom level and clipped with a small buffer around the tile boundary.

In the current implementation of the direct access path:

- Outputs are always generated on the fly, there is no caching (as it can be seen by requesting low zoom level tiles on the larger datasets, e.g., Iraq and Syria)
- All attributes available for the dataset are included in the tile
- All features with a geometry that has not been simplified or collapsed down to a lower dimensional representation are included (e.g., polygons and lines collapsing down to a point are skipped)

The current implementation supports two tiling schemes:

- `GlobalCRS84Geometric`, a global WGS84 based tiling scheme having two tiles at zoom level zero
- `GoogleMapsCompatible`, a global EPSG:3857 based tiling scheme, as described in the WMTS as a well-known scale set

The server also implements the "feature access path API" (also known as the "convenience path") by allowing encoding of Mapbox vector tiles out of the `/collections/{collectionId}/items` resource using the `mvt` format.

All the common `items` resource query parameters apply, in addition to that:

- The `bbox` parameter, if present, is also used to cut the generated tile (if non-square, the returned tile will be a square area containing the requested BBOX, with the lower part empty)
- If the `bbox` parameter is missing, the entire collection bounds are used
- The `resolution` parameter controls the size of the generated MVT in screen space pixels (by default it uses 256 pixels).

The OpenAPI definitions include the tiling extensions, and the collection resources have links pointing to the tiles and tiling schemes resources as part of the "links" array, e.g. (check the "tilingScheme" and "tiles" relations): https://backoffice-maps.geo-solutions.it/geoserver/wfs3/collections/syria_vtp__building_s?f=json

For the tiling scheme / tile matrix set, the implementation currently uses the JSON encoding in the latest version of the OGC Tile Matrix Set draft standard, along with minor deviations requested during the pilot to ease client developers work: Example: https://backoffice-maps.geo-solutions.it/geoserver/wfs3/collections/syria_vtp__building_s/tiles/GoogleMapsCompatible

For the schema extensions for the collection metadata, a list of tiling schemes and the URI template for the tiles as a link are included. Example: https://services.interactive-instruments.de/vtp/daraa/collections/vegetationsrf?f=json

6.2.4. interactive instruments GmbH

Server and clients

The WFS 3.0 landing page for the server is: https://services.interactive-instruments.de/vtp.

There are three datasets:

- Daraa: https://services.interactive-instruments.de/vtp/daraa
- Iraq: https://services.interactive-instruments.de/vtp/iraq
- Syria: https://services.interactive-instruments.de/vtp/syria

All datasets have been configured to provided more readable results in the HTML on the `/collections/{featuretype}/items` path. The GeoJSON and Mapbox Vector Tile outputs continue to use the coded values (one reason is that the values are typically used in style definitions).

For testing, two very simple OpenLayers clients have been created:

- http://portele.de/daraa.html for the Daraa dataset
- http://portele.de/iraq-syria.html for the other two, larger datasets

Both clients use the `/tiles` path (that is, all layers are included in a single Protobuf file) and feature a simple info box displaying feature properties, if the mouse hovers over a feature. The client uses just a single, simple style for all layers.

Implementation and status

The "direct tile access path API" is implemented with Mapbox Vector Tile and GeoJSON output. The Mapbox Vector Tile data is clipped with a small buffer around the tile boundary. No clipping occurs in the GeoJSON.

The Mapbox Vector Tile and GeoJSON output on the `tiles` and `items` paths support a query parameter `properties`. If provided, only the requested properties will be included in the response. Example: `properties=f_code,zi005_fna`.

Properties with values that are null, "No Information" and -999999 are always excluded.

On the `/tiles/{tilingSchemeId}/{level}/{row}/{col}` path a query parameter `collections` is supported for Mapbox Vector Tile output. If provided, only the listed layers/feature types will be included in the response. Example:

`collections=transportationgroundcrv,transportationgroundpnt,transportationgroundsrf`.

Currently only the standard tiling scheme is supported (called "GoogleMapsCompatible" in the WMTS standard), using `default` as the id.

The OpenAPI definitions include the tiling extensions.

For the tiling scheme / tile matrix set, the implementation currently uses the JSON encoding in the latest version of the OGC Tile Matrix Set draft standard. Example: https://services.interactive-instruments.de/vtp/daraa/collections/vegetationsrf/tiles/default?f=json.

For the schema extensions for the collection metadata, a list of tiling schemes and the URI template for the tiles as a link are included. Example: https://services.interactive-instruments.de/vtp/daraa/collections/vegetationsrf?f=json

Source Code

The updated ldproxy code is available in a GitHub repository [https://github.com/interactive-instruments/ldproxy]. During the Pilot, the code was provided through the "vt" branch of the repository. Changes to the branch updates an ldproxy Docker image [https://hub.docker.com/r/iide/ldproxy/] with the tag "vt". The demonstration server is automatically updated, too.

6.2.5. Mapbox

The Mapbox implementation of a WFS 3.0 server has been deployed here:

http://ec2-54-88-75-105.compute-1.amazonaws.com/

The Mapbox WFS 3.0 server runs on Amazon Web Services (AWS) and uses a Relational Database Service (RDS) provided by Amazon. The server is using a node.js (https://nodejs.org/en/) server with express.js (https://expressjs.com/) on top. Express is used to query a PostGIS database, which then serves each available feature along with metadata about that feature.

Tiles are served by using an open source library called tilelive (https://github.com/mapbox/tilelive) and tilelive-postgis (https://github.com/stepankuzmin/tilelive-postgis). These two packages create a tile and send that to the client. A user can generate a specific tile and have it downloaded to their machine by putting in the appropriate url with the {z}/{x}/{y}. Once sent using express.js (https://expressjs.com/), vector tiles are consumed by a client and rendered to the map.

6.3. WFS Clients

6.3.1. Ecere

The Ecere WFS 3.0 client is implemented as a capability of Ecere's GNOSIS software libraries, available from within Ecere's GNOSIS Cartographer GIS tool, as well as from any applications built using the GNOSIS SDK (whether for desktop, web or mobile). Support for accessing tiles through the WFS 3.0 REST API has been added for this pilot, with support for GNOSIS Map Tiles, GML, GeoECON, GeoJSON and Mapbox vector tiles. Visualization of Mapbox vector tiles and GeoJSON was an entirely new capability developed during the pilot.

Figure 10. Vector tiles served by Ecere WFS rendered in Ecere's GNOSIS client

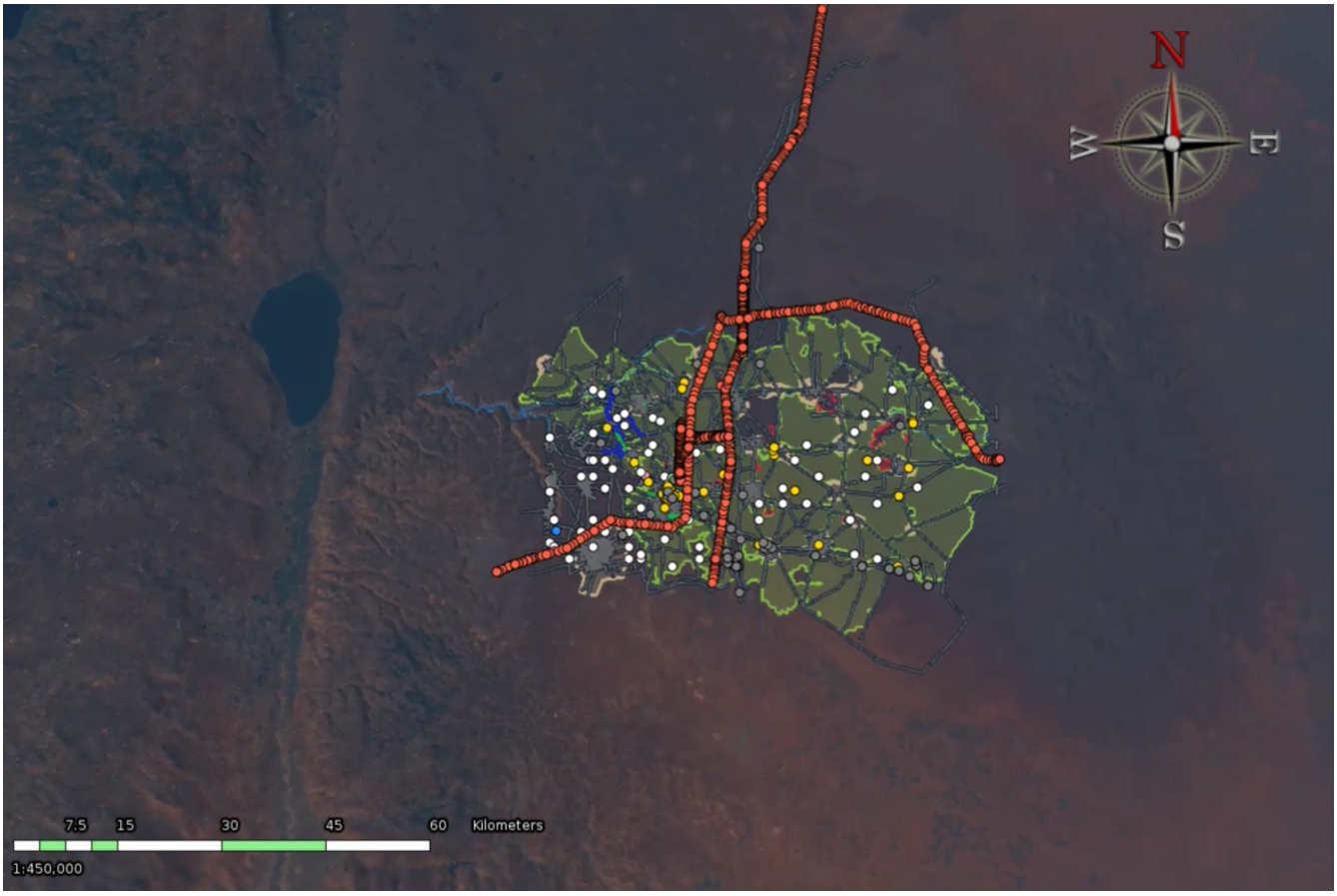

Figure 11. Vector tiles served by CubeWerx WFS rendered in Ecere's GNOSIS client

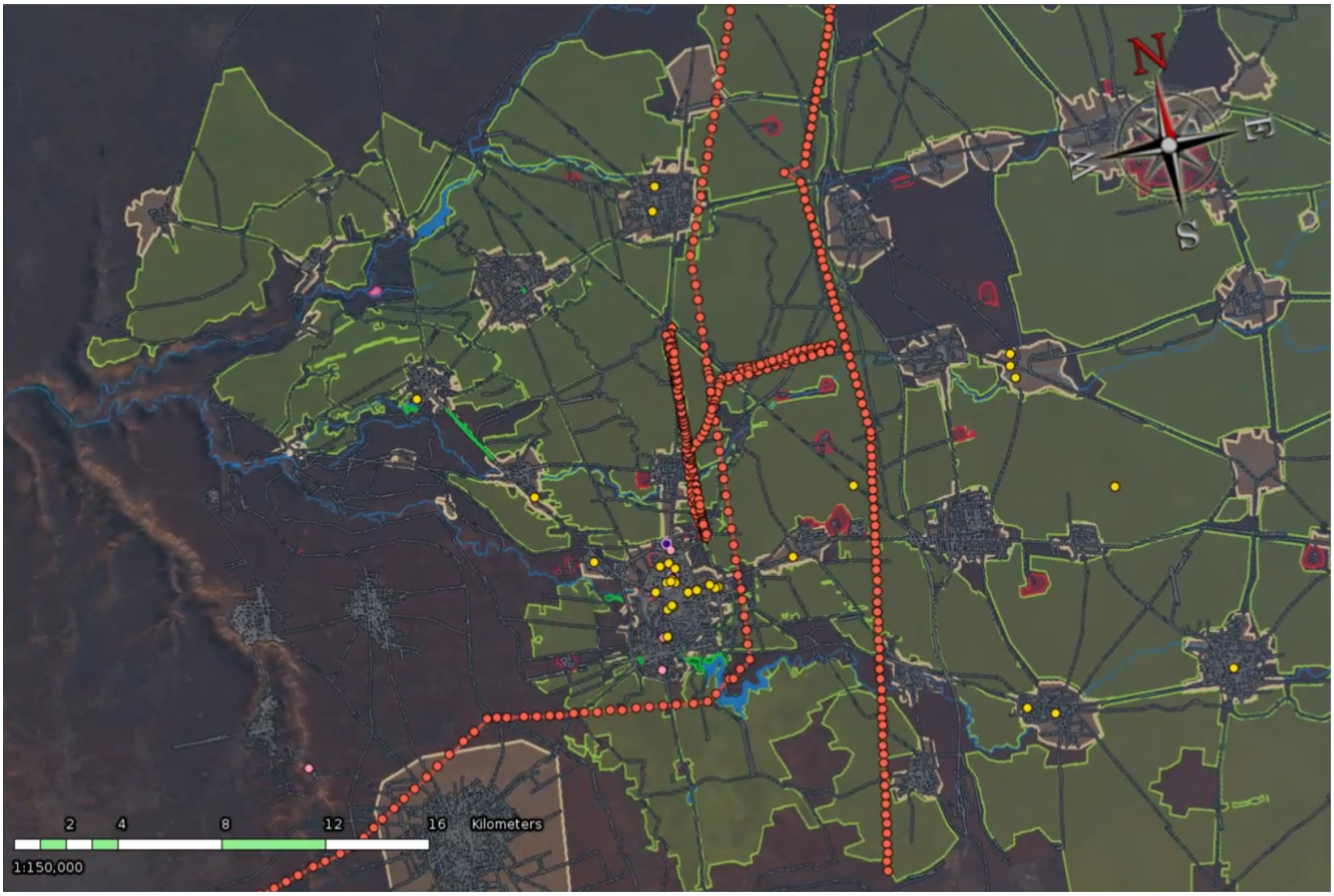

Figure 12. Vector tiles served by GeoSolutions WFS rendered in Ecere's GNOSIS client (surfaces only)

Figure 13. Vector tiles served by Mapbox WFS rendered in Ecere's GNOSIS client (agricultural surfaces)

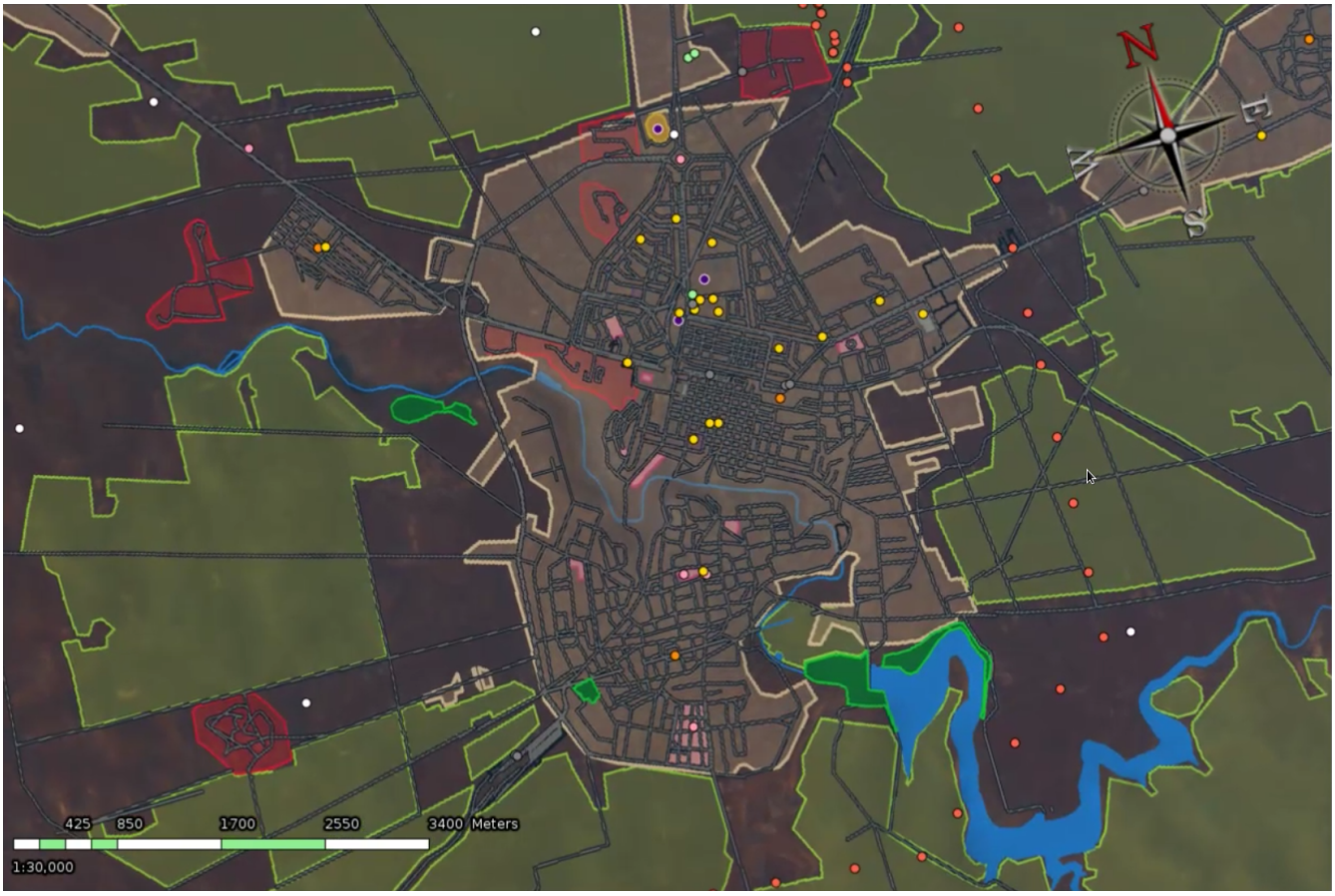

Figure 14. Vector tiles served by Interactive Instruments WFS rendered in Ecere's GNOSIS client

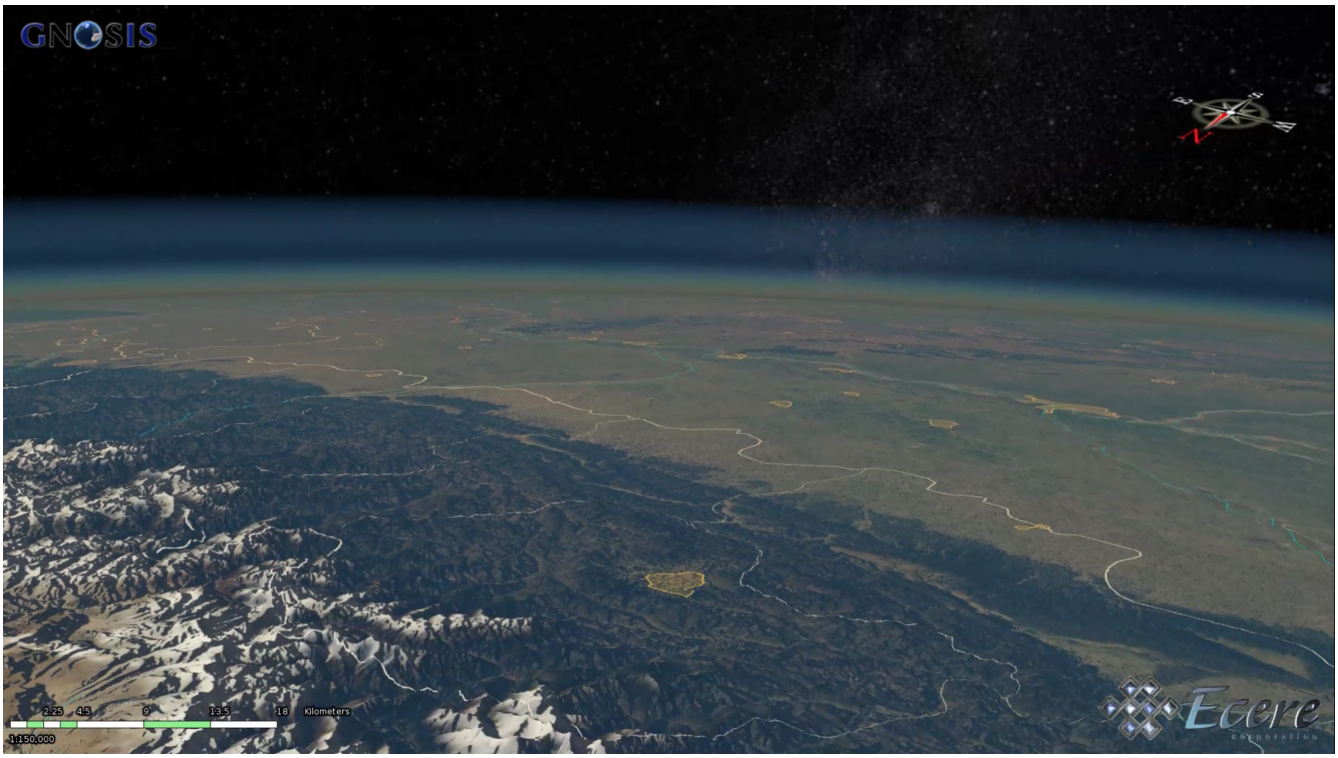

Figure 15. Vector, imagery and terrain elevation tiles served from a single end-point by Ecere's Harmonized (WFS 3.0 style) Map Service rendered in Ecere's GNOSIS client

Mapbox Vector Tiles inner v. outer contours:

- For Mapbox vector tiles not adhering to the 2.1 specifications (with specific ordering rules: outer contours appear clockwise, inner contours appear counter-clockwise), it is tricky to

determine whether a given contour of a polygon is inner or outer. The MVT polygon encoding has no construct either separating the different polygons, or marking a given contour as inner or outer. There is only a 'ClosePath' command, which identifies the end of a contour. One cannot use 'first contour is outer' rule, because it is not known where one polygon ends and the next one begins. The decoder must check whether a contour has an area (absolute value) smaller than the previous contour, as the outer contour would have an area greater than the sum of all inner ones.

Recommendations:

- The possibility to select one of two possible approaches to avoiding unwanted edges of polygons when drawing strokes, as well as allowing feature reconstruction should be offered: an extra border as is used in Mapbox vector tiles, or marking artificial segments (e.g. http://ogc.standardstracker.org/show_request.cgi?id=515). It should be possible for clients and services to implement either or both approaches (the Ecere WFS and clients support both, but no parameters have been defined to handle negotiation, and no standard mechanism for marking edges within GML and GeoJSON have been agreed upon).
- Vector tiles should define an ID for features (including making use of the currently optional and ambiguous 'id' of the Mapbox vector tile specifications) which should match the feature ID before the vector data was tiled. This facilitates recombining the geometry, without having to rely on the attributes (e.g. MVT tags).
- The upcoming candidate for describing tiling schemes (tile matrix sets) across OGC standards should offer additional flexibility, such as the ability to describe tile matrices of different widths at different latitudes which would support both the GNOSIS Global Grid [http://docs.opengeospatial.org/per/17-041.html#_global_gnosis_tiling_scheme_adapted_to_polar_regions], as well as the CDB [https://portal.opengeospatial.org/files/?artifact_id=72714] grid, as previously raised in Testbed 13 (CR 518 [http://ogc.standardstracker.org/show_request.cgi?id=518]).

6.3.2. GIS FCU

The GIS.FCU implementation of WFS 3.0 client has been deployed here:

https://map.gis.tw/pilot/wfs.html

This client shows the maps that compare Mapbox Vector Tile format with GeoJSON Vector Tile format from different WFS3.0 server sources at the same time, as illustrated in Figure 16. Each map 'lazy' loads all of the layers that have been retrieved from the /collections endpoint.

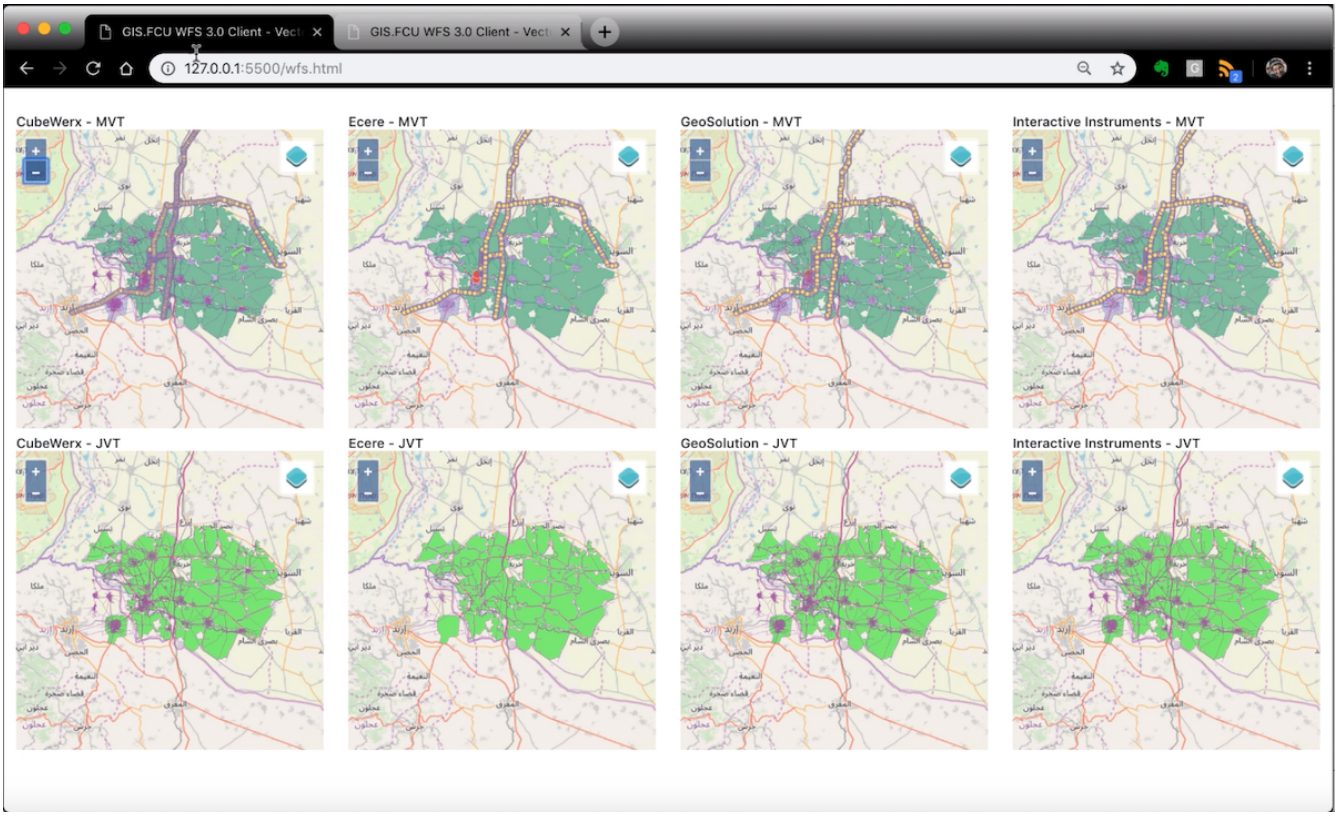

Figure 16. GIS FCU WFS3 vector tiles client

The style of each map emphasized on the layers of agriculture surfaces and facility points, and the colors used are based on the Digital Printing Specification of the National Geospatial Intelligence Agency (NGA). The client both uses Direct Tile Access Path and Feature Access Path based on the URL templates to access the MVT or GeoJSON depends on what format servers provided.

The URL template example of Direct Tile Access Path for MVT is below:

```
https://services.interactive-instruments.de/vtp/daraa/tiles/{tilingSchemeId}/{z}/{y}/{x}?f=mvt
```

The URL template example of Direct Tile Access Path for tiled GeoJSON is below:

```
http://maps.ecere.com/hms/layers/vtp/Daraa/{collection_id}/tiles/{tilingSchemeId}/{z}/{y}/
{x}.json
```

The client is a web application that uses the openlayers v5.1.3 JS library and provides a layer switcher to enable or disable a layer.

6.3.3. Mapbox

The Mapbox implementation of WFS 3.0 client has been deployed here:

http://bl.ocks.org/briandaviddavidson/c81b06997c6bb086ed2e6152ef998fab

The Mapbox WFS 3.0 client is a simple HTML file. It uses mapbox-gl.js (https://github.com/mapbox/mapbox-gl-js) to render tiles sent from the server. Tiles are added to the map once the map is loaded through the `map.addLayer()` method. Styling is also completed inside of `addLayer` by using the `paint` property in mapbox-gl.js.

A tooltip was added to features using the `map.on()` method and passing in `mousemove` and `mouseleave`

for tooltip creation and destruction respectively. The tooltip is added only when users are hovering on an object in the associated layer added to the map.

6.3.4. GeoSolutions

While not directly tasked to build clients, GeoSolutions nevertheless built a couple of clients based on OpenLayers.

The first client shows the same Daraa map as provided by different servers:

http://vtp2018.s3-eu-west-1.amazonaws.com/wfs.html

The top row shows a comparison between tiles provided by GeoServer in the two tiling schemes, while the lower part pulls data off the various other server implementations. All maps are synchronized, panning/zooming on a map updates the other maps as well. The controls at the top right allow a user to change the colors of the vectors, to demonstrate basic client side styling. Clicking on the map opens a popup on each map showing the attributes included in the vector tiles.

Figure 17. Vector Tiles Rendered on the GeoSolutions Client showing features requested client side, directly from vector tiles

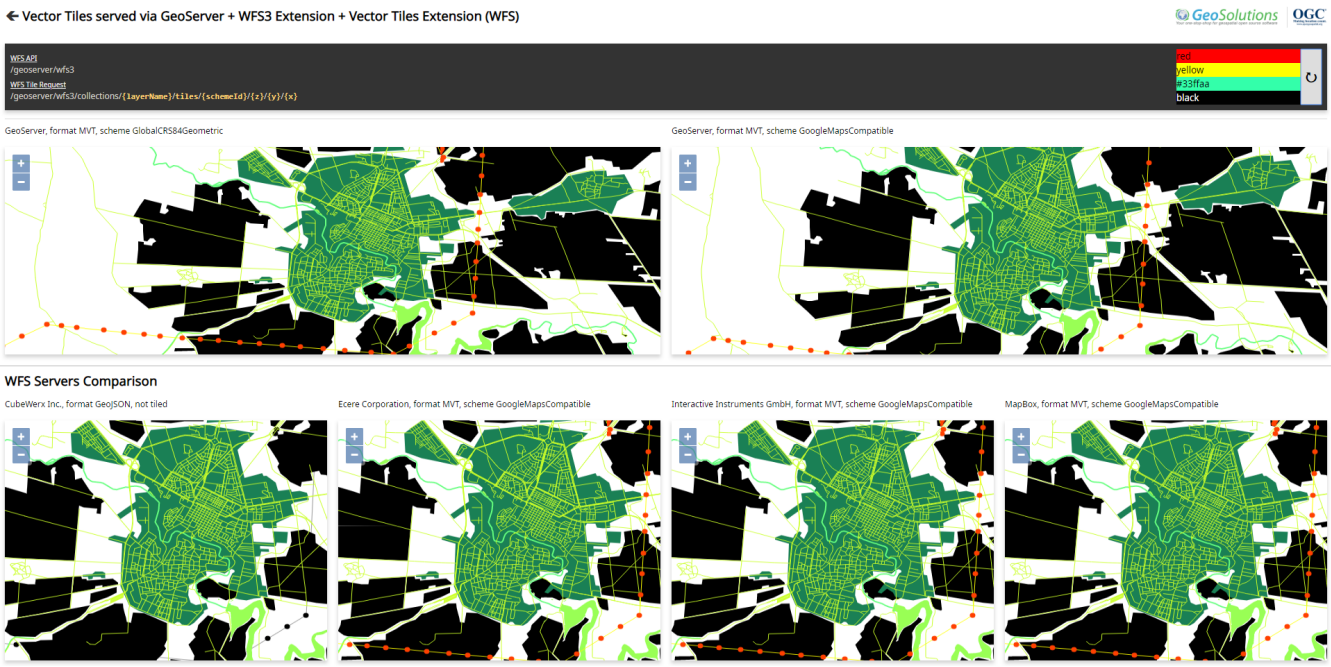

Figure 18. Vector Tiles Rendered on the GeoSolutions Client showing styles applied client side dynamically with a simple style editor

6.4. Technology Integration Experiments (TIE)

Table 3. WFS TIE Matrix

CLIENTS	SERVERS				
	CubeWerx	Ecere	GeoSolutions	interactive instruments	Mapbox
Ecere	Connects	Connects	Connects	Connects	Connects
GIS FCU	Connects	Connects	Connects	Connects	
Mapbox	Connects	Connects	Connects	Connects	Connects

Appendix A: Abstract Test Suite

A.1 Conformance class: Core

Table 4. A.1 Direct Tile Access Path API

Test identifier	http://www.opengis.net/spec/wfs/3.0/vt/conf/core/direct-tile-access-path-api
Test purpose:	To verify that an implementation of the Direct Tile Access Path API of the Vector Tiles extension of WFS 3.0 correctly behaves according to the description in Clause 5.2.
Test method:	Inspect the responses for paths and parameters described in Clause 5.2
Requirement:	http://www.opengis.net/spec/wfs/3.0/vt/req/core/direct-tile-access-path-api
Test type:	Basic

Table 5. A.2Feature Access Path API

Test identifier	http://www.opengis.net/spec/wfs/3.0/vt/conf/core/feature-access-path-api
Test purpose:	To verify that an implementation of the Feature Access Path API of the Vector Tiles extension of WFS 3.0 correctly behaves according to the description in Clause 5.3.
Test method:	Inspect the responses for paths and parameters described in Clause 5.3
Requirement:	http://www.opengis.net/spec/wfs/3.0/vt/req/core/feature-access-path-api
Test type:	Basic

Appendix B: Revision History

Table 6. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
2018-08-28	P. Vretanos	.1	all	initial version
2018-09-28	B. Davidson	.2	all	Mapbox content
2018-10-08	C. Portele	.3	all	ii content
2018-10-16	C. Lin	.4	all	GIS.FCU content
2018-10-19	A. Aime	.5	all	GeoSolutions content
2018-11-01	J. St-Louis	.6	all	Ecere content
2018-11-03	P. Vretanos	.7	all	final version

Appendix C: Bibliography

1. Shang, X.: A Study on Efficient Vector Mapping with Vector Tiles Based on Cloud Server Architecture. University of Calgary, https://prism.ucalgary.ca/handle/11023/2666 (2015).
2. Cavazzi, S.: OGC Testbed-13: Vector Tiles Engineering Report. OGC 17-041,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-041.html (2018).