OGC Vector Tiles Pilot 2

Tile Set Metadata Engineering Report

Publication Date: 2020-07-08

Approval Date: 2020-06-26

Submission Date: 2020-05-01

Reference number of this document: OGC 19-082r1

Reference URL for this document: http://www.opengis.net/doc/PER/vtp2-D001

Category: OGC Public Engineering Report

Editor: Sergio Taleisnik

Title: OGC Vector Tiles Pilot 2: Tile Set Metadata Engineering Report

---

OGC Public 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 Public 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.

Table of Contents

Chapter 1. Subject

The OGC Vector Tiles Pilot 2: Tile Set Metadata Engineering Report (ER) describes a conceptual model for Tile Set Metadata that provides information about the intended usage of a Tile Set as well as the origin, security level, tiling scheme, layers and feature properties contained within. In this ER, a tile set is a series of tiles containing data and following a common tiling scheme.

The metadata is intended to facilitate retrieval of tile sets and describes the major characteristics of tile sets without actually accessing the tiles nor the content contained in a tile. Such a process could be time consuming when there are a large number of tiles in a tile set.

Additionally, this ER summarizes the discussions about Tile Set Metadata among the VTP2 participants, and draws up conclusions and recommendations for future work on the subject.

Finally, this ER describes the Technology Integration Experiments (TIEs) performed to test the prototype implementation of the proposed Tile Set Metadata Model on API endpoints, client applications, and GeoPackages.

Chapter 2. Executive Summary

Tile sets are a well-established method for structuring, storing, and accessing manageable packages of tiled data, both offline and online. As the use of tile sets increases and the need for seamless transitions between online and offline environments increases, the need for efficient access to the tile set metadata becomes more critical.

The objective of VTP2 was to deliver a consistent, interoperable online/offline architecture for vector tiles based on feature servers, tile servers and GeoPackage.

The metadata activities of VTP2 can be grouped into three major components:

- Designing the Tile Set Metadata Model.
- Capturing and summarizing discussions and recommendations.
- Describing TIEs implementing the Tile Set Metadata Model.

The Tile Set Metadata Model built in VTP2 is based on requirements established in the Call For Participation (CFP) and then further refined through the course of the Pilot. The initial metadata requirements specified in the CFP were to:

- Develop a metadata model for vector tiles and stored tile caches.
- Extend the National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF).
- Design the metadata to describe fundamental aspects of the tiles, such as date, creator, source, etc.
- Specify metadata elements that describe the tiling scheme (tile matrix set).
- Define how space is partitioned into individual tiles, substitution variables to identify tiles using a three-part identifier such as {level} (zoom), {row} (vertical) and {col} (column: horizontal).
- Provide elements for Style names or identifiers and metadata to describe multi-layer vector tiles.
- Provide other elements required to support the Vector Tiles Filter language [1].
- Provide a standard resource path structure to consistently present simple metadata for vector tiles as an extension to tile and feature servers, standalone tile caches and GeoPackage tables.

NOTE	The term *tile cache* was used in the VTP2 CFP as a synonym for tile set repositories. The use of this term was subject to discussion throughout the Pilot and reflected by this ER.

Since the CFP was defined a year before its release, many requirements evolved in order to cope with the developments carried out since then. Additionally, conversations throughout the Pilot further refined these requirements: The requirement to extend the NMF was changed to limiting the NMF elements implemented to only those considered of value to the tile set use cases presented in this Pilot, and the requirement for describing tile and tile set metadata was changed to exclusively describing tile set metadata.

The metadata model defined in this Pilot can be divided into three main groups:

- Describing fundamental aspects of the TileSet (representation of a tile set) such as names, identifiers, dates, owners, origin, security levels, etc.
- Describing the tiling scheme by storing the Tile Matrix Set information of the TileSet. This helps identify the geographic extent, the level of detail and the tiling scheme.
- Describing the layers and feature properties stored within, in order to understand the use and application domain of the TileSet.

A novel approach to store and access Tile Set Metadata was implemented. The creation of an additional resource to an OGC Application Programming Interface (API) was discarded for many technical reasons. This led to consensus that all the Tile Set Metadata should be stored in a stand-alone file that would be located in the root folder of the offline repository.

2.1. Business Value

The main benefit of transitioning from raster tiles to vector tiles has been the possibility of flexible map styling and the reduction of storage space required for maps, the latter allowing for maps being stored on devices with lower storage capacity as well as requiring lower bandwidth communications for transmission. In some cases, depending on the file format, maps structured using tiled vector/feature data can be 20 to 30 times smaller than the same maps represented by raster tiles for a similar level of detail. This reduction enables the possibility of storing a large number of maps (i.e.,tile sets) into secure and lightweight removable media devices.

The VTP2 initiative implemented a scenario (Figure 1) in which a tile set repository (such as GeoPackage) is being used by a humanitarian relief convoy in the middle of the desert. The convoy has limited to no connectivity and is only supported by a group of interconnected systems working and communicating with each other. The repositories are generated in Command Post Computing Environments (CPCE) and comprise tile sets, styles, maps and routes served by National-level Services and Enterprise-level Services, which communicate with each other through implementations of OGC APIs for styles, tiles, images and routes.

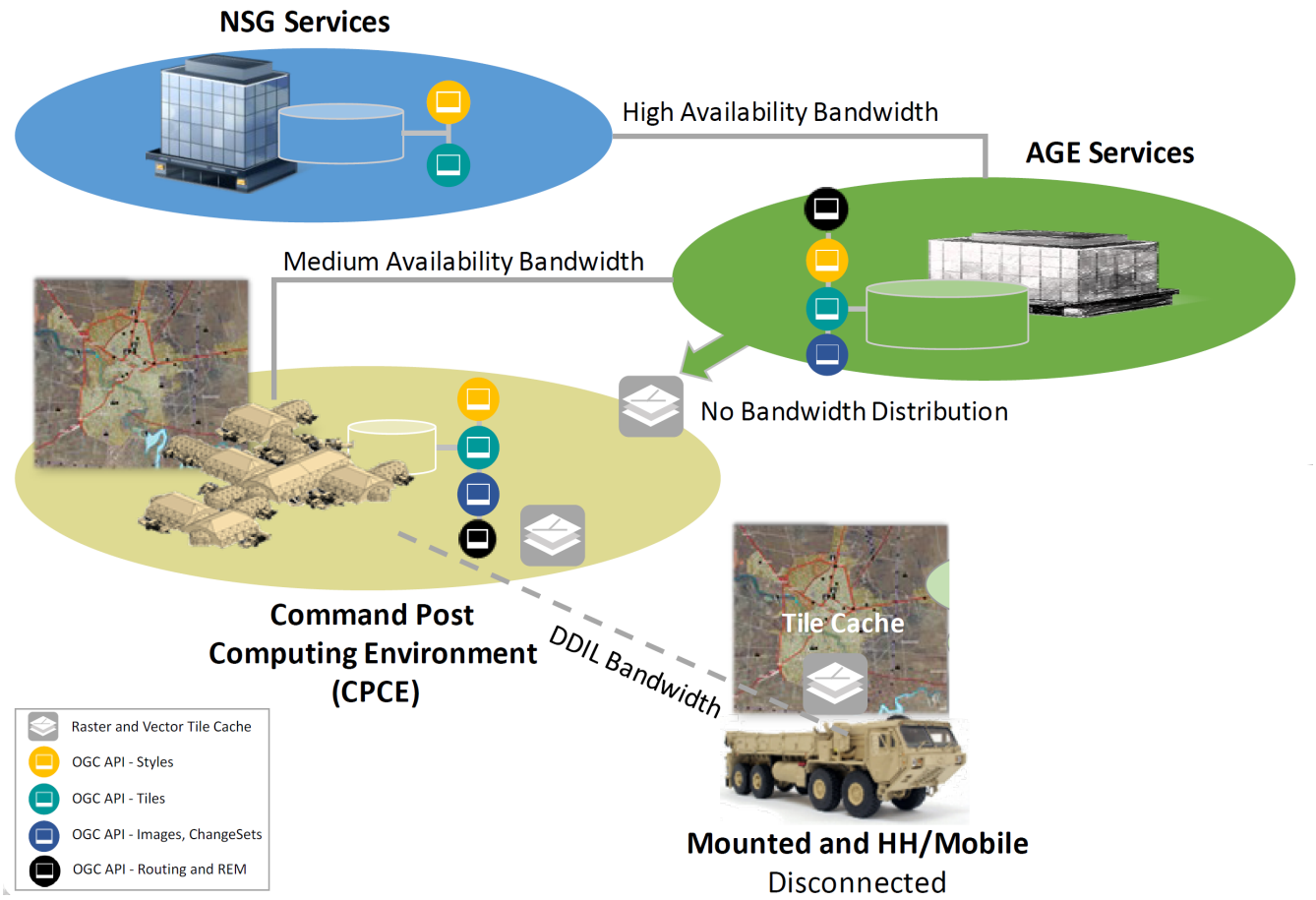

Figure 1. Sponsor Modular Scenario

The intention of working with tiled feature sets is to transition from using classic raster maps into using up-to-date and small-area tiles which enable different styles to be applied as needed. Using this approach, an end user can easily transition from a Day Topographic styling during daytime operations to a Night Topographic styling during night time, or even a hybrid satellite imagery/vector overlay styling. Tile sets containing linear road features also enable routing engines to calculate routes and display them on top of a map. However, this scenario is outside of the scope of this Pilot.

Tile Set Metadata is used to describe the content of these tile sets. The main scenario described by the Pilot sponsor required accessing these vector tile sets in a completely offline environment and utilizing tile set metadata to quickly review the main elements and information about those tile sets without having to access deep into each tile of the tile set. The requirement includes using an open standard for Tile Set Metadata in order to allow for coalition partners from different origins to access these tile sets.

Tile Set Metadata is also critical when fetching tile sets from other sources. As shown in Figure 1, tiles can be served through OGC APIs from National-level Services to Enterprise-level Services, and further to CPCEs with Denied, Degraded, Intermittent, or Limited (DDIL) network connectivity. The inclusion of metadata within those tile sets allows users among all levels to quickly access the fundamental pieces of information related to those tile sets and act accordingly by creating repositories, such as GeoPackages, or simply utilizing them on applications.

Finally, Tile Set Metadata is required for the design of software application systems. Developers from both NSG Services and AGE Services develop the software applications that are eventually used on the field by soldiers and humanitarian relief missions. A comprehensive metadata model

would provide the engineers with a well-defined, complete description of the tile, facilitating the development of new applications utilizing this technology. The use of open standards enables interoperability with other systems, expanding the potential use of tiled geospatial datasets.

2.2. Document Contributor Contact Points

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

Contacts

Name	Organization	Role
Sergio Taleisnik	Skymantics	Editor
Andrea Aime	GeoSolutions	Contributor
Stefano Bovio	GeoSolutions	Contributor
Jeff Harrison	US Army Geospatial Center	Contributor
Gobe Hobona	OGC	Contributor
Terry Idol	OGC	Contributor
Jérôme Jacovella-St-Louis	Ecere Corporation	Contributor
Clemens Portele	interactive instruments GmbH	Contributor
Jeff Yutzler	Image Matters	Contributor

2.3. Foreword

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

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

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 17-045, OGC® Testbed-13: Portrayal Engineering Report [http://docs.opengeospatial.org/per/17-045.html]
- OGC: OGC 17-083r2, OGC® Two Dimensional Tile Matrix Set [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html]
- OGC: OGC 18-101, Vector Tiles Pilot Extension Engineering Report [http://docs.opengeospatial.org/per/18-101.html]
- OGC: OGC 15-113r5, Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure [https://portal.opengeospatial.org/files/15-113r5]
- OGC: OGC 18-076, OGC® Vector Tiles Pilot: Tiled Feature Data Conceptual Model Engineering Report [http://www.opengis.net/doc/PER/vtp-conceptualModel]
- OGC: OGC 19-010, OGC® Testbed-15: Styles API Engineering Report [https://docs.opengeospatial.org/DRAFTS/19-010.html]
- OGC: OGC 19-023, OGC® Testbed-15: Encoding and Metadata Conceptual Model for Styles Engineering Report [http://docs.opengeospatial.org/per/19-023r1.pdf]
- OGC: Core Tiling Conceptual and Logical Models for 2D Euclidean Space [http://docs.opengeospatial.org/DRAFTS/19-014.html]
- OGC: OGC 20-057, OGC API - Tiles - Part 1: Core [https://github.com/opengeospatial/OGC-API-Tiles/blob/master/core/standard/OAPI_Tiles.adoc]
- NGA: National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF) [https://portal.opengeospatial.org/files/?artifact_id=83568], *Not publicly available.*
- Mapbox: TileJSON 3.0.0 Draft Specification [https://github.com/mapbox/TILEjson-spec/tree/3.0/3.0.0]

Chapter 4. Terms and Definitions

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

- **tile**

  geometric shape with known properties that may or may not be the result of a tiling (tessellation) process. A tile consists of a single connected "piece" without "holes" or "lines" (topological disc).

  NOTE For the purposes of this OGC ER, a tile is a small rectangular representation of geographic data, often part of a set of such elements, covering a tiling scheme and sharing similar information content and graphical styling. A tile can be uniquely defined in a tile matrix by one integer index in each dimension. Tiles are mainly used for fast transfer (particularly in the web) and easy display at the resolution of a rendering device. Tiles can be grid based pictorial representations, coverage subsets, or feature based representations (e.g., vector tiles).

- **tile cache**

  no formal definition established. Widely considered to be a repository containing a pre-generated tile set. Its use was suggested to be dropped during the Pilot. This ER provides discussion, findings and recommendations about the use of this term.

- **tile matrix**

  a grid tiling scheme that defines how space is partitioned into a set of conterminous tiles at a fixed scale.

  NOTE A tile matrix constitutes a tessellation of the space that resembles a matrix in a 2D space characterized by a matrix width (columns) and a matrix height (rows).

- **tile matrix set**

  a tiling scheme composed by collection of tile matrices defined at different scales covering approximately the same area and has a common coordinate reference system.

- **tiling scheme | tile set scheme**

  a scheme that defines how space is partitioned into individual tiled units. A tiling scheme defines the spatial 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.

  NOTE A tiling scheme is not restricted to a coordinate reference system or a tile matrix set and allows for other spatial reference systems such as Discrete Global Grid Systems (DGGS) and other organizations including irregular ones.

- **tile set**

  set of tiles - a collection of subsets of the space being partitioned.

NOTE	For the purposes of this OGC ER, a tile is a series of actual tiles contain data and following a common tiling scheme.

- **tile set metadata**

  In addition to the common properties that define a tile set, additional metadata may be provided. Such metadata could be an abstract, the owner, the author, or other common metadata. [OGC 19-014r2]

NOTE	In this ER, these metadata include essential information about a tile set needed to support users in their efforts to discover, select and modify tile sets.

- **vector tile**

  a tile that contains vector information that has been simplified at the tile scale resolution and clipped by the tile boundaries.

4.1. Abbreviated Terms

- AGE Army Geospatial Enterprise
- API Application Programming Interface
- CM Conceptual Model
- CRS Coordinate Reference System
- ER Engineering Report
- JSON JavaScript Object Notation
- MVT Mapbox Vector Tiles
- NMF NSG Metadata Foundation
- NSG National System for Geospatial Intelligence
- OGC Open Geospatial Consortium
- OPF Open Portrayal Framework
- TFD Tiled Feature Data
- TIE Technology Integration Experiment
- TSM Tile Set Metadata
- UML Unified Markup Language
- URI Uniform Resource Identifier
- URL Uniform Resource Locator
- VT Vector Tile
- VTP2 Vector Tiles Pilot 2

Chapter 5. Overview

- Section 6 presents background information that served as a base for the development of this metadata model.
- Section 7 describes the Tile Set Metadata Model, its elements and sources of data.
- Section 8 describes the implementations involving Tile Set Metadata carried out by each participant organization.
- Section 9 provides discussions.
- Section 10 summarizes the results of this Pilot in terms of metadata work.
- Section 11 describes the findings.
- Section 12 outlines the lessons learned.
- Section 13 provides conclusions and recommended future work.
- Annex A provides a sample JSON implementation of the Tile Set Metadata Model.
- Annex B contains the document revision history.
- Annex C contains bibliographic references.

Chapter 6. Background

6.1. Tiling Conceptual Model

The Tiling Conceptual Model, as showcased in Figure 2, describes the relationship between Tiles, Tile Sets, Tile Scheme and Tile Set Metadata. The tiling conceptual model shown in Figure 2 is a previous version of the one provided by the draft Abstract Specification for Core Tiling Conceptual and Logical Models for 2D Euclidean Space (OGC 19-014r1). The model shown in the figure was replicated in the Tile Set Metadata Model, although the `Tile` element was eventually discarded from the Tile Set Metadata Model in order to focus on the concept of a tile set.

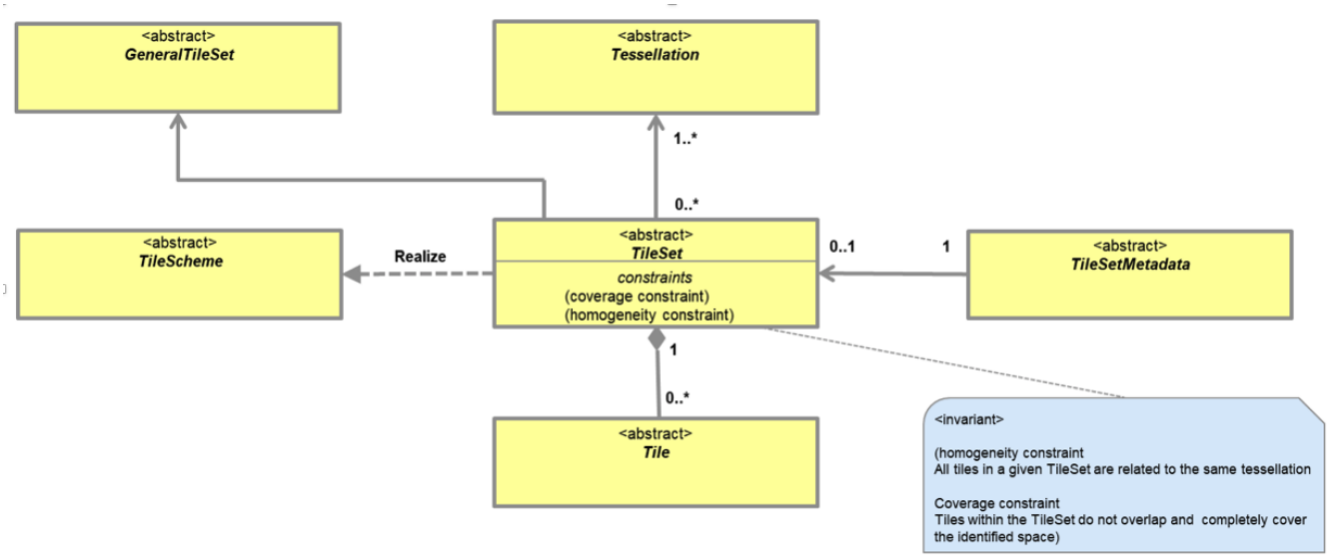

Figure 2. Tiling Conceptual Model

6.2. Tile Matrix Set Standard

The OGC Two Dimensional Tile Matrix Set Standard (OGC 17-083r2), originally developed as part of OGC Web Map Tile Service (WMTS) 1.0 Standard, specifies the concepts of a tile matrix set and tile matrix set limits and their implementation in 2D space.

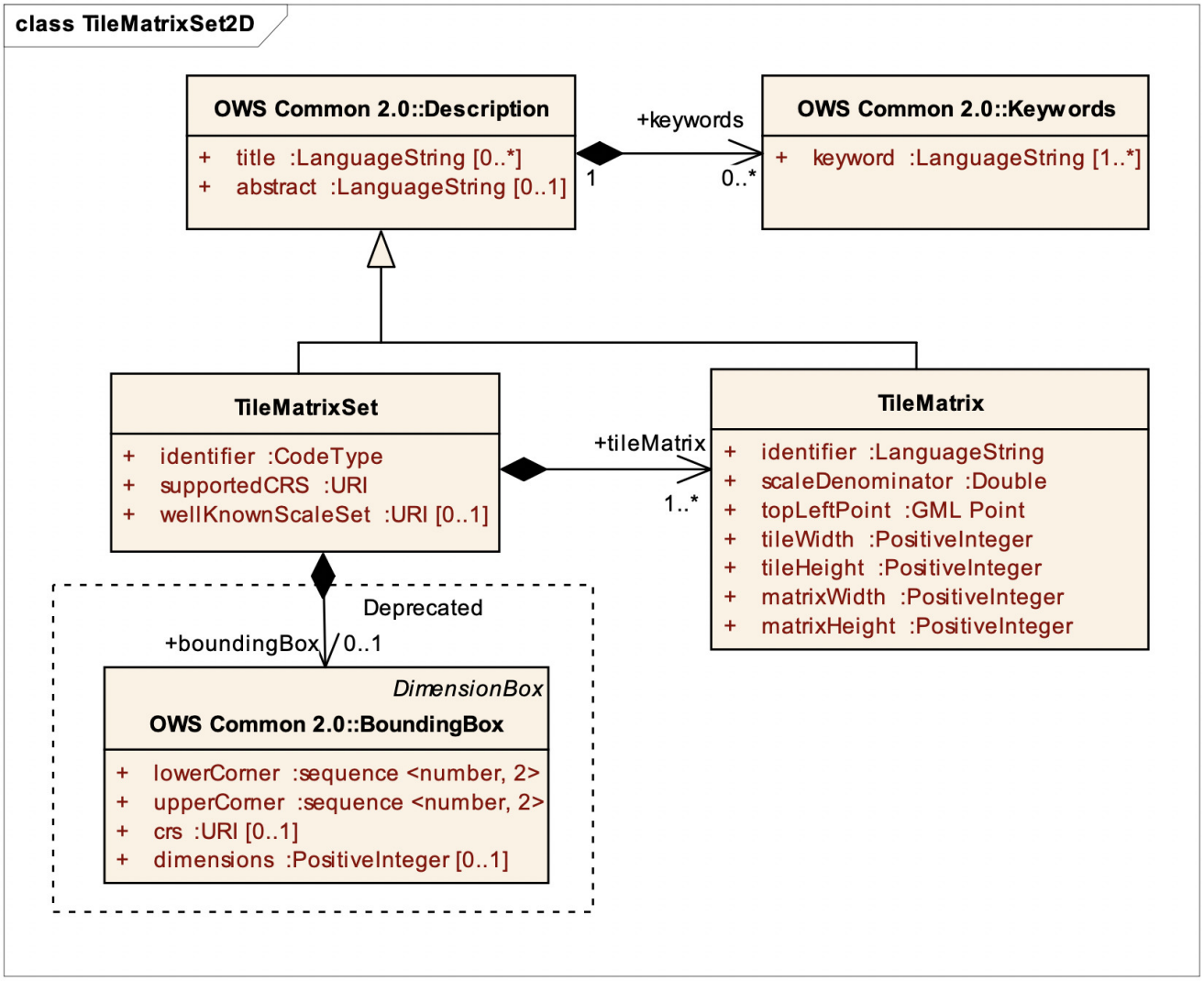

Figure 3. TileMatrixSet UML Model

The OGC Two Dimensional Tile Matrix Set Standard was one of the fundamental contributors to the Tile Set Metadata Model, since it provided concepts and class structures for the scheme description of tile sets. The main elements of the OGC Two Dimensional Tile Matrix Set Standard, as shown in the TileMatrixSet Unified Markup Language (UML) Model in Figure 3, were all included in the Tile Set Metadata Model although some UML notations were changed. Other elements were not described in the same TileMatrixSet UML Model but were nonetheless modeled together in the Tile Set Metadata Model:

- **TileMatrixSetLink, TileMatrixSetLimits and TileMatrixLimits**: as shown in Figure 4, were grouped under a common requirement class.
- **VariableMatrixWidth**: described outside the TileMatrixSet UML Model, but also defined in the Standard as an optional element of a Tile Matrix.

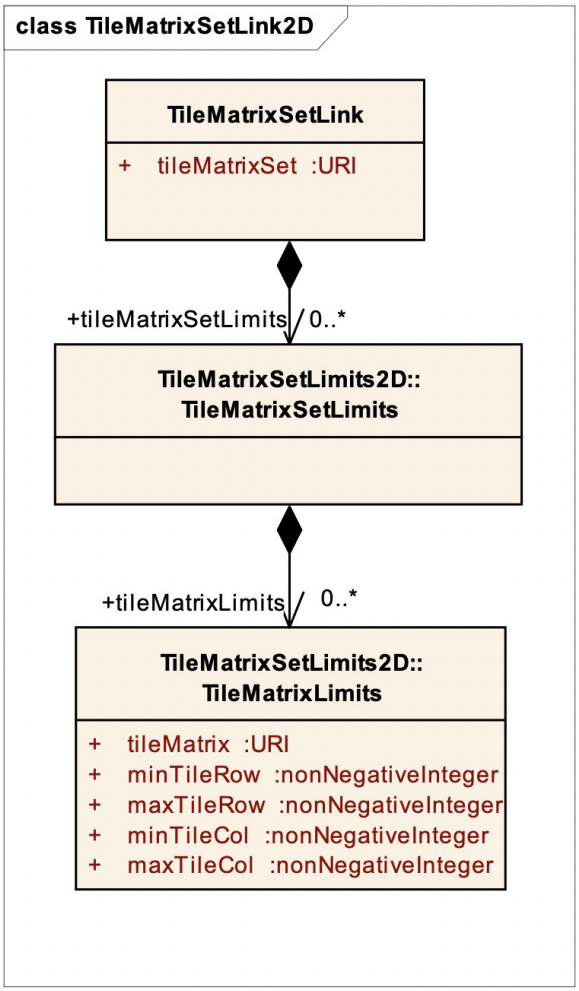

Figure 4. TileMatrixSetLink UML Model

6.3. NSG Metadata Foundation

The National System for Geospatial-Intelligence (NSG) defines itself as *"a unified community of geospatial intelligence (GEOINT) experts, producers and users organized around the goal of integrating technology, policies, capabilities and doctrine to produce GEOINT in a multi-intelligence environment"*. The NSG Metadata Foundation (NMF) is a GEOINT Metadata Standard applicable across the NSG for both data and service resources. [2]

The VTP2 CFP required extending the NSG Metadata Foundation into the Tile Set Metadata. Over the course of the Pilot, some elements were implemented while others were replaced by simpler data structures and deferred for future innovation initiatives to further examine their potential benefits.

As shown on Figure 5, the NMF specifies a central metadata element containing a small number of metadata attributes and associated with other elements described by the standard, each one satisfying a specific requirement related with metadata functionality. As defined by the NSG, the NMF elements and their respective functionalities are:

- **Metadata Scope**: provides the scope/type of the resource for which metadata is provided.
- **Metadata Constraints**: provides information on mandatory restrictions to the access and use of a resource or a set of resource metadata based on ISO 19115-1:2014. The NSG follows

Department of Defense/Intelligence Community (DoD/IC) directives for the use of IC Abstract Data Definition Version 2 (IC.ADD.V2) and extends the Information Security Marking (ISM) elements. Constraints also provide support for ISM Notices and Need-To-Know Metadata.

- **Metadata Identification**: provides a citation identifying the content of the data or service resource. It is composed of over ten classes comprising (among others) title, abstract, point of contact, geographic location, NMF category code, language, character set, keywords, format and revision recalls. There are sub classes designed to provide support to information used by records management systems at different US Government Institutions.
- **Metadata Lineage**: provides information about the sources and/or production processes used in creating the resource.
- **Metadata Reference System**: provides information about a spatial or temporal reference system used by representations in the resource.

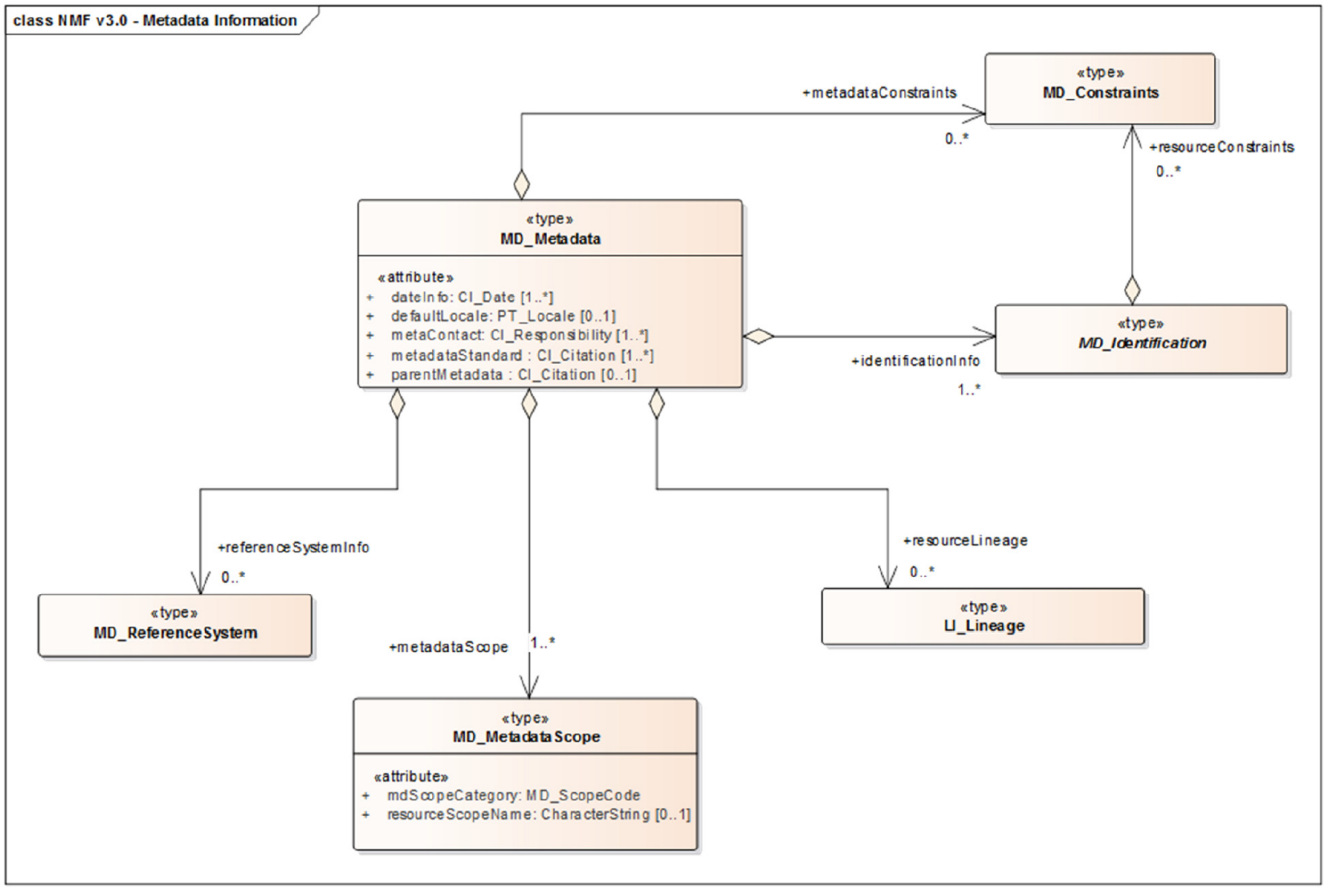

Figure 5. NMF Metadata Information Model

6.4. OGC Testbed-15 Styles API and Styles Metadata ER

The OGC Testbed-15 Open Portrayal Framework (OPF) task in OGC Testbed-15 addressed the issue of the absence of a conceptual model and APIs capable of supporting multiple style encodings.

6.4.1. Styles API

The OGC Testbed-15 Styles API Engineering Report describes a draft specification defining five requirements/conformance classes for a Styles API and two requirements/conformance classes extending the information about Collection resources as specified in OGC API - Features - Part 1:

Core.

Metadata API Resource

One of the resources described as part of the Styles API was the Styles Metadata for use when implementing an API and a resource path to access Tile Set Metadata. The Styles API provided a resource path `styles/{styleId}/metadata` with the `HTTP` methods `GET`, `PUT` and `PATCH` that fetched, replaced and updated style metadata, respectively.

Queryables

In order to support styles, data access APIs (supporting OGC API - Features and/or Tiles) needed additional capabilities to support styling in an Open Portrayal Framework. One of the additional capabilities needed was the `Queryables` Resource, which contains an array with a description of the queryable properties of the feature collection. This resource was proposed to be added to the API in order to support clients such as visual style editors to construct expressions for selection criteria in queries on features in a collection [3].

The path of the resource was set to `/collections/{collectionId}/queryables`. The response was an object with a member `queryables`, containing a list of attribute names paired up with their data types. "Queryable" means that the property may be used in query expressions. For example, this could be used in a CQL (Common Query Language) query expression or as part of a selection criteria in an OGC Styled Layer Descriptor/Symbology Encoding (SLD/SE) or Mapbox styling rule. Often the list will be a subset of all available properties in the features and be restricted to those properties that are, for example, indexed in the backend datastore to support performant queries.

The `Queryables` Resource initially became a major component of the Tile Set Metadata Model. This was because the `queryable` properties were thought to have the capacity to summarize the type of information being stored on a particular Layer (or Collection). An end user would find this information useful when being offline and deciding which Tile Set to choose out of a group. Eventually, `Queryables` was discarded from the metadata model because it was found that not all elements in `Queryables` were actually attributes that could be found and used. Nonetheless, from a conceptual point of view, `Queryables` provided the foundation for the inclusion of feature attribute types in the metadata model.

6.4.2. Styles Metadata

The OGC Testbed-15: Encoding and Metadata Conceptual Model for Styles Engineering Report described a conceptual model for a style encoding and metadata that provided information for understanding a style's intended usage, availability, compatibility with existing layers, as well as supporting style search [4]. Many elements of the Styles Metadata Model (as shown on Figure 6) inspired the design of the Tile Set Metadata Model.

The StyleMetadata and MetadataDate classes, DateType enumeration and the accessConstraints association, were all major components of the early design of the Tile Set Metadata Model.

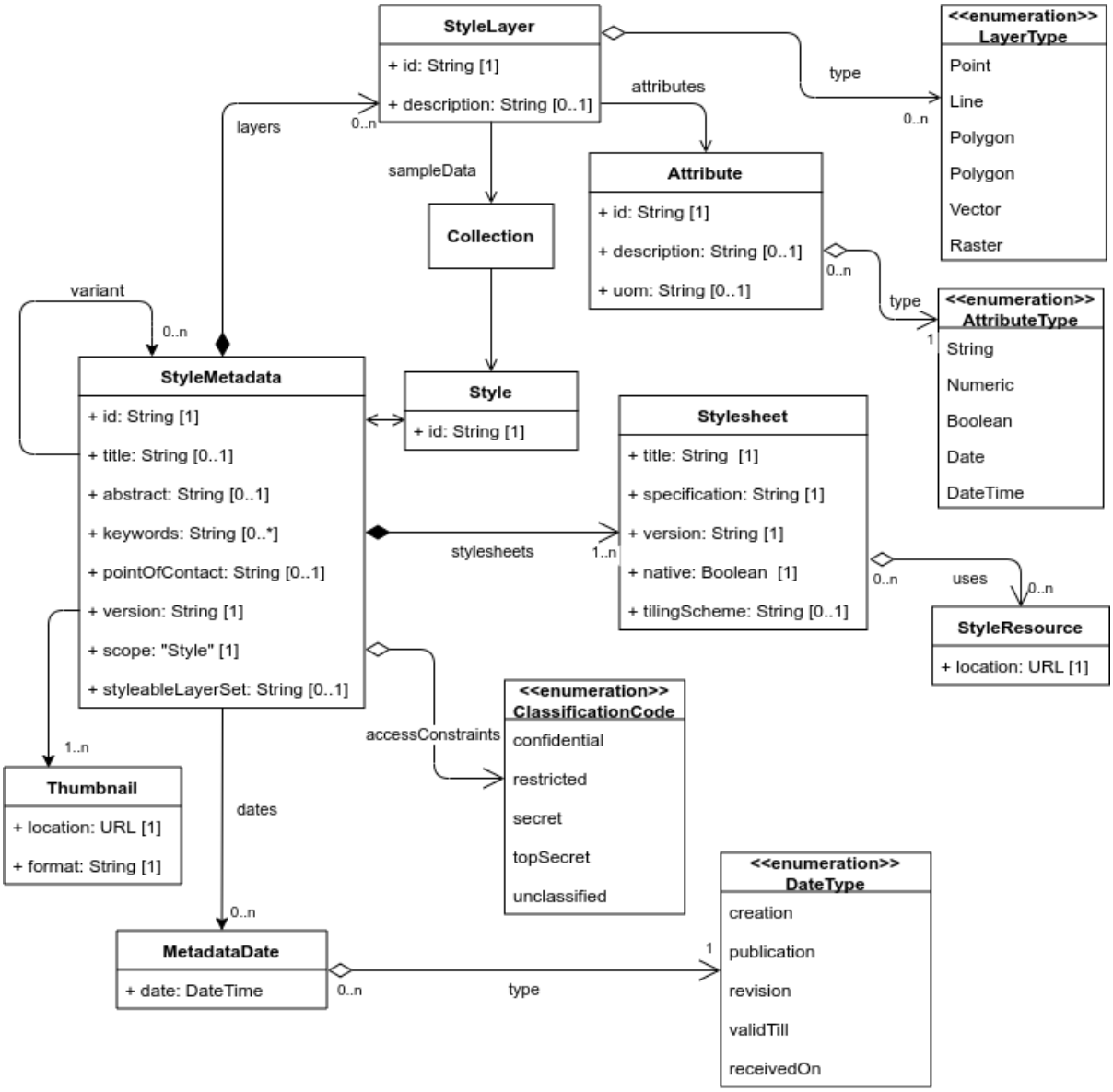

Figure 6. Style metadata and encoding conceptual model

6.5. TileJSON 3.0.0

TileJSON is a specification that provides guidance *for representing metadata about multiple types of web-based map layers, to aid clients in configuration and browsing* [5].

Developed by Mapbox, TileJSON version 3.0.0 was still a draft when this Pilot started. However, the participants decided not to use version 2.2.0 because that version did not support layer information. Further, TileJSON 3.0.0 is compatible with TileJSON 2.2.0. Notably, TileJSON 3.0.0 provides support for multi-layer tiles.

The specification describes two elements in the TileJSON root:

- **"tiles"**: contains an array of tile endpoints. If multiple endpoints are specified, clients may use any combination of endpoints.
- **"vector_layers"**: containing an array of objects, where each object describes one layer of vector tile data. A `vector_layer` contains the following required fields:
  - **id**: a String value representing the layer id.

- **fields**: an object whose keys and values are the names and descriptions of attributes available in this layer.

In the VTP2 initiative, TileJSON became a component of the Tile Set Metadata Model when it was selected for use in the Tiles API to support multi-layer Mapbox Vector Tiles.

6.6. VTP1 Tiled Feature Data Conceptual Model

The Vector Tiles Pilot Phase 1 (VTP1) initiative preceded the VTP2 project. One goal of VTP1 was to define candidate extensions to existing OGC standards as a way to advance the use of vector tiles technology as part of the OGC baseline. This work was done through experimentation with the Web Feature Service (WFS), Web Map Tile Service (WMTS) and GeoPackage standards [6].

As part of VTP1, the *OGC Vector Tiles Pilot: Tiled Feature Data Conceptual Model Engineering Report* provided a draft specification for a Tiled Feature Data Conceptual Model (TFD CM) [7]. Other ERs produced by VTP1 participants documented the work in relation to GeoPackage [8], WMTS [9], and WFS [10] standards. Further work on GeoPackage is documented in the VTP1 Summary Engineering Report [11].

The contribution of the TFD CM in the Tile Set Metadata Model was ultimately reduced in scope although it contributed to shape the initial iterations of the Tile Set Metadata Model. The UML class diagram of the TFD CM specifies the relationship between Tiles, Layers (later renamed "Collections"), General Features and General Feature Properties, but few of these elements were included in the Tile Set Metadata Model.

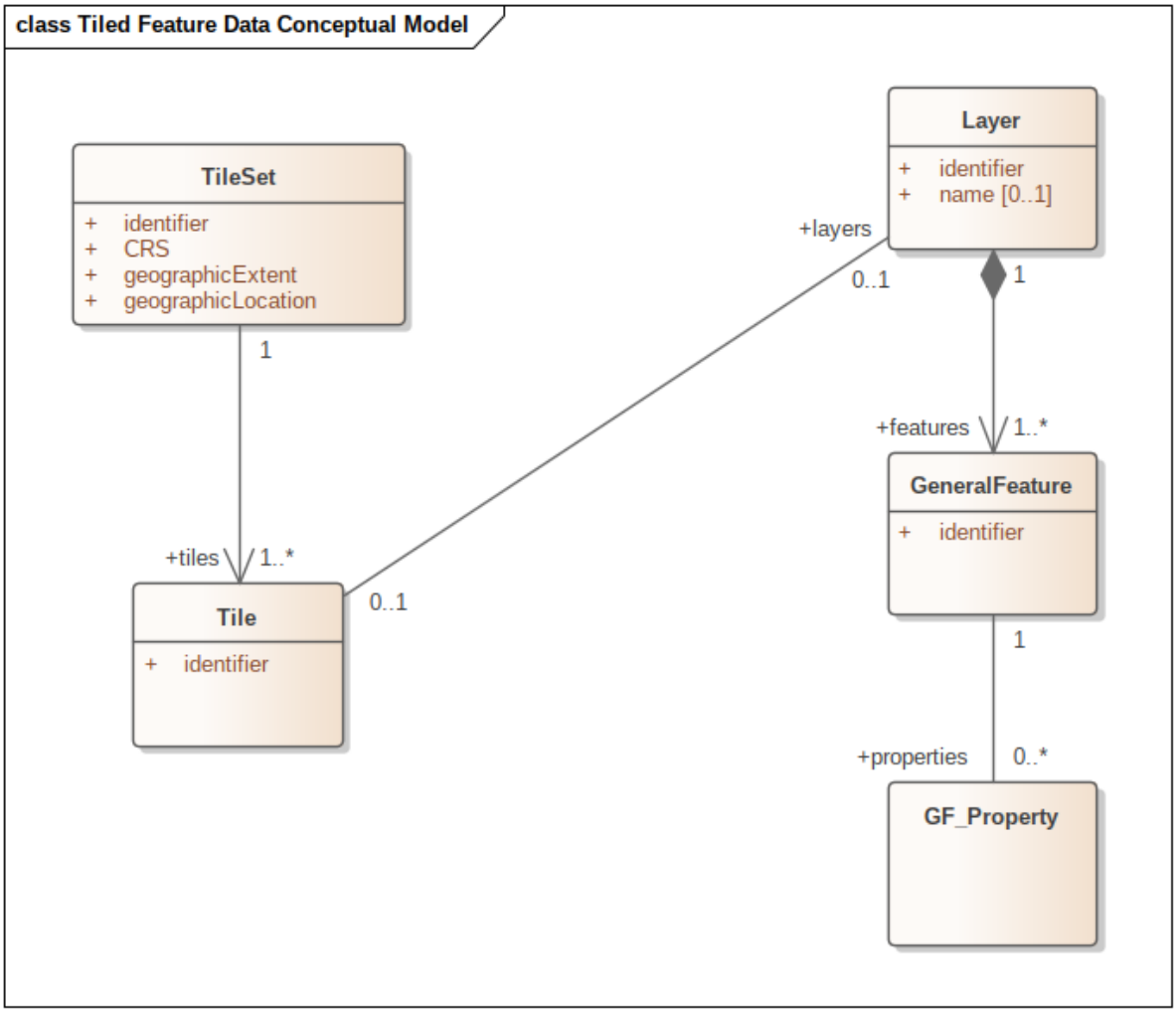

Figure 7. Vector Tiles Conceptual Model

The TFD CM UML diagram, as shown on Figure 7 includes a `TileSet` element, defined as "a definition of how tiles are organized", which constitutes a radically different definition compared with the one used in this ER for the same element name.

The first versions of the TileSet Metadata Model included all the elements of this UML diagram, except the `TileSet` element. Subsequent versions of the Tile Set Metadata Model saw the `Tile`, `GeneralFeature` and `GF_Property` elements eliminated, as:

- They did not represent true metadata information and;
- Accessing the large number of Tiles, Features or Feature Properties that a Tile Set would have been against the basic principle of summarizing that any metadata model should implement as well as the requirement of the Pilot of not accessing each individual tile.

Chapter 7. Tile Set Metadata Model

7.1. Overview

The Tile Set Metadata Model is the conceptual foundation for describing the metadata for a tile set. The model contains many of the elements required to describe the main characteristics of a tile set. The conceptual model covering Tile Set metadata is summarized in the UML class diagram in Figure 8.

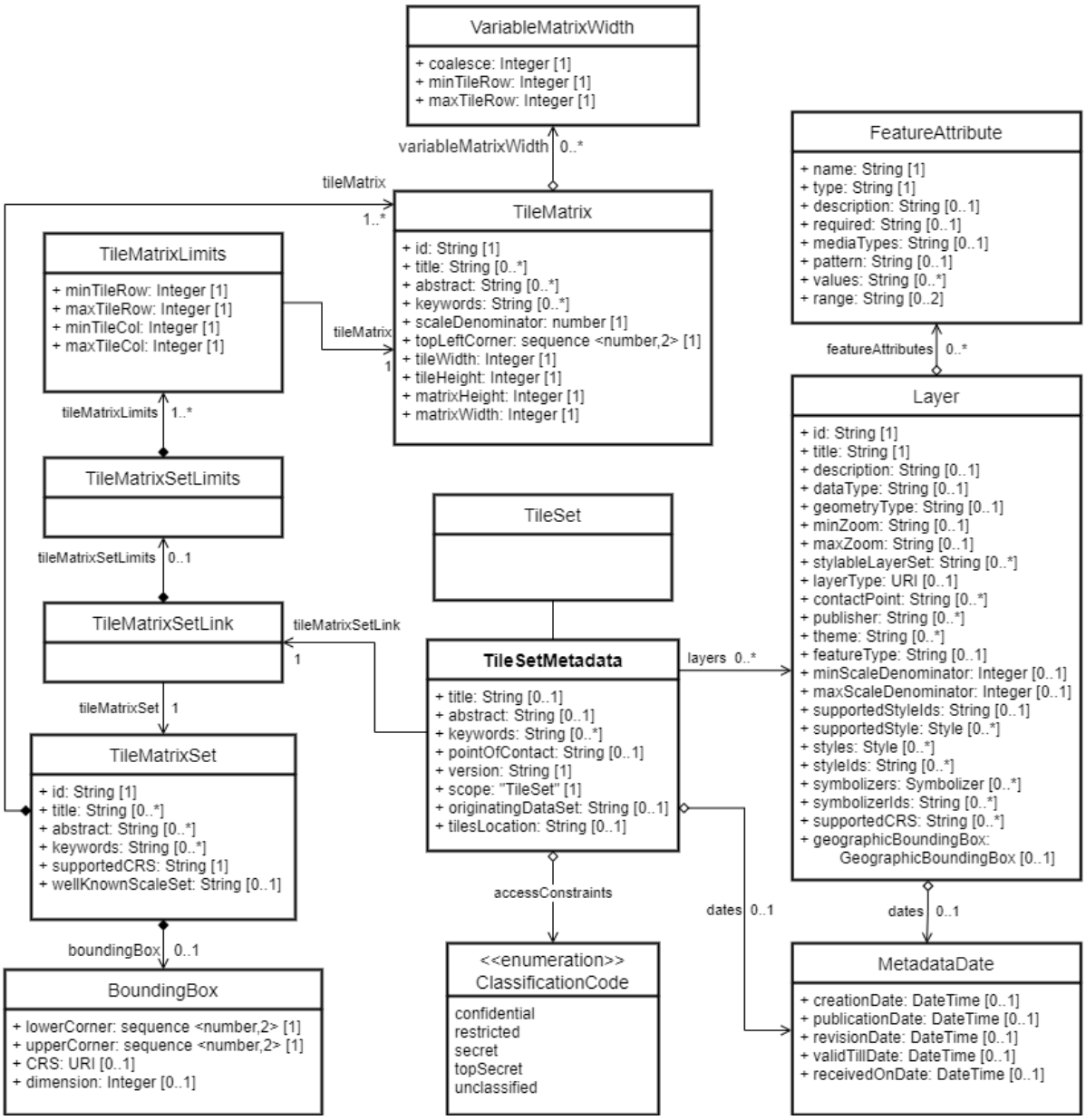

Figure 8. The Tile Set Metadata Model

The model can be succinctly divided into three main components:

- The `TileSet` and `TileSetMetadata`, comprising the two major components of the model, where every other major component in the model is connected to.
- The `TileMatrixSet`, `BoundingBox`, `TileMatrix`, `TileMatrixLimits` and `VariableMatrixWidth` represent the tiling scheme of the tile set as defined by the OGC Tile Matrix Set Standard.

- The `Layers` and `FeatureAttributes` classes are associated, the latter comprising the actual feature properties that could be found within the Tile Set.

7.2. Elements

7.2.1. TileSet

The TileSet class defines how tiles are organized. This class contains a definition of the geographic extent and geographic location as well as the CRS. One tile set contains one to many tiles.

7.2.2. TileSetMetadata

The TileSetMetadata class describes the tile set with a simple metadata entry, as well as some structural information. The elements for this class are defined in Table 1.

Table 1. Structure of TileSetMetadata

Attribute	Definition
title	Title of the tile set, normally used for display to a human
abstract	Brief narrative description of the tile set, normally available for display to a human
keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this tile set..
pointOfContact	Information that can be used to contact the authors or custodians for the Tile Set (free form, can be anything from an e-mail address to physical address and phone numbers)
version	Version number for the Tile Set.
scope	Fixed value of 'TileSet'.
originatingDataSet	Reference of the origin of the Tile Set.
tilesLocation	Description of the location of tiles. Can be a folder in an offline file system, or a Uniform Resource Locator (URL) in an online environment.

Attribute	Definition
accessConstraints	Indication about the availability of the Tile Set that the user with access to the Tile Set needs to be aware of before using or redistributing the Tile Set in question. Possible values are confidential, restricted, secret, topSecret and unclassified.
dates	Collection of creation, publication, revision, valid until and received on dates that can be associated with the tile set
layers	A collection of one or many Layer elements the tile set possesses
tileMatrixSetLink	A TileMatrixSetLink element

7.2.3. MetadataDate

A Tile Set or a Layer can contain a single date entry of each of the types defined in Table 2.

Table 2. Structure of MetadataDate

Attribute	Definition
creationDate	Timestamp when the Tile Set was first produced.
publicationDate	Timestamp when the Tile Set was first made available to the users.
revisionDate	Timestamp of the last Tile Set change/revision.
validTillDate	Timestamp marking the future validity of the Tile Set (the Tile Set may no longer be applicable at this date, or that a new revision of the Tile Set is going to be issued).
receivedOnDate	Timestamp indicating when the Tile Set was received from an external provider.

7.2.4. TileMatrix

The TileMatrix class describes a tiling scheme defined as a grid coverage. Its elements are defined in Table 3[12].

Table 3. Structure of TileMatrix

Attribute	Definition
id	Tile matrix identifier
title	Title of this style, normally used for display to a human
abstract	Brief narrative description of this style, normally available for display to a human
keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this dataset
scaleDenominator	Scale denominator level of this tile matrix
topLeftCorner	Position in CRS coordinates of the top-left corner of this tile matrix
tileWidth	Width of each tile of this tile matrix in pixels
tileHeight	Height of each tile of this tile matrix in pixels
matrixHeight	Height of the matrix (number of tiles in height)
matrixWidth	Width of the matrix (number of tiles in width)
variableMatrixWidth	Reference to a VariableMatrixWidth element, if needed

7.2.5. TileMatrixSet

The TileMatrixSet class is a tiling scheme composed of a collection of tile matrices, optimized for a particular scale and identified by a tile matrix identifier. Its elements are defined in Table 4[12].

Table 4. Structure of TileMatrixSet

Attribute	Definition
id	Tile matrix set identifier
title	Title of this tile matrix set, normally used for display to a human

Attribute	Definition
abstract	Brief narrative description of this tile matrix set, normally available for display to a human
keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this dataset
supportedCRS	Reference to one coordinate reference system (CRS)
wellKnownScaleSet	Reference to a well-known scale set
tileMatrix	Describes a scale level and its tile matrix
boundingBox	Minimum bounding rectangle surrounding the tile matrix set, in the supported CRS

7.2.6. BoundingBox

The BoundingBox class describes a Minimum Bounding Rectangle (MBR) surrounding the tile matrix set, in the supported CRS.

Table 5. Structure of BoundingBox

Attribute	Definition
lowerCorner	Southern-most coordinate of the limit of the Tile Matrix Set extent
upperCorner	Northern-most coordinate of the limit of the Tile Matrix Set extent
CRS	Coordinate reference system
dimension	The number of dimensions defined in the coordinate reference system

7.2.7. VariableMatrixWidth

When a tiled resource or dataset has variable width tiles, the resource or dataset shall define the variable matrix width in a tile matrix set. Its elements are defined in Table 6[12].

Table 6. Structure of VariableMatrixWidth

Attribute	Definition
coalesce	Coalescence factor

Attribute	Definition
minTileRow	Minimum tile row index valid for this layer
maxTileRow	Maximum tile row index valid for this layer

7.2.8. TileMatrixSetLink

The TileMatrixSetLink class serves as a container of both a Tile Matrix Set and, if existent, the limits defined to it by a TileMatrixSetLimits element. [12]

Table 7. Structure of TileMatrixSetLink

Attribute	Definition
tileMatrixSetLimits	Provides the instance of TileMatrixSetLimits, if available.
tileMatrixSet	Provides the instance of TileMatrixSet

7.2.9. TileMatrixSetLimits

The TileMatrixSetLimits class contains a collection of TileMatrixLimits elements, therefore comprising the extent of limits applied to all the Tile Matrices that are part of the Tile Matrix Set.

Table 8. Structure of TileMatrixSetLimits

Attribute	Definition
tileMatrixLimits	Provides the collection of instances of TileMatrixLimits.

7.2.10. TileMatrixLimits

The TileMatrixLimits class informs the minimum and maximum limits of the tile indices for each Tile Matrix that contains actual data. The area outside these limits is considered empty space. Its elements are defined in Table 9[12].

Table 9. Structure of TileMatrixLimits

Attribute	Definition
tileMatrix	Reference to a tileMatrix identifier
minTileRow	Minimum tile row index valid for this layer
maxTileRow	Maximum tile row index valid for this layer

Attribute	Definition
minTileCol	Minimum tile column index valid for this layer
maxTileCol	Maximum tile column index valid for this layer

7.2.11. Layer

A Layer is a set of geographic objects of the same type. Its elements are defined in Table 10.

Table 10. Structure of Layer

Attribute	Definition
id	Unique identifier of the Layer
title	Title of this Layer, normally used for display to a human
description	Description of this Layer, more detailed than its title
dataType	Specifies whether the layer contains raster or vector tiles
geometryType	The geometry type of the tiles of this layer
minZoom	Minimum zoom level
maxZoom	Maximum zoom level
stylableLayerSet	StylableLayerSets the Layer belongs to
layerType	Concept Uniform Resource Identifier (URI) of layer type taxonomy [13]
contactPoint	This property contains contact information that can be used for sending comments about the layer [13]
publisher	This property refers to an entity (organization) responsible for making the layer available [13]
theme	This property refers to a category of the layer. A layer may be associated with multiple themes. [13]
featureType	Feature type identifier [13]

Attribute	Definition
minScaleDenominator	Suggested Minimum scale denominator for usage of the layer [13]
maxScaleDenominator	Suggested Maximum scale denominator for usage of the layer [13]
supportedStyleIds	Reference to style identifiers managed by the service [13]
supportedStyle	References to supported Styles (in RDF) [13]
styles	Reference to Style specification (inline or by URI) in RDF [13]
styleIds	Reference to style identifiers (JSON) managed by the service [13]
symbolizers	Reference to Symbolizer Object (inline or by URI)[13]
symbolizerIds	Reference to symbolizer ids managed by service (JSON) [13]
supportedCRS	Reference to standard identifier (use of URL) [13]
geographicBoundingBox	Geographic Bounding Box of the layer (can be calculated on server side when layer is submitted) [13]
dates	Collection of creation, publication, revision, valid until and received on dates that can be associated with the layer
featureAttributes	Collection of FeatureAttribute elements

7.2.12. FeatureAttribute

A FeatureAttribute element contains attributes that can be found in at least one feature belonging to the layer the FeatureAttribute element belongs to. Its elements are defined in Table 11.

Table 11. Structure of FeatureAttribute

Attribute	Definition
name	The property identifier for use in expressions
type	The data type of the property

Attribute	Definition
description	Description of the property
required	Indicator whether the property is always present in features
mediaTypes	Feature encodings for which the feature is limited to
pattern	Regular expression to validate the values of the property
values	Array of valid values of the property
range	Range of valid values expressed as an array with two items

7.2.13. ClassificationCode - Enumeration

The level of classification applicable to the tile set.

Table 12. Structure of ClassificationCode

Name	Definition
unclassified	Available for general disclosure
restricted	Not for general disclosure
confidential	Available for someone who can be entrusted with information
secret	Kept or meant to be kept private, unknown, or hidden from all but a select group of people
topSecret	Of the highest secrecy

7.3. Origin of data

The origin of the data that make up a Tile Set Metadata varies according to the TSM element and are described in Table 13

Table 13. Origin of Tile Set Metadata Elements

TileSet Metadata Element	Origin API	Path	Comments
TileSet Metadata	No API	No path	Generated by the Tile Set creator during Tile Set generation

TileSet Metadata Element	Origin API	Path	Comments
TileMatrixSet and TileMatrix	Tiles API	/tileMatrixSets/{tileMatrixSetId}/	
		/collections/{collectionId}/tiles/tileMatrixSets/{tileMatrixSetId}/	
TileMatrixLimits	Tiles API	/collections/{collectionId}/tiles/	
Layer	No API	No path	Exclusive for layer metadata attributes (e.g., dates, contact point, publisher)
	Features API	/collections/{collectionId}/	Exclusive for single-layer tiles
	Tiles API	/collections/{collectionId}/	Exclusive for single-layer tiles
	Tiles API	/tiles/{tileMatrixSetID}/metadata	"vector_layers" element from enclosed TileJSON V3
FeatureAttributes	Tiles API	/collections/{collectionId}/tiles/{tileMatrixSetID}/metadata	"fields" sub element from each "vector_layers" element from enclosed TileJSON V3
		/tiles/{tileMatrixSetID}/metadata	

NOTE In case the Tile Set has Mapbox Vector Tiles (MVTs), the Collection and Feature Attributes elements must be retrieved utilizing the TileJSON V3 Metadata resource, accessed through the link found in the "describedby" relation of the corresponding Tile Matrix Set. This process can also be executed for Tile Sets with single-layer tiles.

7.4. Creation, Storage and Access

7.4.1. Standalone JSON

The Tile Set Metadata is stored in JSON encoding as a stand-alone file located in the folder containing the tile repository. Clients may access this JSON file to look up for Tile Set Metadata without requiring to access the tiles that make up the Tile Set.

The Tile Set Metadata is fully created when the actual Tile Set is generated; most of its content can indeed be created before, but some elements (e.g., Tile Set creation datetime) must be defined during Tile Set generation. Many TSM elements are retrieved through different OGC API resources, as specified in Table 13.

7.4.2. GeoPackage Loading

TileSet metadata is required to store tiled feature data in a GeoPackage. The TileSet metadata contains the details of layers and feature properties (fields) that are needed to make full use of the vector tiles layers. In VTP2, GeoPackage producers read TileSet metadata in the form of TileJSON [https://github.com/mapbox/tilejson-spec/tree/3.0/3.0.0] documents.

The primary purpose of the TileSet metadata is to populate the tables that are part of the Vector Tiles Extension (see the Summary ER). This extension includes two tables, `gpkgext_vt_layers` and `gpkgext_vt_fields`, that are designed to mirror the vector tileset metadata [https://github.com/mapbox/mbtiles-spec/blob/master/1.3/spec.md#vector_layers] of the MBTiles specification. The layers table mirrors the `vector_layers` key and the fields table mirrors the `fields` key. The MBTiles specification permits three possible field types, "Number", "Boolean" and "String". However, in this pilot, some participants used a wider range of field types. The GeoPackage Producers compensated by mapping to the allowed field types.

For completeness, it is also appropriate to store the TileJSON in the GeoPackage via the Metadata Extension [http://www.geopackage.org/spec121/#extension_metadata]. Once the Metadata Extension is enabled through an appropriate row in `gpkg_extensions`, the TileJSON may be inserted in `gpkg_metadata` as follows:

- `id`: integer primary key
- `md_scope`: "dataset"
- `md_standard_uri`: "https://github.com/mapbox/tilejson-spec/tree/3.0/3.0.0"
- `mime_type`: "application/json"
- `metadata`: the TileJSON document

The `gpkg_metadata` row is linked to a tiles table through `gpkg_metadata_reference` as follows:

- `reference_scope`: "table"
- `table_name`: the tiles table name
- `column_name`: *null*
- `row_id_value`: *null*
- `timestamp`: the current date and time
- `md_file_id`: a foreign key to `gpkg_metadata.id`
- `md_parent_file_id`: *null*

There is not currently a defined use for this metadata, but it may be inspected by GeoPackage clients.

Chapter 8. Implementations

In order to validate the proposed metadata model, a series of tests were performed by the pilot participants. The feedback provided by these TIEs helped shape the final version of the metadata model.

Three different encodings of the tile set metadata are considered in this section:

- TileJSON Metadata: consisting of importing Tiles Metadata from client providers, stored as TileJSON encodings.
- Reference JSON Encoding of Tile Set Metadata: consisting of implementing a JSON encoding of the Tile Set Metadata Conceptual Model.
- GeoPackage Tile Set Metadata: consisting of encoding Tile Set Metadata in GeoPackage relational tables.

Experiments were conducted for each encoding.

8.1. TileJSON Metadata TIEs

All four participants successfully advertised TileJSON Metadata from their respective APIs. Additionally, all servers were successfully accessed by at least one of the client applications developed by the participants throughout this Pilot, resulting in the TIE matrix Table 14. Details for the Tiles URL, the TileJSON URL template and the supported Tile Matrix Sets by each participant's API can be found in Table 15.

Table 14. TileJSON Technology Integration Experiments (TIE)

	GeoSolutions D104 Client	**Skymantics D104 Client**	**Ecere D105 Client**	**Image Matters Client**	**Terranodo Client (in kind)**
Ecere D103 Tiles API	X		X	X	X
GeoSolutions D102 Tiles API	X	X	X	X	X
interactive instruments D101 Tiles API	X	X	X	X	X
Terranodo D100 Tiles API	X	X	X	X	X

Table 15. TileJSON Metadata Specifications

Participant	Tiles URL	TileJSON URL template	Tile Matrix Sets
Ecere	http://maps.ecere.com/geoapi/collections/vtp/Daraa2/tiles?f=json	http://maps.ecere.com/geoapi/collections/vtp/Daraa2/tiles/{tileMatrixSetId}/metadata	CDBGlobalGrid, GlobalCRS84Pixel, GlobalCRS84Scale, GNOSISGlobalGrid, GoogleCRS84Quad, WebMercatorQuad, WorldCRS84Quad, WorldMercatorWGS84Quad
GeoSolutions	http://vtp2.geo-solutions.it/geoserver/ogc/tiles/collections/vtp:daraa_vtp/tiles?f=application%2Fjson	http://vtp2.geo-solutions.it/geoserver/ogc/tiles/collections/vtp:daraa_vtp/tiles/{tileMatrixSetId}/metadata	WebMercatorQuad, WorldCRS84Quad, WorldMercatorWGS84Quad, EPSG:4326, EPSG:4326_512, EPSG:900913
interactive instruments	https://services.interactive-instruments.de/t15/daraa/tiles?f=json	https://services.interactive-instruments.de/t15/daraa/tiles/{tileMatrixSetId}/metadata	WebMercatorQuad, WorldCRS84Quad, WorldMercatorWGS84Quad
Terranodo	https://ogc-vtp.gospatial.org/ogc-api-tiles/tiles	https://ogc-vtp.gospatial.org/ogc-api-tiles/tiles/{tileMatrixSetId}/metadata	WebMercatorQuad, WorldCRS84Quad

8.1.1. interactive instruments

A design goal in OGC APIs is to prioritize reuse of existing building blocks over specifying new ones. The following approach was taken by interactive instruments to representing additional metadata about the API offerings based on existing specifications with broad tool support.

For each tile set in the Mapbox Vector Tiles encoding, a TileJSON document was provided as tile set metadata complementing the tile set metadata that was already provided based on the draft OGC API Tiles specification. This was implemented as shown in the code block of the Tile Sets resource shown below, i.e., a path that ends in `tiles`:

- There is one tile set per tile matrix set and tile encoding.
- The three available tile matrix sets are listed in `tileMatrixSet` / `tileMatrixSetURI` members.
- The available tile encodings are identified in the `type` members of the link templates with `rel=item`, in this case only Mapbox Vector Tiles are provided.
- The available tiles per tile matrix are listed in the `tileMatrixSetLimits` members.
- `title` and `description` provide general information about the contents of the tile sets.

- To provide access to tile set metadata as a single document, a link template with `rel=describedby` was added pointing to a TileJSON document. The link template includes the `{tileMatrixSetId}` parameter since there are differences in the metadata for each tile set.

interactive instruments - an abbreviated Tile Sets resource


```
{
  "title" : "Daraa",
  "description" : "This is a test dataset for the Open Portrayal Framework thread in
the OGC Testbed-15 as well as for the OGC Vector Tiles Pilot Phase 2. The data is
OpenStreetMap data from the region of Daraa, Syria, converted to the Topographic Data
Store schema of NGA.",
  "links" : [ {
    "rel" : "self",
    "type" : "application/json",
    "title" : "This document",
    "href" : "https://services.interactive-instruments.de/t15/daraa/tiles?f=json"
  }, {
    "rel" : "alternate",
    "type" : "text/html",
    "title" : "This document as HTML",
    "href" : "https://services.interactive-instruments.de/t15/daraa/tiles?f=html"
  }, {
    "rel" : "item",
    "type" : "application/vnd.mapbox-vector-tile",
    "title" : "Mapbox vector tiles; the link is a URI template where
{tileMatrix}/{tileRow}/{tileCol} is the tile based on the tiling scheme
{tileMatrixSetId}",
    "href" : "https://services.interactive-
instruments.de/t15/daraa/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}?f=mv
t",
    "templated" : true
  }, {
    "rel" : "describedby",
    "type" : "application/json",
    "title" : "Tile Set metadata in the tilejson format",
    "href" : "https://services.interactive-
instruments.de/t15/daraa/tiles/{tileMatrixSetId}/metadata",
    "templated" : true
  } ],
  "tileMatrixSetLinks" : [ {
    "tileMatrixSet" : "WebMercatorQuad",
    "tileMatrixSetURI" : "https://services.interactive-
instruments.de/t15/daraa/tileMatrixSets/WebMercatorQuad",
    "tileMatrixSetLimits" : [ {
      "tileMatrix" : "6",
      "minTileRow" : 25,
      "maxTileRow" : 25,
      "minTileCol" : 38,
      "maxTileCol" : 38
    }, ... {
```

```
      "tileMatrix" : "18",
      "minTileRow" : 105359,
      "maxTileRow" : 106147,
      "minTileCol" : 157108,
      "maxTileCol" : 158164
    } ]
  }, {
    "tileMatrixSet" : "WorldCRS84Quad",
    "tileMatrixSetURI" : "https://services.interactive-
instruments.de/t15/daraa/tileMatrixSets/WorldCRS84Quad",
    "tileMatrixSetLimits" : [ ... ]
  }, {
    "tileMatrixSet" : "WorldMercatorWGS84Quad",
    "tileMatrixSetURI" : "https://services.interactive-
instruments.de/t15/daraa/tileMatrixSets/WorldMercatorWGS84Quad",
    "tileMatrixSetLimits" : [ ... ]
  } ]
}
```

The TileJSON document for the tile set with all collections in the Daraa dataset in the `WorldMercatorWGS84Quad` tiling scheme is shown in the following code block (only two of the collections are shown).

interactive instruments - an abbreviated TileJSON document

```
{
  "tilejson" : "3.0.0",
  "name" : "Daraa",
  "description" : "This is a test dataset for the Open Portrayal Framework thread in
the OGC Testbed-15 as well as for the OGC Vector Tiles Pilot Phase 2. The data is
OpenStreetMap data from the region of Daraa, Syria, converted to the Topographic Data
Store schema of NGA.",
  "tiles" : [ "https://services.interactive-
instruments.de/t15/daraa/tiles/WorldMercatorWGS84Quad/{z}/{y}/{x}?f=mvt" ],
  "bounds" : [ 35.7550727, 32.3573507, 37.2052764, 33.2671397 ],
  "minzoom" : 6,
  "maxzoom" : 18,
  "center" : [ 36.48017455, 32.8122452, 12 ],
  "vector_layers" : [ {
    "id" : "AeronauticCrv",
    "description" : "Aeronautical Facilities: Information about an area specifically
designed and constructed for landing, accommodating and launching military and/or
civilian aircraft, rockets, missiles and/or spacecraft.<br/>Aeronautical Aids to
Navigation: Information about electronic equipment, housings, and utilities that
provide positional information for direction or otherwise assisting in the navigation
of airborne aircraft.",
    "minzoom" : 6,
    "maxzoom" : 18,
    "geometry_type" : "line",
    "fields" : {
      "id" : "string",
```

```
      "F_CODE" : "string",
      "ZI001_SDV" : "dateTime",
      "UFI" : "string",
      "ZI005_FNA" : "string",
      "FCSUBTYPE" : "integer",
      "ZI006_MEM" : "string",
      "ZI001_SDP" : "string"
    }
  }, ... {
    "id" : "VegetationSrf",
    "description" : "Vegetation: Information about the plant life in an area, or the
lack thereof.",
    "minzoom" : 6,
    "maxzoom" : 18,
    "geometry_type" : "polygon",
    "fields" : {
      "id" : "string",
      "F_CODE" : "string",
      "ZI001_SDV" : "dateTime",
      "UFI" : "string",
      "ZI005_FNA" : "string",
      "FCSUBTYPE" : "integer",
      "ZI024_HYP" : "integer",
      "ZI006_MEM" : "string",
      "ZI001_SDP" : "string",
      "VEG" : "integer"
    }
  } ]
}
```

The combination of the information about the URI template of the tile set, the tiling scheme / tile matrix set used and the TileJSON is sufficient for most clients to handle and process the tiles in a tile set. The additional information in the `tileMatrixSetLimits` is to some extent redundant with the bounding box information and currently vector tiles clients typically ignore this information.

Despite the fact the latest version of TileJSON is 2.2.0, this implementation included also the additional `vector_layers` member that is part of the draft of TileJSON 3.0.0 as this information was essential for the GeoPackage providers in the pilot.

8.1.2. GeoSolutions

GeoSolutions used TileJSON metadata information in the client application in the following ways:

- populate the featureAttributes property of the downloaded Tile Set Metadata JSON Figure 14
- create a style on the fly using the information inside the vector_layers property: geometry_type and id; when MapStore is using the Mapbox GL library

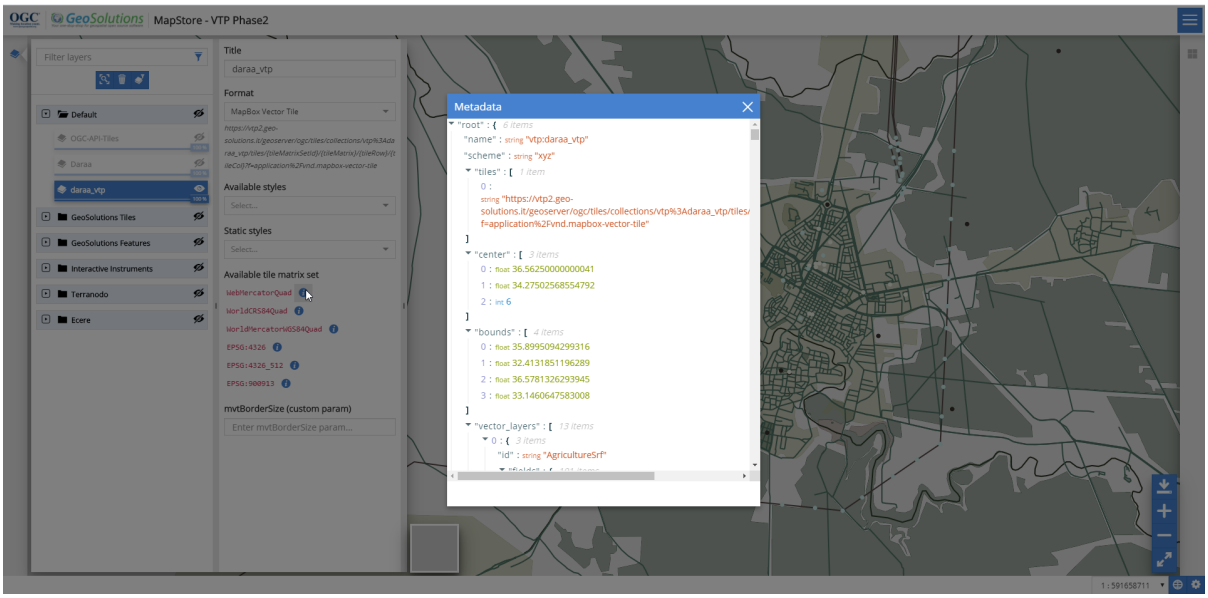

Figure 9. GeoSolutions - MapStore web client - display information from tilejson metadata in layer settings when available

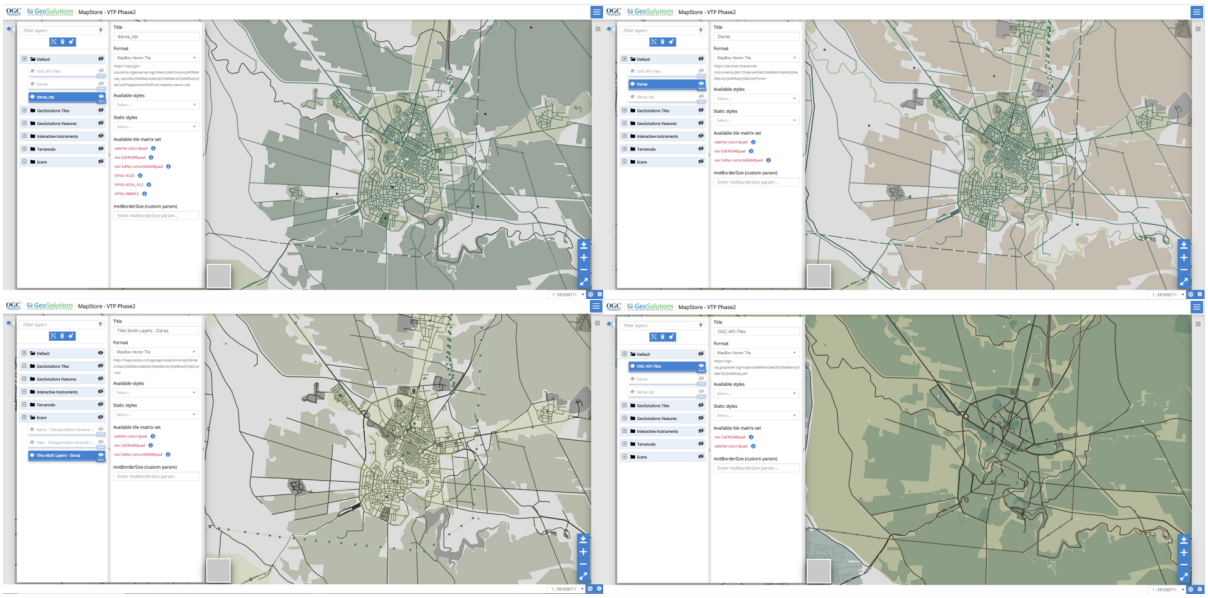

Figure 10. GeoSolutions - MapStore web client - generate on the fly mapbox style based on the vector_layers information of tilejson metadata from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

8.1.3. Image Matters

As described by GeoPackage Loading, Tile Set metadata is required to populate a GeoPackage properly, and this was obtained from Tiles APIs through the TileJSON document.

A successful TIE consists of the following:

1. Reading a Tiles Capabilities JSON document (i.e., a collection) representing a multi-layer collection with a URL pattern like `⋯/{collectionId}`
2. Reading a Tiles JSON document representing a multi-layer tiles set via a link with a `rel` of "tiles" (these conventionally have a URL pattern like `⋯/{collectionId}/tiles`
3. Reading a Tiles Metadata TileJSON document via a link with a `rel` of "describedby" (these

conventionally have a URL pattern like `…/{collectionId}/tiles/{tileMatrixSetId}/metadata`

Through the GeoPackage Producer, Image Matters TIEd successfully with each of the four vendors.

8.1.4. Ecere

Ecere provided tile set metadata encoded as TileJSON from its Tiles API. This information is available for both multi-layer collection (e.g. at *../collections/vtp/Daraa2/tiles/{tileMatrixSetId}/metadata*) and for individual layers (e.g. at *../collections/vtp/Daraa2/tiles/{tileMatrixSetId}/metadata*).

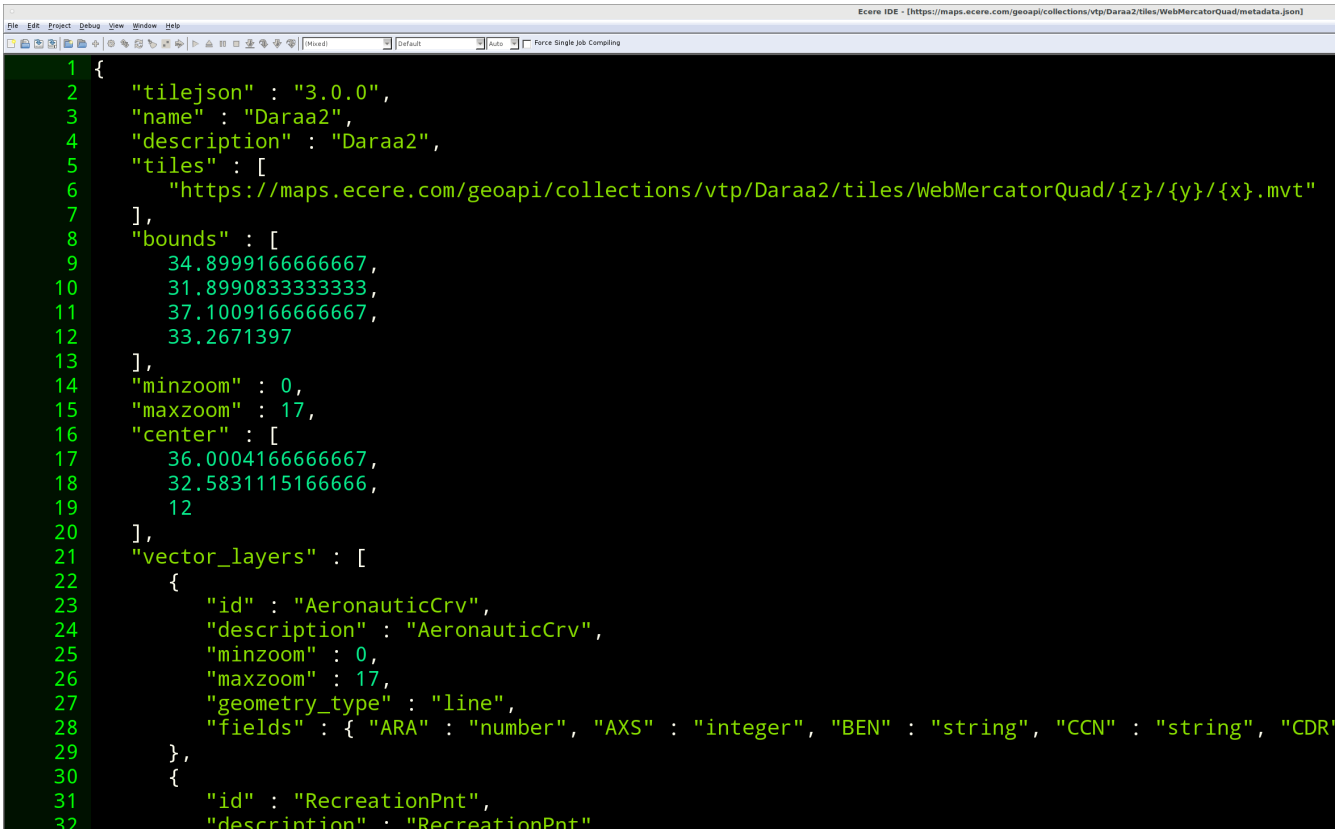

Figure 11. TileJSON as served by the Ecere Tiles API

Ecere's visualization client made use of TileJSON from all Tiles API providers to obtain the tiles template to use for accessing vector tiles, and to learn about the layers and data fields contained within them, the extent of the data, the maximum zoom level available, as well as the geometry type (points, lines or polygons).

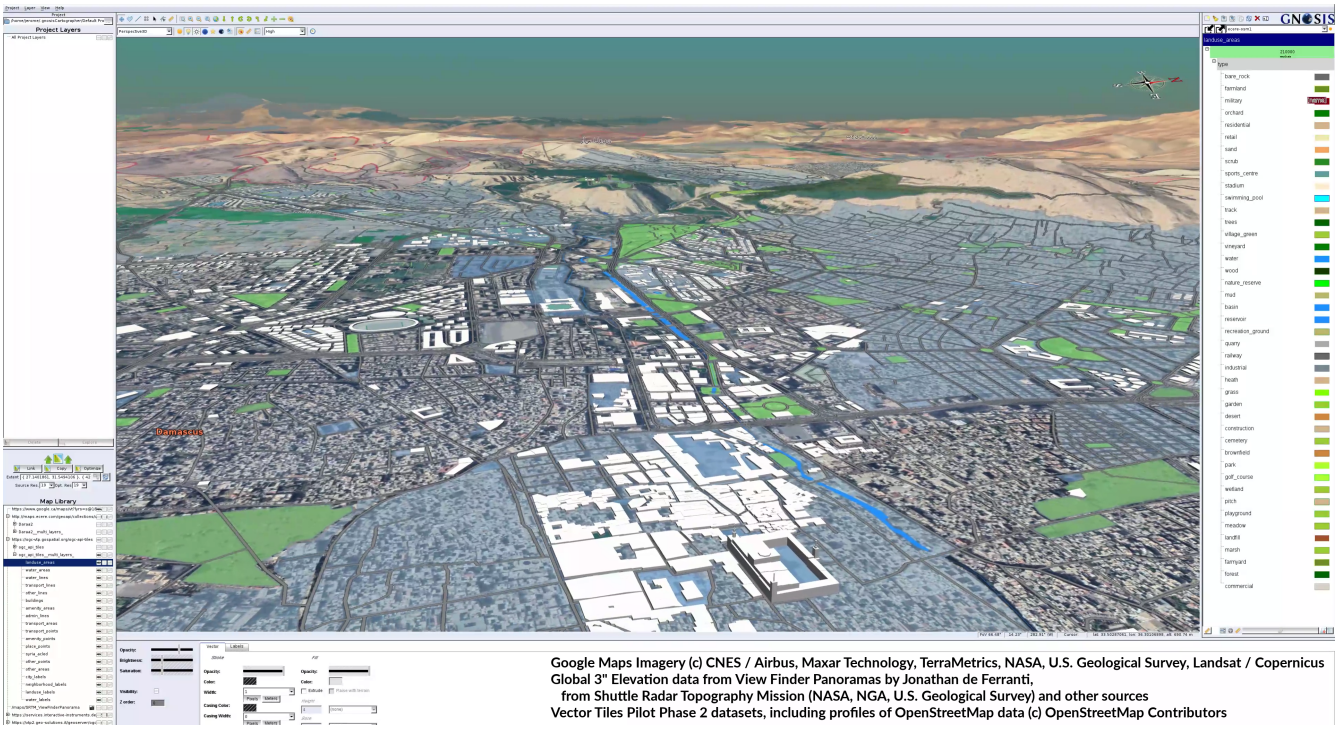

Figure 12. Ecere's GNOSIS Cartographer displaying multi-layer tiles described using TileJSON from Terranodo Tiles API

8.2. Reference JSON Encoding of Tile Set Metadata

8.2.1. GeoSolutions

GeoSolutions developed WebGIS applications based on the Open Source Framework MapStore [https://mapstore.geo-solutions.it/] aimed to show a complete workflow where tile set metadata were in use. The clients developed were two:

- A MapStore client working on the browser to generate a ZIP file containing information related tiles
  - live demo [http://demo.vtp2.geo-solutions.it/mapstore/index.html#/]
  - repository [https://github.com/geosolutions-it/ogc-vector-tiles-vtp/tree/master/vtp2]
- A MapStore client working on an Electron [https://www.electronjs.org/] app to read and display the generated ZIP file
  - offline client (stand alone app for Windows) [http://demo.vtp2.geo-solutions.it/mapstore-electron-client/windows.zip]
  - repository [https://github.com/geosolutions-it/ogc-vector-tiles-vtp/tree/master/vtp2/electron]

Generate Tile Set Metadata

GeoSolutions developed a MapStore client where, once tiles and layers were loaded using the corresponding OGC APIs, they could be downloaded in a ZIP file together with its corresponding Tile Set Metadata. This client connected to the APIs provided by other participants in order to load the tiles the end user wanted. The end user would then choose which collection they want to show (Figure 13) on the client and then by clicking on a download button (Figure 14) they would trigger the creation of the ZIP file containing the tiles of the collection currently selected.

During the creation of the ZIP file, a metadata JSON file would be generated on the client side inside the compressed file; the content of this JSON file was consistent with the Tile Set Metadata Model and was generated based on a combination of rules as specified in Chapter 6 and user input. The end user was only requested to specify a Tile Set name before downloading.

The generated Tile Set Metadata contained also an additional property at root level named tiles. The tiles property provided the location and file extension of downloaded tiles using an array of string templates (e.g.: `tilesDirectoryName/{tileMatrix}_{tileRow}_{tileCol}.mvt`).

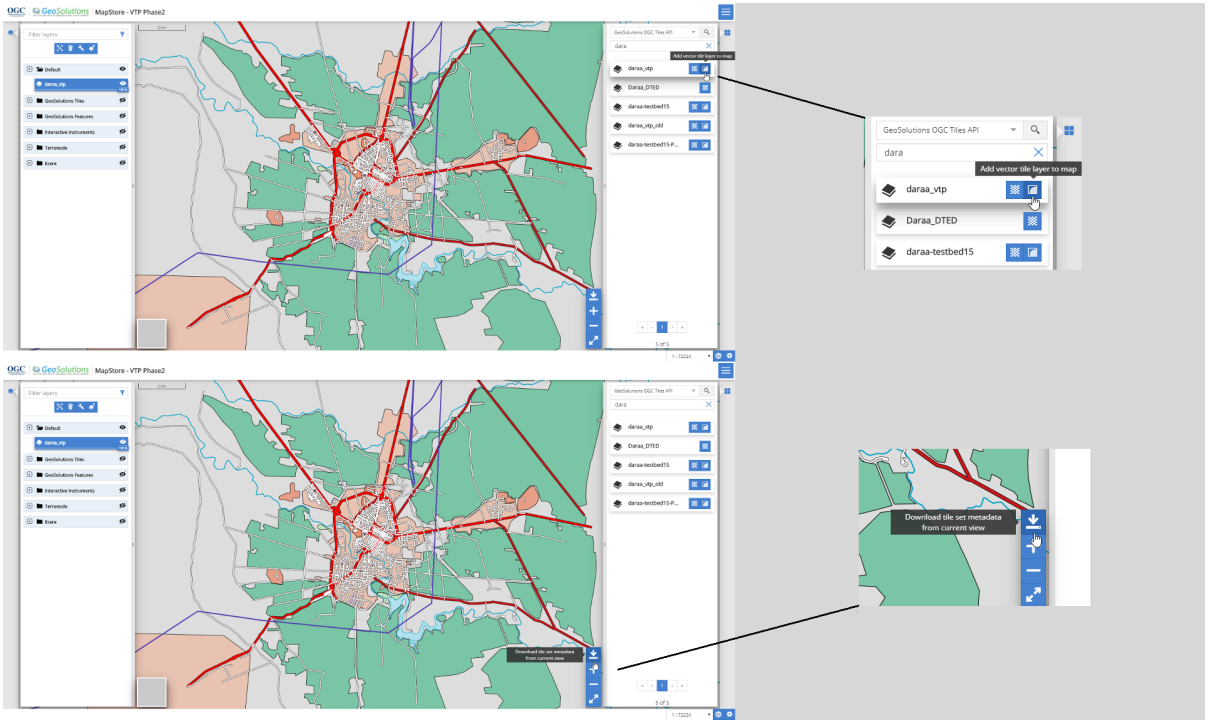

Figure 13. GeoSolutions - MapStore web client - steps to open tile set metadata download form: 1 add collection from an OGC Tiles API and 2 enable the download metadata plugin

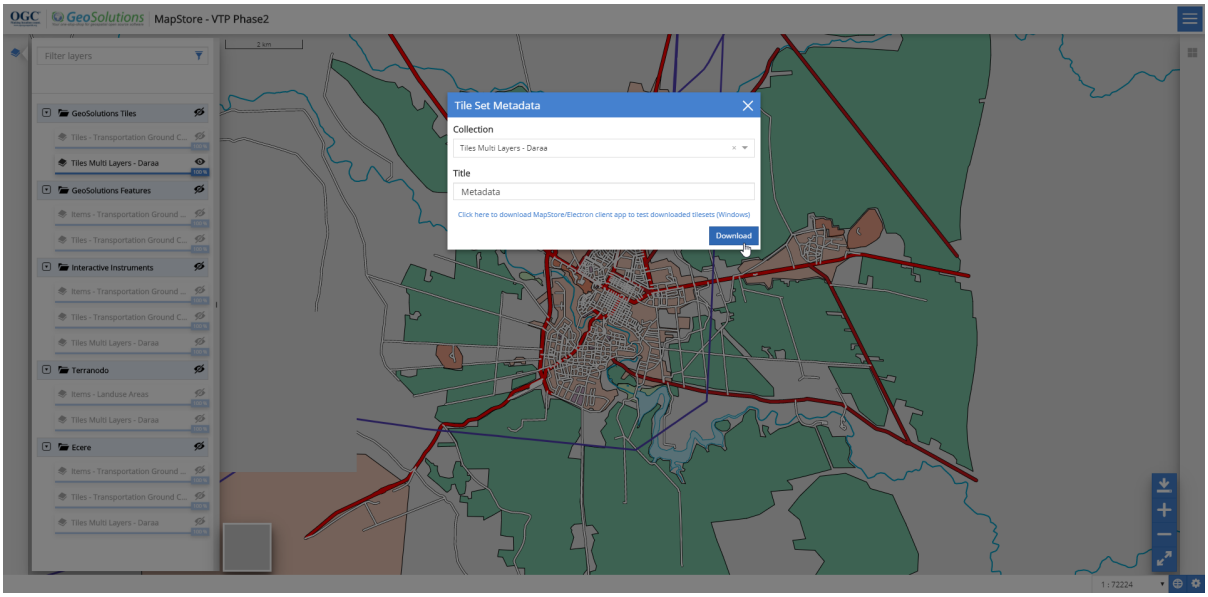

Figure 14. GeoSolutions - MapStore web client - display the tile set metadata download form

Structure of downloaded zip folder:

```
tileSetMetadata.zip
 ├──── tileSetMetadata.json
 ├──── downloadedTiles
 │     ├──── {tileMatrix}_{tileRow}_{tileCol}.mvt
 │     ├──── ...other downloaded tiles in MVT files
```

Read tile set metadata

A separate client was developed by GeoSolutions in order to work with the downloaded Tile Sets found in the ZIP files. The application layout was virtually identical to the application that created the ZIP files. This client had the capability of loading the ZIP files generated by the client application described on the previous paragraph and, upon the request of the end user, display the chosen Tile Set and the desired layer contained within. As shown on Figure 16, a button allowed the end user to visualize the Tile Set Metadata in a tree structure.

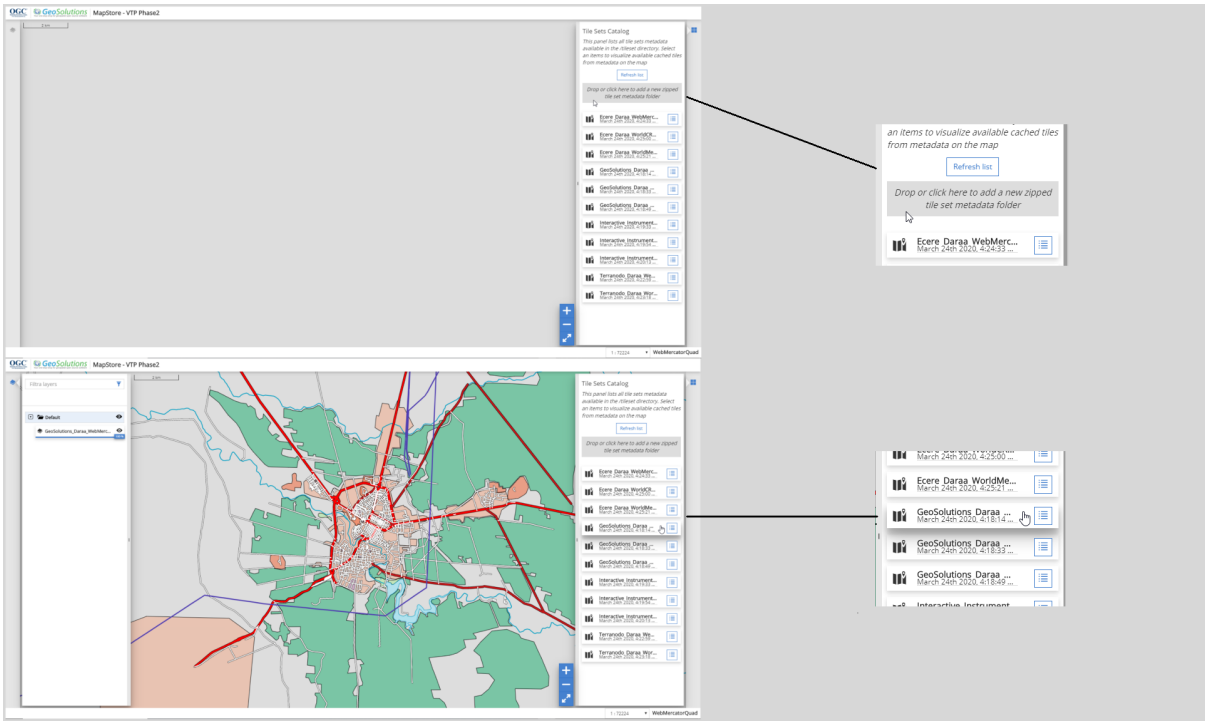

Figure 15. GeoSolutions - MapStore Electron client - steps to read and display tile set metadata zip file: 1 drag and drop zip folder on the grey field and 2 click on the item in the list representing the new tile set

Figure 16. GeoSolutions - MapStore Electron client - display tile set metadata json generated in the zip folder

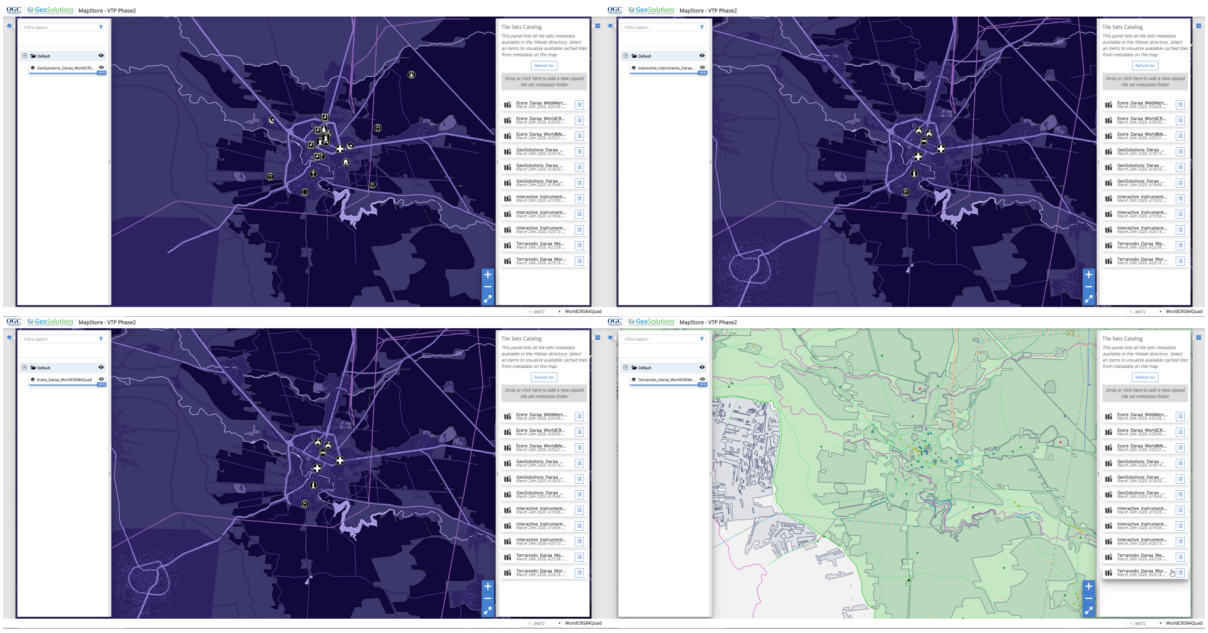

Figure 17. GeoSolutions - MapStore Electron client - display tile sets generated in WorldCRS84Quad from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

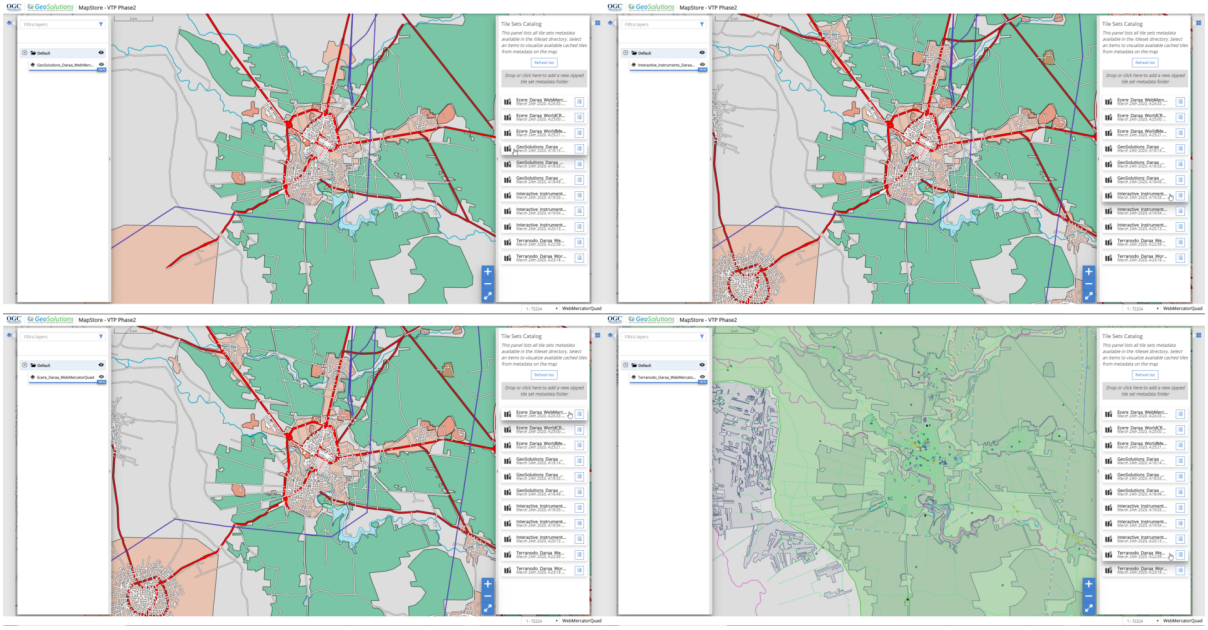

Figure 18. GeoSolutions - MapStore Electron client - display tile sets generated in WebMercatorQuad from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

NOTE	for this demo styles was located in a static folder and not downloaded from the servers

8.3. GeoPackage Tile Set Metadata

GeoPackages encode tile set metadata in relational tables defined in the standard as well as in the extensions for vector tiles.

This primarily consists of the *gpkgext_vt_layers* table, containing the list of all vector layers, as well as the *gpkgext_vt_fields* table, containing the list of attributes fields with names and types. The tile matrix sets information is stored in the *gpkg_tile_matrix_sets* and *gpkg_tile_matrix* tables. With the new TMS extension, the same tile matrix set description can be re-used for multiple layers, while also adding support for variable width, as required by the GNOSIS and CDB global grids. The spatial reference system information is stored in the *gpkg_contents* and *gpkg_spatial_ref_sys* tables.

8.3.1. Image Matters

Image Matters developed a GeoPackage Producer that implemented the proposed Tile Set Metadata Model in a GeoPackage. The GeoPackage producer populated the GeoPackage with the appropriate information.

8.3.2. Ecere

Ecere provided a GeoPackage producer which encodes tile set metadata in the GeoPackage, as described above.

The Ecere GeoPackage visualization tools make use of all this information to properly access and interpret the tiled vector data and associated attributes. The GeoPackages produced by both Ecere and Image Matters, in numerous combinations of different Tile Matrix Sets, with embedded

attributes or using attributes tables, and with individual layers or multiple layers per tiles, were all successfully tested.

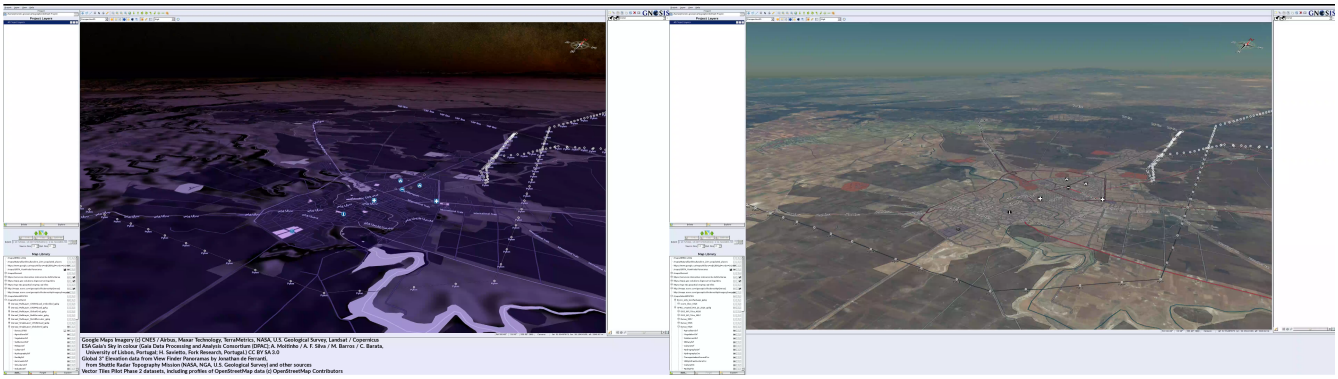

Figure 19. GeoPackages produced by Image Matters and Ecere using relational representation of tileset metadata displayed in Ecere's visualization client

Chapter 9. Discussion

The development of the Tile Set Metadata Model raised discussions regarding tile sets, tiling schemes, terms and definitions. This section summarizes those discussions.

9.1. Use of the Term 'Tile Cache'

The term "Tile Cache" has been widely considered a repository of pre-generated tile sets.

Even though one of the definitions of the word `cache` does imply a storage of elements, the term `tile cache` conflicts with the definition of what a `cache` means in computer science:

- *a hardware or software component that stores data so that future requests for that data can be served faster (source: Wikipedia).*

Caches are extensively used in software applications due to the performance benefits they offer. However, in terms of the phrase “tile cache” this can generate confusion as some software applications might actually store tiles in a cache in order to access those tiles more quickly. This instance creates a tile cache as defined in computer science terminology.

Consensus was reached in the Pilot on dropping the use of the term `Tile Cache` in favor of using `Tile Set`. For the reasons described in the Findings section of this report the adoption of this new term was not completely successful during the Pilot.

9.2. Location of Tile Set Metadata

Ideally, any data regarding geospatial elements should be accessible by any of the OGC APIs. As new elements are created to expand the functionalities of geographic information systems, new resources would be created on one or several APIs in order to access these new elements.

A considerable number of elements of data included in the Tile Set Metadata Model are already available via OGC APIs:

- The Tile Matrix Sets Bounding Box, Tile Matrices and Tile Matrix Limits are already available through several resources offered by the draft Tiles API.
- The Layers and the Feature Attributes found in them are also available through several resources of the OGC API - Features.

The remaining metadata not currently available in existing or draft OGC API specifications was provided by the main `TileSetMetadata` element described in the Metadata Model in Chapter 7. This was expected, as the `TileSetMetadata` is a novel element introduced during the course of this Pilot and the main objective of this particular ER.

In order to access the `TileSetMetadata` element, participants discussed where a new resource should be created and in which OGC API. Four proposals were discussed, having in common the use of `/tileSetMetadata` but differing in the API and path. The following alternatives were proposed, all were rejected for different reasons:

- Adding the resource at any level of the in-development OGC API - Records specification. This alternative was rejected due to the Records API being out of scope for this Pilot.
- Adding the resource at the root of either the Tiles API or Features API. This alternative was rejected due to concerns over compatibility with the design of the OGC APIs.
- Adding the resource at the `TileMatrixSet` level, i.e.,`/tileMatrixSet/{tileMatrixSetId}/metadata`. This alternative was rejected due to the fact the metadata belongs to a Tile Set and not a Tile Matrix Set. Nonetheless, this path ended up being used to access TileJSON Metadata.
- Adding the resource at the Collection level, i.e.,`/collections/{collectionId}/metadata`. This alternative was rejected due to the fact that one Tile Set can be composed of several Collections. This proposal evolved into the requirement of including Collection metadata as part of the Tile Set metadata.

NOTE	Styles API advertises the metadata of a style in a specific resource path that includes the Style identifier. This was not possible for Tile Sets because no API advertised Tile Set elements. Tile Sets are composed by several elements and not found on any OGC API by themselves.

The lack of feasible alternatives using OGC APIs led to the proposal of creating a stand-alone file containing the Tile Set Metadata. The alternative to this approach was to store only the TileSet metadata information that could not be accessed through an OGC API. The main benefit of this approach was to prevent storing duplicated metadata information. However, this approach was ultimately discarded in favor of storing all the metadata information in one place (even if most of it is duplicated) so as to keep all the metadata together and accessible through only one call to the file. After analyzing the pros and cons, consensus was reached on storing in a stand-alone file the entire extent of the Tile Set Metadata information defined in the Model developed in this Pilot.

9.3. Usage of TileJSON as a Source of Layer Metadata

Before this Pilot, the draft OGC API – Tiles specification did not support delivering multi-layer Mapbox Vector Tiles, nor did it have support for advertising the structure of the layers and their attributes. This issue was discussed at length during the course of the Pilot and the solution agreed to by the participants consisted of serving multi-layer MVTs out of a single collection in the Tiles API as well as having a single `describedby` link reporting the structure of the multi-layer vector tile, encoded as a TileJSON 3.0.0 document.

The apparent inability of the Tiles API (at the time) to support multi-layer MVTs posed a limitation for Tile Set Metadata (TSM) to support tile sets containing multi-layer MVTs, thus limiting the use cases where the TSM could actually be applied to. This limitation was addressed by the VTP2 initiative because the VTP2 solution enabled the Tiles API to support multi-layer MVTs and consequently also enabled Tile Set Metadata to support them.

The decision to use the TileJSON 3.0.0 draft in this Pilot was taken for the following reasons:

- TileJSON provided support for multi-layer MVTs.
- At the time the Pilot was executed, TileJSON was a widely-adopted specification for describing a tile set. The wide adoption was evidenced by implementations by Azavea [https://www.azavea.com/

blog/2019/03/04/introducing-tilejson-io/], Mapbox [https://docs.mapbox.com/help/glossary/tilejson/], OpenMapTiles [https://openmaptiles.org/docs/style/mapbox-gl-style-spec/] and others. Using TileJSON also has the benefit that the APIs developed in this Pilot could immediately be used out-of-the-box with existing tools such as visual style editors.

- Not all tools support tiling schemes beside WebMercatorQuad, i.e. APIs that want to be compatible with all the tools supporting TileJSON should also support tile sets using WebMercatorQuad.
- TileJSON 3.0.0 does not specify all of the tile set metadata defined in the Model but its content was sufficient for the purposes of this Pilot. Since additional elements may be added to a TileJSON document, additional information could be added in the future as needed.
- Even though the latest version of TileJSON was 2.2.0, this version did not support layer information, which was added in the 3.0.0 draft. Since 3.0.0 was compatible with 2.2.0 (as it only added additional JSON members) existing tools supporting 2.2.0 were still able to handle the TileJSON from the APIs.

9.4. Inclusion of Elements in the Metadata Model

9.4.1. Metadata Dates

At the beginning of the Pilot, the Tile Set Metadata Model was designed to include a similar class structure for metadata dates as its Styles Metadata Model counterpart as described in Figure 6.

The Styles Metadata Date design did not prevent adding multiple dates of the same type (e.g., adding multiple creation dates), something that must be avoided due to the incoherence it could produce. In order to prevent this from happening, the participants agreed on dropping the Styles Metadata Model Date design in favor of including all five dates in a single element named `MetadataDate`, specifying the date type in the actual name of each date attribute.

9.4.2. Lineage

The NSG Metadata Foundation (NMF) defines a set of classes to describe the change history of the element being described by the metadata, named Lineage, as shown on Figure 20.

After discussion, consensus was reached on not implementing the Lineage class structure in favor of a simplified change history registry. The ability to track the change history of a Tile Set was considered out of scope for this Pilot. Tile Set Metadata would only keep track of the creation and last modification date, as well as a point of contact.

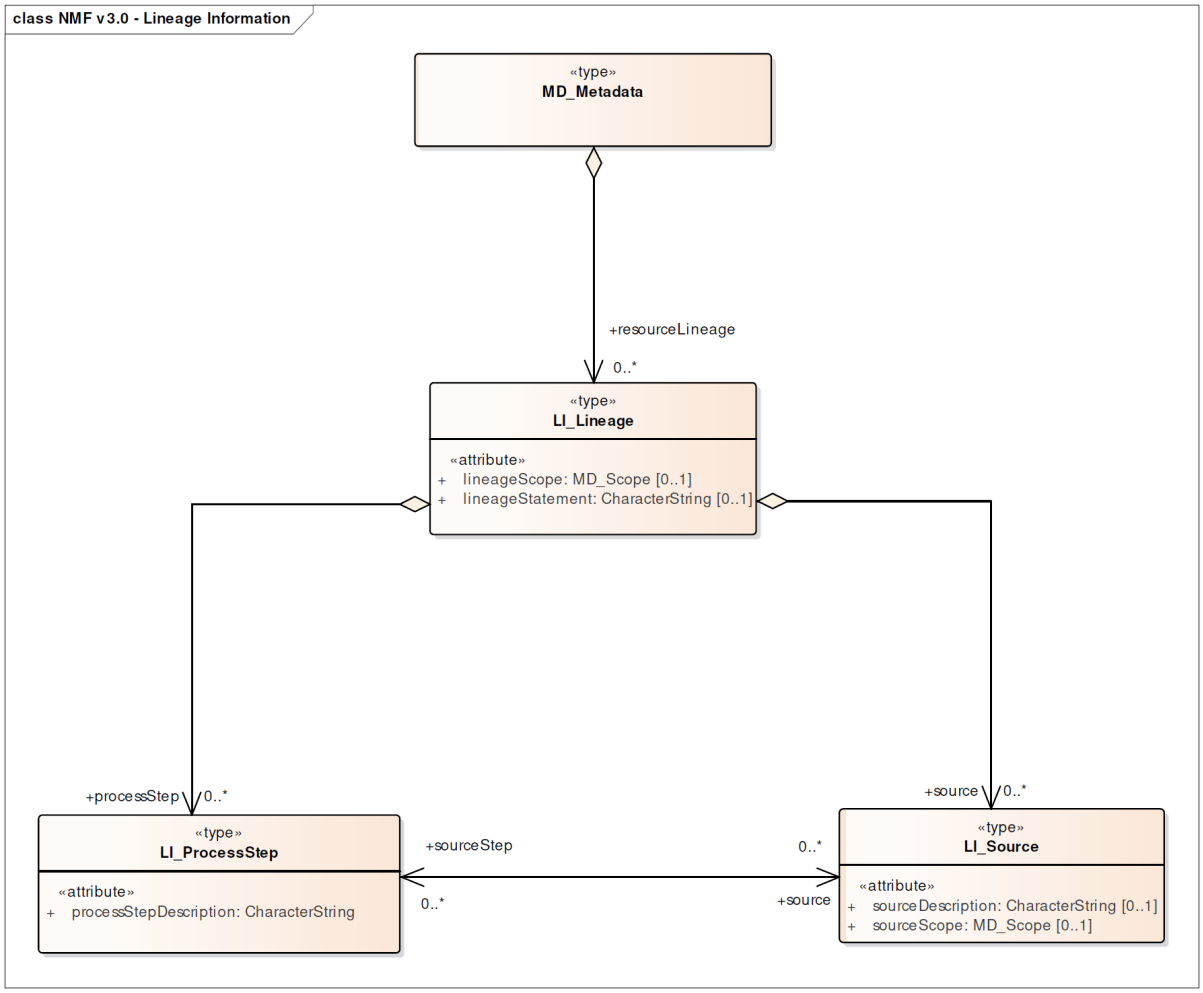

Figure 20. NMF Lineage Class Structure

9.4.3. Constraints

The Resource Constraints in the NMF, as shown on Figure 21, provide information on mandatory restrictions to the access and use of a resource or a set of resource metadata based on ISO 19115-1:2014.

After discussion, consensus was reached on not implementing the Resource Constraints class structure in favor of a simplified access restrictions approach. The majority of the classes within Resource Constraints are designed to provide support for Information Security Markings (ISM) Notices and Need-To-Know Metadata by following DoD/IC directives. These capabilities were considered outside of the scope for this Pilot. Tile Set Metadata would only keep track of a single identifier of the access restrictions of the Tile Set, providing the following options: confidential, restricted, secret, topSecret and unclassified.

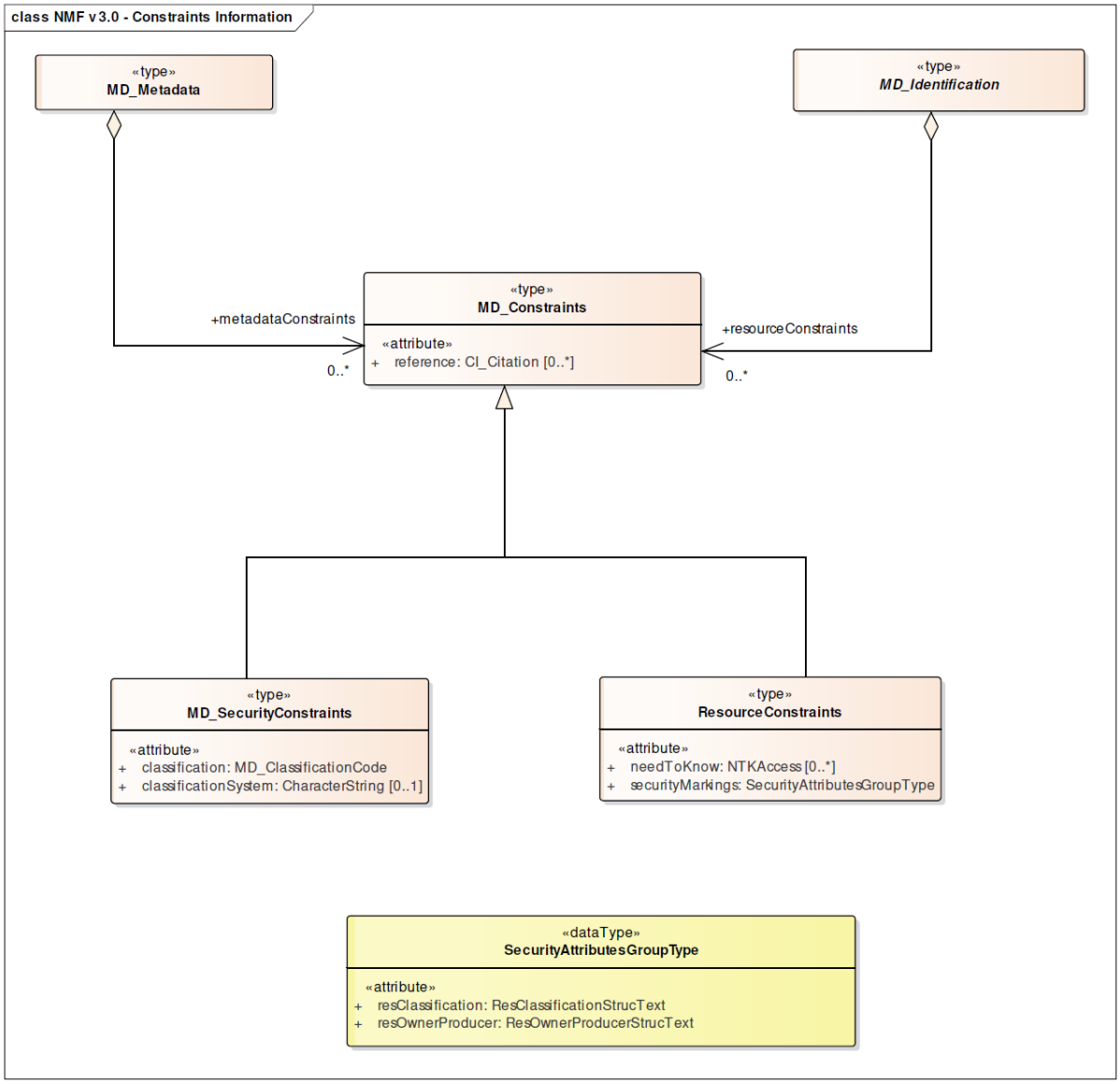

Figure 21. NMF Resource Constraints Class Structure

9.4.4. Bounding Box

The Tile Matrix Set Standard defines the Bounding Box as the *minimum bounding rectangle surrounding the tile matrix set, in the supported CRS* [12]. The Tile Matrix Set Standard has deprecated its use and stated the Bounding Box is informative and that it *should not be used to calculate the position of the tiles in the CRS space; please use topLeftCorner of the corresponding TileMatrix instead.* Nonetheless, per sponsor advise, the Bounding Box element of Tile Matrix Set was consequently included in the Tile Set Metadata Model.

In the Logical Model chapter of the draft OGC Tiling Conceptual Model and Logical Model for 2-D Planar Space Abstract Specification (OGC 19-014) the Bounding Box is referred to as a "tile set extent". This is a mandatory element. The requirement states, "Each tile set SHALL have an extent in the TileSet coordinate reference system (CRS). The extent SHALL be expressed following the rules as specified in Clause 10.2.1 Basic bounding box parameters (OWS Common)". At the time of publishing this VTP2 Tile Set Metadata Engineering Report, the OGC Two Dimensional Tile Matrix Set standard (OGC 17-083r2) is an approved standard, whereas the draft Tiling Abstract Specification has not yet been approved by the OGC Membership but has completed the Public Review [https://www.ogc.org/roadmap] stage. Future work on standardization of OGC API - Tiles should

therefore align with both the draft Tiling Abstract Specification and the OGC Two Dimensional Tile Matrix Set standard.

Chapter 10. Results

The work in this Pilot resulted in the following achievements:

- The participants agreed on a draft Tile Set Metadata Model.
- One implementation successfully demonstrated the use of the Tile Set Metadata Model in an offline environment.
- A GeoPackage implementation successfully implemented Tile Set Metadata, paving the way for offline implementations.
- All four tile servers successfully advertised TileJSON tiles metadata and all of them were consumed by at least one client.

Chapter 11. Findings

This section discusses the VTP-2 findings in relation to the metadata aspects of the Pilot. These findings, together with the lessons learned, will serve as a foundation for future work related to the development of OGC tile set metadata standards.

The draft OGC Tiling Conceptual Model and Logical Model for 2-D Planar Space Abstract Specification (OGC 19-014) identifies a generalization relationship between the Tile Set and Tile Set Metadata. The generalization relationship is shown in Figure 2. Indeed, in some applications the Tile Set Metadata may be serialized as a subclass of the Tile Set itself. However, other applications may wish to separate the metadata from the data described by the metadata. The VTP2 project modeled the relationship between a Tile Set and Tile Set Metadata as a bi-directional simple association, as shown in Figure 8.

In relation to TileJSON, the participants found that the path `⋯/tiles/{tileMatrixSetId}/metadata` used in the APIs implemented by the pilot does not have to be fixed in a standard. The use of "metadata" reflects that other representations of tile set metadata could also be supported in parallel, using the usual content negotiation mechanisms. Further, the latest published version of TileJSON should be used in an OGC API standard for informative purposes and extended as necessary. Having a registered media type for TileJSON is also important. Unfortunately, TileJSON does not appear to be currently under active development.

In the pilot, TileJSON was only used for tile sets using the Mapbox Vector Tile encoding. The link templates used in the pilot therefore do not have a parameter for the tile content encoding. For rendered tiles in image formats another parameter would be needed for the style used to render the tiles.

For GeoPackages, the latest draft extensions (Vector Tiles, Tile MatrixSet, Mapbox and GeoJSON Vector Tile (VT) Encoding, VT attributes table — https://gitlab.com/imagemattersllc/ogc-vtp2/-/tree/master/extensions) do a great job at providing all of the required information.

The Tile Set Metadata Conceptual Model and associated UML diagram featured in this ER summarize the possible tile set metadata properties very well. However, it does not have a direct representation in either the draft Tiles API or GeoPackage and that is perfectly fine and should not be necessary.

11.1. Terminology Conflicts

As discussed in the Use of the Term 'Tile Cache' section, the term "Tile Cache" has been widely considered a repository of pre-generated tile sets. After internal discussions, consensus was reached on dropping the use of the term "Tile Cache" in favor of using "Tile Set".

This recommendation was not fully implemented during the Pilot and frequently generated confusion. This confusion was mainly due to two factors:

- The extensive use of the term *Tile Cache* as a synonym of an offline repository of tile sets. This constantly made participants talk about offline tile set repositories using the terms *Tile Set*, *Tile Cache* as well as combinations of these terms with the words *offline* and *repository*.

- The lack of two different terms to differentiate the two types of tile sets that can actually be created:
  - A tile set could be created and visualized during the execution of a software application (i.e. on runtime).
  - A tile set could also be stored in an offline repository.

Chapter 12. Lessons Learned

The VTP2 participants learned the following in relation to the metadata aspects of the project.

12.1. Use of Queryables

Before this pilot project, it was not immediately clear what information was needed to populate the GeoPackage properly with tiled vector feature data. Before the decision was made to implement Tile Set metadata, the GeoPackage Producer software was designed to retrieve layer and field information from queryables. This proved to be an insufficient solution for a number of reasons, including the following:

- Queryables do not necessarily contain all of the fields in a Tile Set
- Queryables may contain the geometry, which is a special case in tiled feature data and not treated the same as fields
- Queryables support is an optional capability that was not consistently implemented by vendors

Based on this experience, The participants recommend that Tile Set metadata be included with any Tile Set containing tiled feature data.

12.2. Key Information for Clients

Clients need key information about tiles and their content to be able to use them properly. With the draft Tiles API, a lot of that information is available from the /tiles resource, which itself also references TileMatrixSets descriptions or URIs.

Additional information currently not specified by the Tiles API is needed to describe the fields and geometry type and in the case of multi-layer tiles the list of sub-layers and that same information for each sub-layer. One idea on how to do handle these capabilities is to define this information at the 'collection' level, as this information is the same whether clients/services are dealing with tiles or not. This approach would work especially well for multi-layer tiles if hierarchical collections are supported.

Towards the end of VTP2, participants agreed to encode this missing information by using the TileJSON format with minor tweaks (fields description being replaced by fields data types — however fields description might still be a desired optional capability). This has the benefits of working with tools intended to work with Mapbox technology.

However, the availability of that same information without relying on that TileJSON /tiles/metadata is still not standardized. For features, support for more complex schema is being proposed, but might not be essential, as a simpler solution similar to the queryables or TileJSON could potentially serve that purpose for most use cases. For sub-layers, hierarchical collections would be highly valuable.

Chapter 13. Conclusions

Tile sets are already a critical tool for accessing manageable sizes of tiled feature data. With its use increasing over time and its fundamental role when working offline, the metadata describing these tile sets will become equally critical.

The efforts carried out throughout the pilot revealed the importance of publishing tile set metadata in OGC APIs and demonstrated the potential for offline utilization of tile set metadata.

13.1. Future Work

The OGC Vector Tiles Pilot Phase 2 initiative explored in depth the metadata requirements for tile sets and specifically their requirements when transitioning from online to offline environments.

- Some metadata properties included in the conceptual model were not actually implemented nor analyzed in depth, leaving for future work to expand on several concepts.
- The creation of a Tile Set Metadata resource in the OGC API - Records specification that is under development should be studied in the future, as this alternative was not explored in this Pilot for being out of scope. This could be of particular importance for online and offline tile set repositories.
- VTP2 demonstrated a scenario whereby all of the tiles are of the same classification. This simplified the challenge of data sources from different security domains. Given that the classification of the tile set has to be equal to the classification of the tile with the highest classification, a single tile can make a complete tile set unusable within lower classification domains. Future work should therefore explore situations whereby security classifications are applied at a tile or feature level.
- Future projects might consider adding NMF elements into the Metadata Model, by looking closer at potential new use cases for tile sets that would actually require any of the NMF elements not included in this Pilot.

13.1.1. Standardization

This Pilot has demonstrated the need for tile set metadata to support interoperability between different tile-based geospatial applications, APIs and encodings. Future work may include advancing the results of Vector Tiles Pilot metadata activities into standardization processes.

The Tile Set Metadata Conceptual Model successfully meets the need for metadata to describe offline tile sets. However, the conceptual model includes elements which may not be needed in all market implementations (such as security metadata elements). For online APIs, testing indicated Mapbox TileJSON offers wide support in commercial and open source software, but may be viewed as being in the domain of a single vendor and may not be actively updated. Pilot results also indicated that Mapbox TileJSON can be effectively used to create Tile Set Metadata in working applications (as demonstrated by GeoSolutions). Accordingly, TileJSON and Tile Set Metadata do not compete, but complement each other.

Tile Set Metadata should therefore advance into standardization processes while recognizing many existing implementations may use TileJSON. In addition, OGC API - Tiles should adopt TileJSON as

an optional return format for API descriptions. This combined approach appears to offer a practical path forward for the open geospatial community.

13.1.2. Adding Styles to Downloaded Tile Sets

The direct encoding of the Tile Set Metadata Model in JSON, as implemented by the work of GeoSolutions, is a promising candidate for providing a proper comprehensive description of an offline tile set, outside of a Tiles API or GeoPackage context. Wider implementation of support for such a simple offline tile set (either as folders or a zipped file) could be the subject of further work focusing on generating and making use of them.

The GeoSolutions implementation, consisting of downloading a tile set and its corresponding metadata in a compressed ZIP file and then loading it in an offline environment, did not include downloading styles into the ZIP file. This implementation could be expanded by including in the same ZIP file a collection of styles stored in separate files and adapting the client to retrieve those styles in an offline environment. This addition would complete the extent of geospatial elements needed to work with maps based on tiled feature data in a completely offline environment.

Furthermore, the two software applications developed by GeoSolutions could be fused into one single software application capable of fetching tiled feature data from both online servers via OGC APIs, or downloaded ZIP files containing tiles, styles and metadata. If this application would automatically generate and download in ZIP files all the tiles, layers and styles that were downloaded online, the same application would be able to automatically fetch those ZIP files in case the user lost connectivity, allowing for a seamless transition between an online and offline environment.

13.1.3. Refined Terminology

Future projects should resolve the terminology conflicts outlined in this ER. Separate terms should be defined for these three different concepts:

- Repositories of tile sets, stored online (i.e.,S3 Buckets)
- Repositories of tile sets, stored offline (i.e.,file system or ZIP file)
- Tile sets generated and utilized on runtime (i.e.,a tile set retrieved from an API and immediately used on a mapping application)
- Actual caches (i.e.,data storage meant for fast access at runtime) of tile sets

Annex A: Tile Set Metadata JSON Encoding

The following is an example JSON document implementing the tile set metadata.

Tile Set Metadata example


```
{
    "title": "Daraa",
    "abstract": "This is a test dataset for the Open Portrayal Framework thread in the
OGC Testbed-15 as well as for the OGC Vector Tiles Pilot Phase 2. The data is
OpenStreetMap data from the region of Daraa, Syria, converted to the Topographic Data
Store schema of NGA.",
    "keywords": null,
    "pointOfContact": null,
    "version": null,
    "scope": "Tile Set",
    "accessConstraints": null,
    "dates": {
        "creation": 1584351161456,
        "publication": null,
        "revision": null,
        "validTill": null,
        "receivedOn": null
    },
    "layers": [
        {
            "id": "AgriculturePnt",
            "title": "AgriculturePnt",
            "description": "Agricultural: Information about activities and man-made
features involved in the raising of crops and animals, for food and non-food
purposes.",
            "featureAttributes": [
                {
                    "id": "id",
                    "stype": "string (0..1)"
                },
                {
                    "id": "F_CODE",
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI001_SDV",
                    "stype": "dateTime (0..1)"
                },
                {
                    "id": "UFI",
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI005_FNA",
```

```
                    "stype": "string (0..1)"
                },
                {
                    "id": "FCSUBTYPE",
                    "stype": "integer (0..1)"
                },
                {
                    "id": "ZI006_MEM",
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI001_SDP",
                    "stype": "string (0..1)"
                }
            ]
        }, ...  {
            "id": "VegetationSrf",
            "title": "VegetationSrf",
            "description": "Vegetation: Information about the plant life in an area,
or the lack thereof.",
            "featureAttributes": [
                {
                    "id": "id",
                    "stype": "string (0..1)"
                },
                {
                    "id": "F_CODE",
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI001_SDV",
                    "stype": "dateTime (0..1)"
                },
                {
                    "id": "UFI",
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI005_FNA",
                    "stype": "string (0..1)"
                },
                {
                    "id": "FCSUBTYPE",
                    "stype": "integer (0..1)"
                },
                {
                    "id": "ZI024_HYP",
                    "stype": "integer (0..1)"
                },
                {
                    "id": "ZI006_MEM",
```

```
                    "stype": "string (0..1)"
                },
                {
                    "id": "ZI001_SDP",
                    "stype": "string (0..1)"
                },
                {
                    "id": "VEG",
                    "stype": "integer (0..1)"
                }
            ]
        }
    ],
    "tileMatrixSetLink": {
        "tileMatrixSet": {
            "id": "WebMercatorQuad",
            "title": "This tile matrix set is the most used tile matrix set in the
mass market: for example, by Google Maps, Microsoft Bing Maps and Open Street Map
tiles. Nevertheless, it has been long criticized because it is a based on a spherical
Mercator instead of an ellipsoid. The use of WebMercatorQuad should be limited to
visualization. Any additional use (including distance measurements, routing etc.)
needs to use the Mercator spherical expressions to transform the coordinate to an
appropriate CRS first. (from D.1 http://docs.opengeospatial.org/is/17-083r2/17-
083r2.html)",
            "abstract": null,
            "keywords": null,
            "supportedCRS": "http://www.opengis.net/def/crs/EPSG/0/3857",
            "tileMatrix": [
                {
                    "id": "0",
                    "title": null,
                    "abstract": null,
                    "keywords": null,
                    "scaleDenominator": 559082264.0287178,
                    "topLeftCorner": [
                        -20037508.3427892,
                        20037508
                    ],
                    "tileWidth": 256,
                    "tileHeight": 256,
                    "matrixHeight": 1,
                    "matrixWidth": 1
                }, ... {
                    "id": "24",
                    "title": null,
                    "abstract": null,
                    "keywords": null,
                    "scaleDenominator": 33.3238997476528,
                    "topLeftCorner": [
                        -20037508.3427892,
                        20037508
```

```
                    ],
                    "tileWidth": 256,
                    "tileHeight": 256,
                    "matrixHeight": 16777216,
                    "matrixWidth": 16777216
                }
            ]
        },
        "tileMatrixSetLimits": {
            "tileMatrixLimits": [
                {
                    "minTileRow": 3308,
                    "maxTileRow": 3311,
                    "minTileCol": 4915,
                    "maxTileCol": 4919,
                    "tileMatrix": "13"
                }
            ]
        }
    },
    "tiles": [
        "11a47900-6769-11ea-9c79-e10ad1351a38/{tileMatrix}_{tileRow}_{tileCol}.mvt"
    ]
}
```

Annex B: Revision History

Table 16. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
January 21, 2020	S. Taleisnik	.1	all	initial outline
March 23, 2020	S. Taleisnik	.2	all	preliminary version
March 31, 2020	S. Taleisnik	.3	all	preliminary version
April 27, 2020	S. Taleisnik	.4	all	final version

Annex C: Bibliography

[1] Aime, A., Bovolo, S.: OGC Vector Tiles Pilot 2: Vector Tiles Filtering Language Engineering Report. OGC 19-084,Open Geospatial Consortium, https://www.opengeospatial.org/docs/er (2020).

[2] Geospatial Intelligence, N.S. for: National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF) Version 3.0, (2016).

[3] OGC Testbed-15: Styles API Engineering Report, (2019).

[4] Aime, A.: OGC Testbed-15: Encoding and Metadata Conceptual Model for Styles Engineering Report. 19-023r1,Open Geospatial Consortium, http://docs.opengeospatial.org/per/19-023r1.html (2019).

[5] Mapbox: TileJSON 3.0.0 Daft Specification, (2018).

[6] Meek, S.: OGC Vector Tiles Pilot: Summary Engineering Report. OGC 18-086,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-086r1.html (2019).

[7] Ingensand, J., Maia, K.: OGC Vector Tiles Pilot: Tiled Feature Data Conceptual Model Engineering Report. OGC 18-076,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-076.html (2019).

[8] Yutzler, J.: OGC Vector Tiles Pilot: GeoPackage 1.2 Vector Tiles Extensions Engineering Report. OGC 18-074,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-074.html (2019).

[9] Vretanos, P.A.: OGC Vector Tiles Pilot: WMTS Vector Tiles Extension Engineering Report. OGC 18-083,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-083.html (2019).

[10] Vretanos, P.A.: OGC Vector Tiles Pilot: WFS 3.0 Vector Tiles Extension Engineering Report. OGC 18-078,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-078.html (2019).

[11] Yutzler, J.: Vector Tiles Pilot Extension Engineering Report. OGC 18-101,Open Geospatial Consortium, http://docs.opengeospatial.org/per/18-101.html (2019).

[12] Joan Masó, J.Y., Rushforth, P.: OGC Two Dimensional Tile Matrix Set, (2019).

[13] Stephane Fellah, S.G., Emily Mitchell: OGC Testbed-13: Portrayal Engineering Report, (2018).