DRAFT

OGC Vector Tiles Pilot 2

Tile Set Metadata Engineering Report

Table of Contents

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: OGC 19-082

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

LICENSE AGREEMENT

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

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

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

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

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

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

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

This Vector Tiles Pilot 2 (VTP-2) Engineering Report (ER) describes a conceptual model for Tile Set Metadata that provide information about the intended usage of the Tile Set as well as the origin, security level, tiling scheme, layers and feature properties contained within.

The metadata is intended to facilitate Tile Set search and describe the major characteristics of Tile Sets without actually accessing its tiles or features, a process that would be time consuming due to the large amount of tiles stored inside a Tile Set.

Additionally, this document summarizes the discussions about Tile Set Metadata among the participants of VTP-2, and draws up conclusions and recommendations for future work on the subject.

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

Chapter 2. Executive Summary

Tile Sets, sometimes called Tile Caches, are a well-established method for storing and accessing manageable packages of tiled feature data both offline and online. As Tile Set use increases and the need of seamless transitions between online and offline environments become greater, the need for an efficient access to the metadata of Tile Sets becomes more apparent.

The objective of VTP-2 is to deliver a consistent, interoperable online/offline architecture for vector tiles based on feature and tile servers and GeoPackage.

Regarding metadata, the activities of VTP-2 can be grouped into three major components:

- Building a Tile Set metadata model.
- Capturing and summarizing discussions and recommendations.
- Describing TIEs implementing Tile Set metadata.

The Tile Set Metadata Model built in VTP-2 responded to a series of requirements established in the Call For Participation (CFP) stage and further refined through the course of the Pilot. The initial metadata requirements in the CFP consisted in:

- Develop a metadata model for vector tiles and stored tile caches.
- The National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF) shall be extended.
- Metadata shall describe fundamental aspects of the tiles, such as date, creator, source, etc.
- Metadata elements shall 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).
- Style names or identifiers and metadata to describe multi-layer vector tiles.
- Other elements required to support the Vector Tiles Filter language described below.
- 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.

Since the CFP was defined a year before its release, many requirements evolved in order to cope with the developments carried out ever since. Additionally, conversations throughout the Pilot further refined these requirements. The need to extend the NMF was changed to limiting the NMF elements implemented to only those considered of value to the TileSet use cases. The metadata model was oriented exclusively to TileSets and not on both TileSets and the Vector Tiles.

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

- Describing fundamental aspects of the TileSet such as names, identifiers, dates, owners, origin, security levels, etc.
- Describing the tiling scheme by storing Tile Matrix Set information of the TileSet. This helps identify the geographic extent, the detail level 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 TileSet Metadata was implemented. The creation of an additional resource to one of the current APIs of OGC was discarded for many technical reasons and consensus was reached on storing all the TileSet Metadata in a stand-alone file that would be located in the root folder of the offline repository.

2.1. Business Value

The main benefits for 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. On some cases, maps represented with vector tiles can be 20 to 30 times smaller than the same maps represented by raster tiles. This reduction enables the possibility of storing large sets of maps (i.e. tile sets) into secure and lightweight removable media devices. These tile set repositories are sometimes called by the sponsor "tile caches".

As described in Figure 1, a tile set repository (such as GeoPackage, Static Cache, or Compact Cache V2) is being used by a humanitarian relief convoy in the middle of the desert and with limited to no connectivity, supported by a group of interconnected systems working and communicating with each other. The repositories are generated at 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 throughout OGC API Styles, API Tiles, API Images and API Routes.

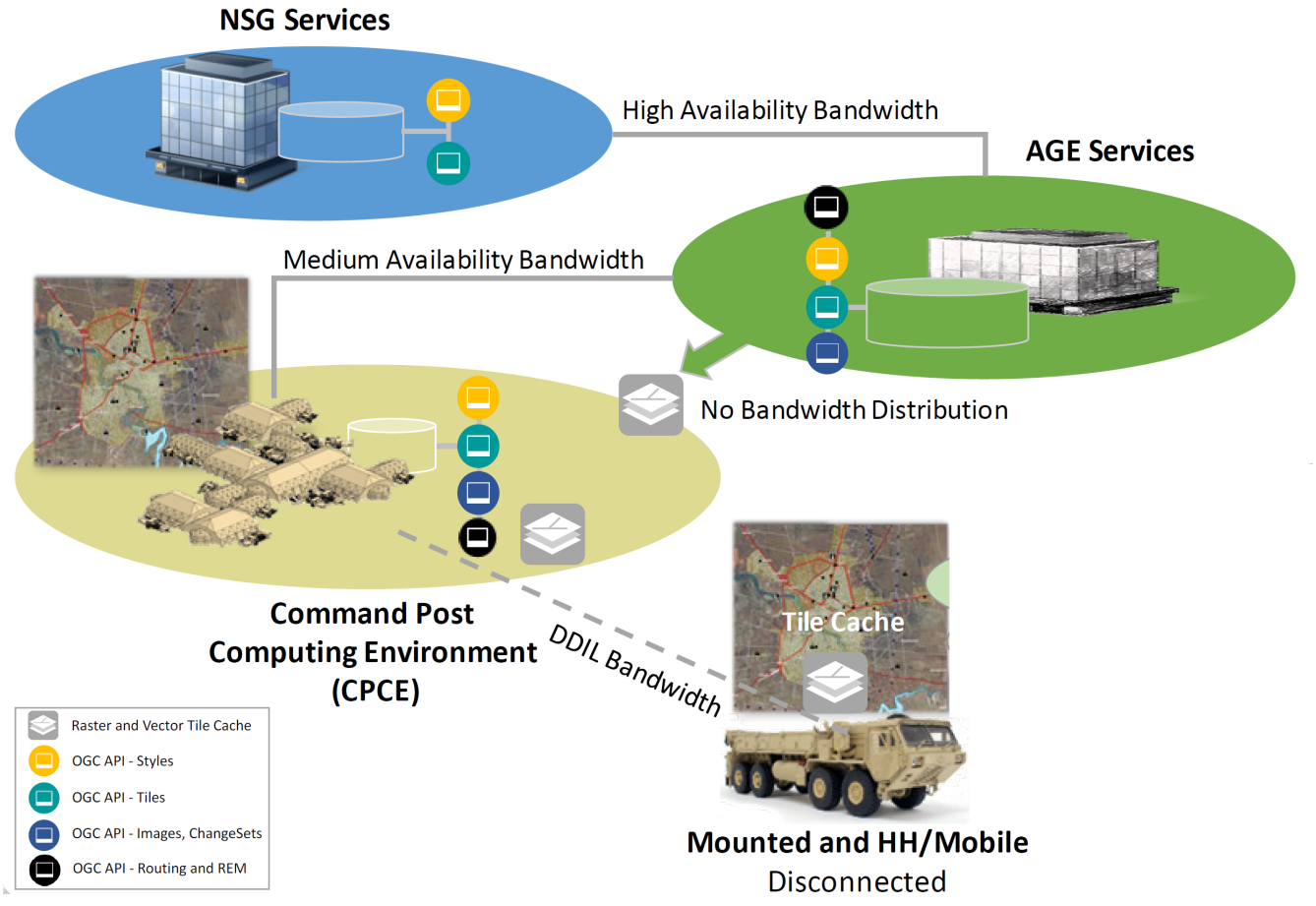

Figure 1. Sponsor Modular Scenario

The intention with working with vector tile sets is to transition from using old and outdated raster maps into using up-to-date and small-sized vector maps which allow for different styles to be applied as needed. An end user would easily transition from a day topographic styling during daytime operations, to night topographic during night time, or even a satellite-overlay styling. Vector tile sets also allow for routing engines to calculate routes and displaying them on top of the map, although 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 sponsor of this Pilot required accessing these vector tile sets in a complete 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 throughout the entire System of Geospatial Intelligence. Tiles are served between NSG Services (National Level), AGE Services (Enterprise Level) and CPCEs (Command Post), throughout Tiles API. The inclusion of metadata within those tile sets allow users among all levels to quickly access the fundamental pieces of information related to those tile sets and act accordingly by creating repositories, 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 the information they need to describe tile sets to their complete extent, facilitating the development of new applications utilizing this technology. The use of open standards would allow for interoperability with other systems, expanding the potential use for vector tile sets.

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

Name	Organization	Role
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 18-076, OGC® Vector Tiles Pilot: Tiled Feature Data Conceptual Model Engineering Report [http://www.opengis.net/doc/PER/vtp-conceptualModel] [http://www.opengis.net/doc/PER/vtp-conceptualModel]
- OGC: OGC 17-083r2, OGC® Two Dimensional Tile Matrix Set [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html] [http://docs.opengeospatial.org/is/17-083r2/17-083r2.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] [http://docs.opengeospatial.org/per/19-023r1.pdfhttp://docs.opengeospatial.org/per/19-023r1.pdf]
- OGC: OGC® Tiling Conceptual Model and Logical Model for 2-D Planar Space draft Abstract Specification [https://portal.opengeospatial.org/files/?artifact_id=83973&version=1] [https://portal.opengeospatial.org/files/?artifact_id=83973&version=1]
- National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF) [https://portal.opengeospatial.org/files/?artifact_id=83568] [https://portal.opengeospatial.org/files/?artifact_id=83568]
- OGC: OGC® API-Tiles [https://github.com/opengeospatial/OGC-API-Tiles/blob/master/standard/OAPI_Tiles.pdf] [https://github.com/opengeospatial/OGC-API-Tiles/blob/master/standard/OAPI_Tiles.pdf]
- OGC: OGC 19-010, OGC® Testbed-15: Styles API Engineering Report [https://docs.opengeospatial.org/DRAFTS/19-010.html] [https://docs.opengeospatial.org/DRAFTS/19-010.html]
- TileJSON 3.0.0 Daft Specification [https://github.com/mapbox/TILEjson-spec/tree/3.0/3.0.0] [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**

  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 as a repository containing a pre-generated tile set. Its use was suggested to be dropped during the Pilot.

- **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**

  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 DGGS and other organizations including irregular ones.

- **tile set**

  a series of actual tiles contain data and following a common tiling scheme.

- **tile set metadata**

  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

NOTE: The abbreviated terms clause gives a list of the abbreviated terms and the symbols necessary for understanding this document. All symbols should be listed in alphabetical order. Some more frequently used abbreviated terms are provided below as examples.

- CRS Coordinate Reference System
- JSON JavaScript Object Notation
- NMF NSG Metadata Foundation
- NSG National System for Geospatial Intelligence
- OGC Open Geospatial Consortium
- TMS Tile Matrix Set
- TSM Tile Set Metadata

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 OGC Tiling Conceptual Model and Logical Model for 2-D Planar Space provided a foundation for the Tile Set Metadata Model. It described a general tiling conceptual model and defined a logical model for tiling 2D Euclidean Space.

The Tiling Conceptual Model, as showcased in Figure 2, describes the relationship between Tiles, Tile Sets, Tile Scheme and Tile Set Metadata. This model was replicated in the Tile Set Metadata Model, although the `Tile` element was eventually discarded from the model since including it would mean including in the metadata all the tiles contained in the tile set.

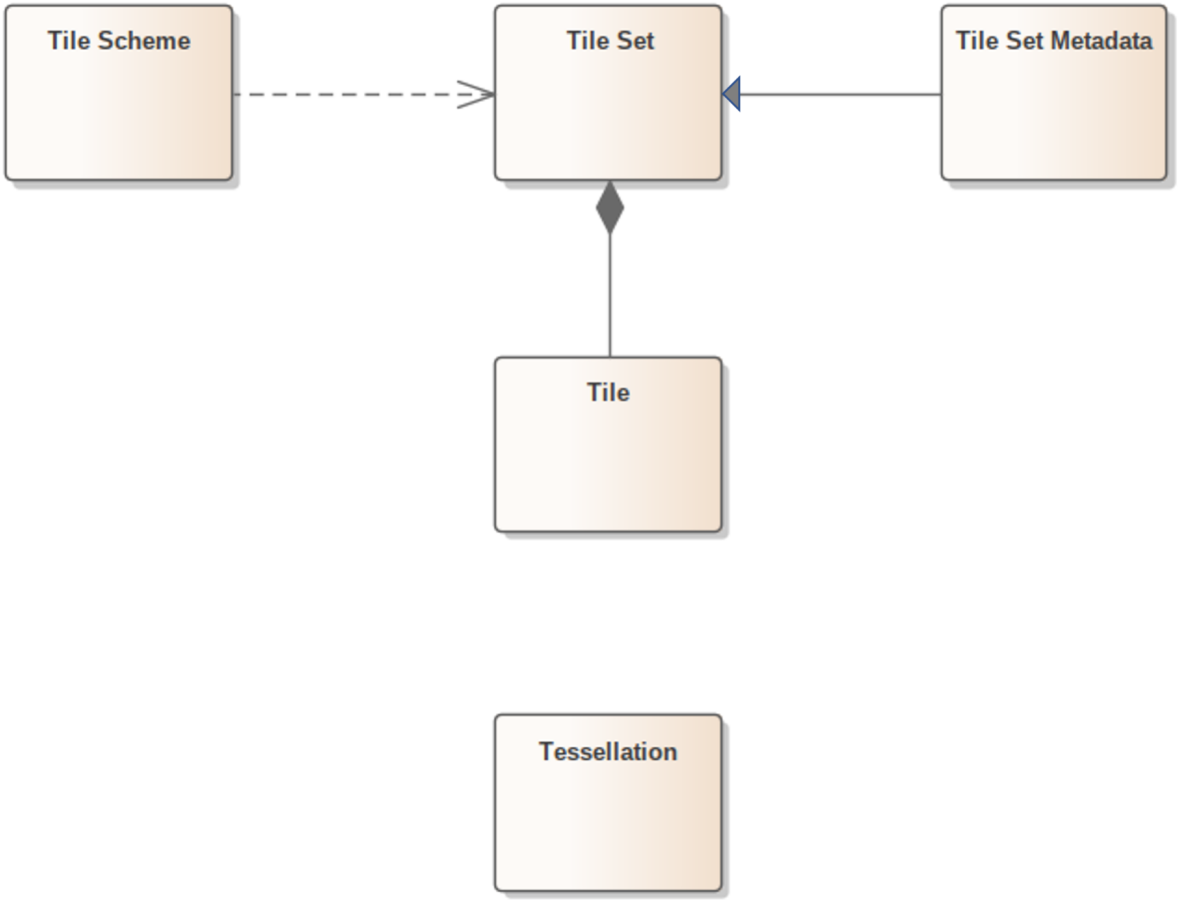

Figure 2. Conceptual for space partitioning based on tiles

6.2. Tile Matrix Set Standard

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

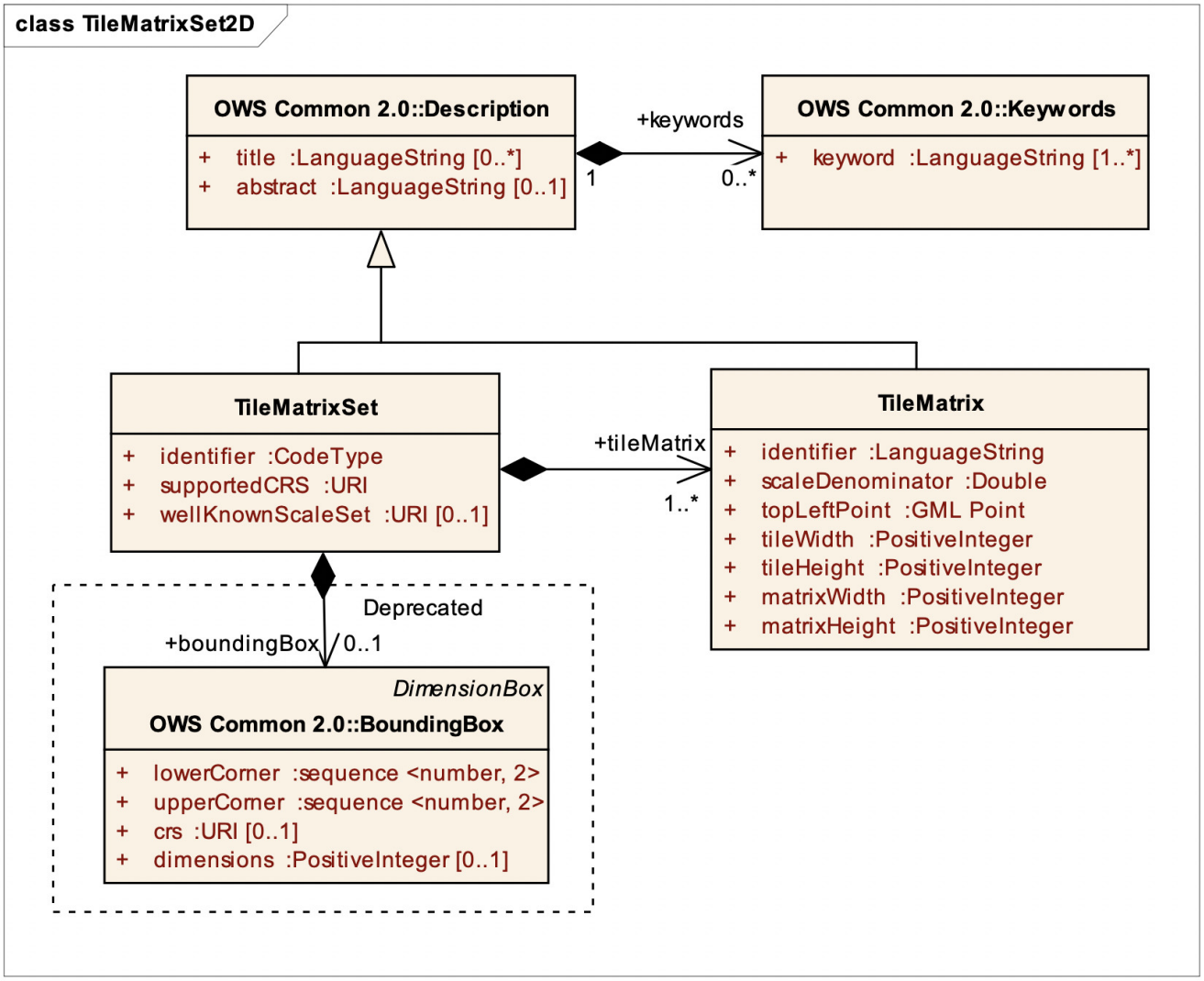

Figure 3. TileMatrixSet UML Model

The 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. It's main elements, as shown in the TileMatrixSet UML Model in Figure 3, were all included in the TSM 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.

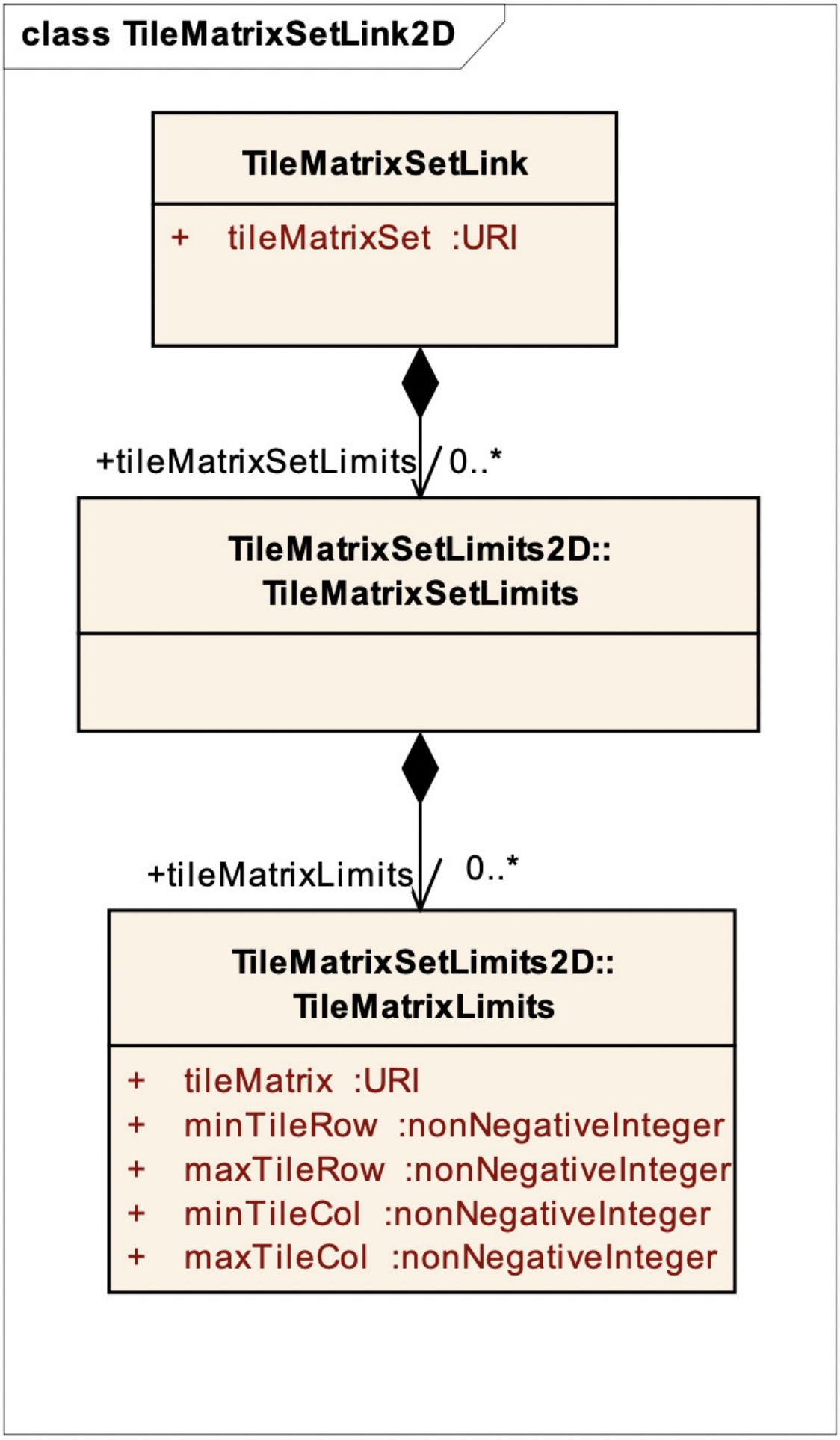

Figure 4. TileMatrixSetLink UML Model

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. cite:[NSG2016]

The Call For Proposal of VTP-2 required extending the NSG Metadata Foundation into the Tile Set Metadata. Over the course of the Pilot, some elements were implemented while some others were disregarded in favor of simpler data structures and left for future innovation initiatives to further examine their potential benefits.

As shown on Figure 5, the NMF is mainly composed by one central metadata element containing a small number of metadata attributes and binding together the remainder 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 DoD/IC directives for the use of IC.ADD.V2 (Chapter 3, Information Security Marking) and extends these elements. Provides 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 by 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.

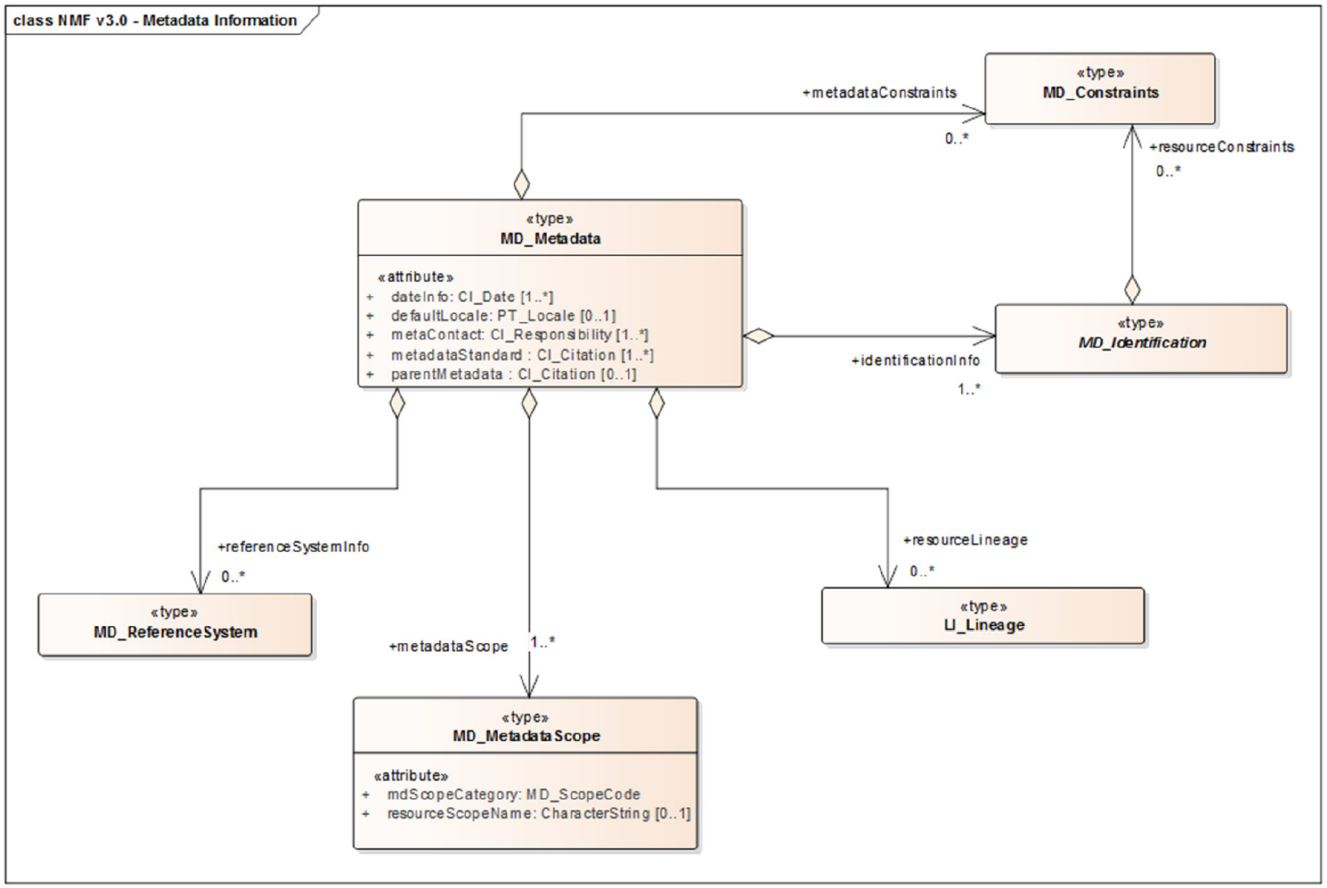

Figure 5. NMF Metadata Information Model

6.4. Testbed 15 Styles API and Styles Metadata ER

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

6.4.1. Styles API

The Styles API ER of Testbed 15 described a draft specification that defined five requirements/conformance classes for Styles API and two requirements/conformance classes extending the information about Collection resources specified in OGC API - Features - Part 1: Core.

Metadata API resource

The Engineering Report specified the Styles API as designed and implemented in the Open Portrayal Framework task of OGC Testbed 15. One of the resources described as part of Styles API was the Styles Metadata, which was considered when defining an API and a resource path to access Tile Set Metadata.

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 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 support clients like visual style editors to construct expressions for selection criteria in queries on features in the collection. cite:[Portele2019]

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, in a CQL query expression or as part of a selection criteria in a 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, as the `queryable` properties have the power to summarize the type of information being stored on a particular Collection. An end user could find this information useful when being offline and deciding which Tile Set to choose out of a group. Eventually, it was found to have multiple flaws that discarded it as part of the metadata model, although from a conceptual point of view it prolvided the foundation for including feature attribute types in the metadata model. This resource was eventually discarded because it was found that not all elements found in `Queryables` were actually attributes that could be found and used.

6.4.2. Styles Metadata

The Encoding and Metadata Conceptual Model for Styles ER of Testbed 15 described a conceptual model for style encoding and metadata that provided information for understanding styles intended usage, availability, compatibility with existing layers, as well as supporting style search. Being a recent OGC ER and the only OGC ER focused on building a Metadata Model, many elements of the Styles Metadata Model (as shown on Figure 6) inspired the design of the Tile Set Metadata Model.

The central Metadata element, the metadata date and date type and the access constraints elements, were all major components of the early design of the Tile Set Metadata Model.

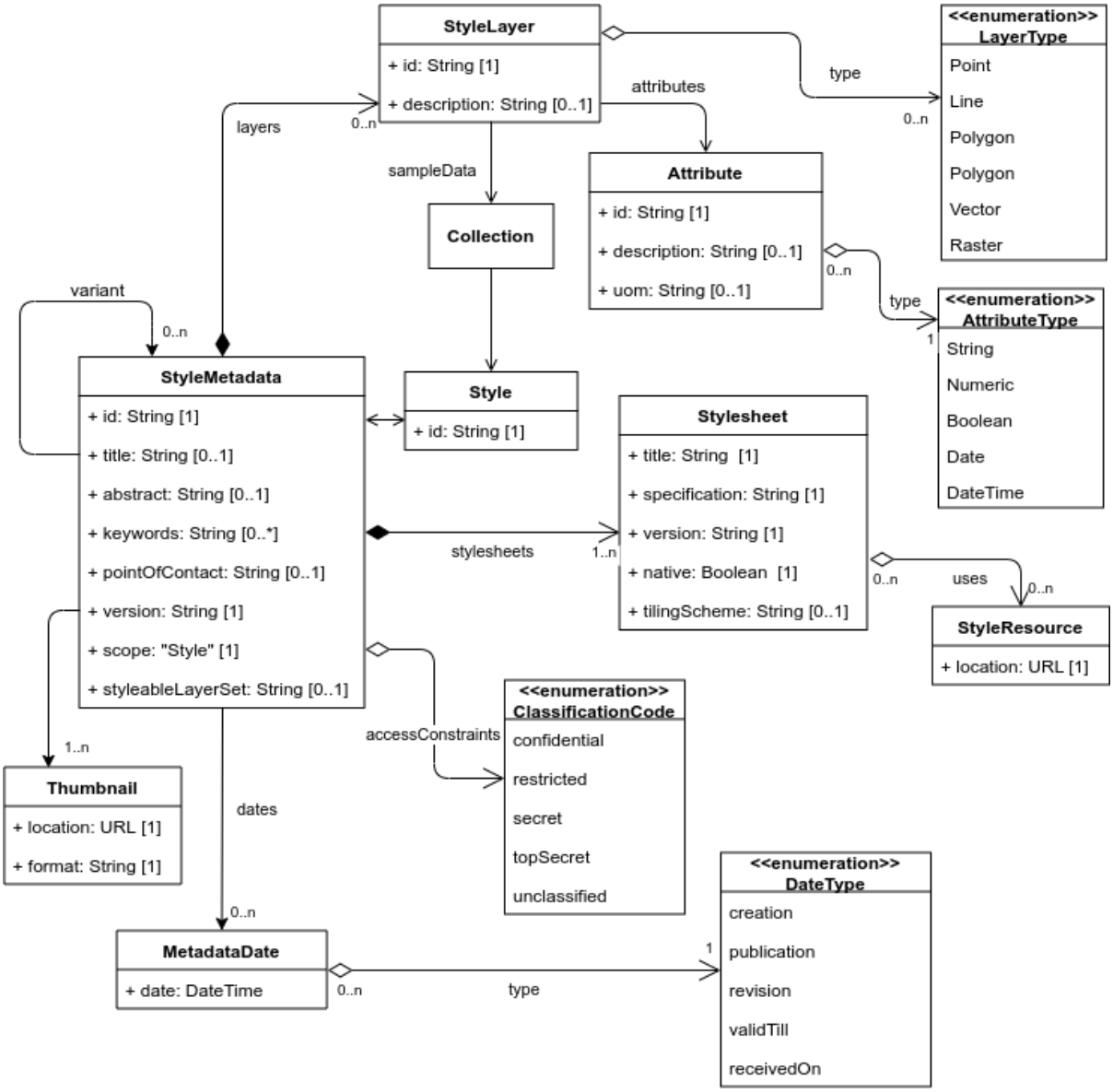

Figure 6. Style metadata and encoding conceptual model

6.5. TileJSON 3.3.0

TileJSON is self-defined as a specification that *attempts to create a standard for representing metadata about multiple types of web-based map layers, to aid clients in configuration and browsing.* cite:[Mapbox2018]

Developed by Mapbox, version 3.3.0 was still a draft during the course of this Pilot but was chosen by participants before the last version (2.2.0) because the latter did not support layer information and 3.0.0 was compatible with 2.2.0. TileJSON V3 provides support for multi-layer tiles and was at the time of the Pilot a de-facto standard for describing a tile set.

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.

TileJSON became a critical component of the Tile Set Metadata Model when it was chosen as the solution for Tiles API to support multi-layer Mapbox Vector Tiles.

6.6. VTP1 Tiled Feature Data Conceptual Model

As part of the Vector Tiles Pilot 1, the ER created to provide a specification for a conceptual model for vector tiles was the Tiled Feature Data Conceptual Model (TFD CM).

The contribution of the TFD CM in the Tile Set Metadata Model was ultimately reduced, though 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.

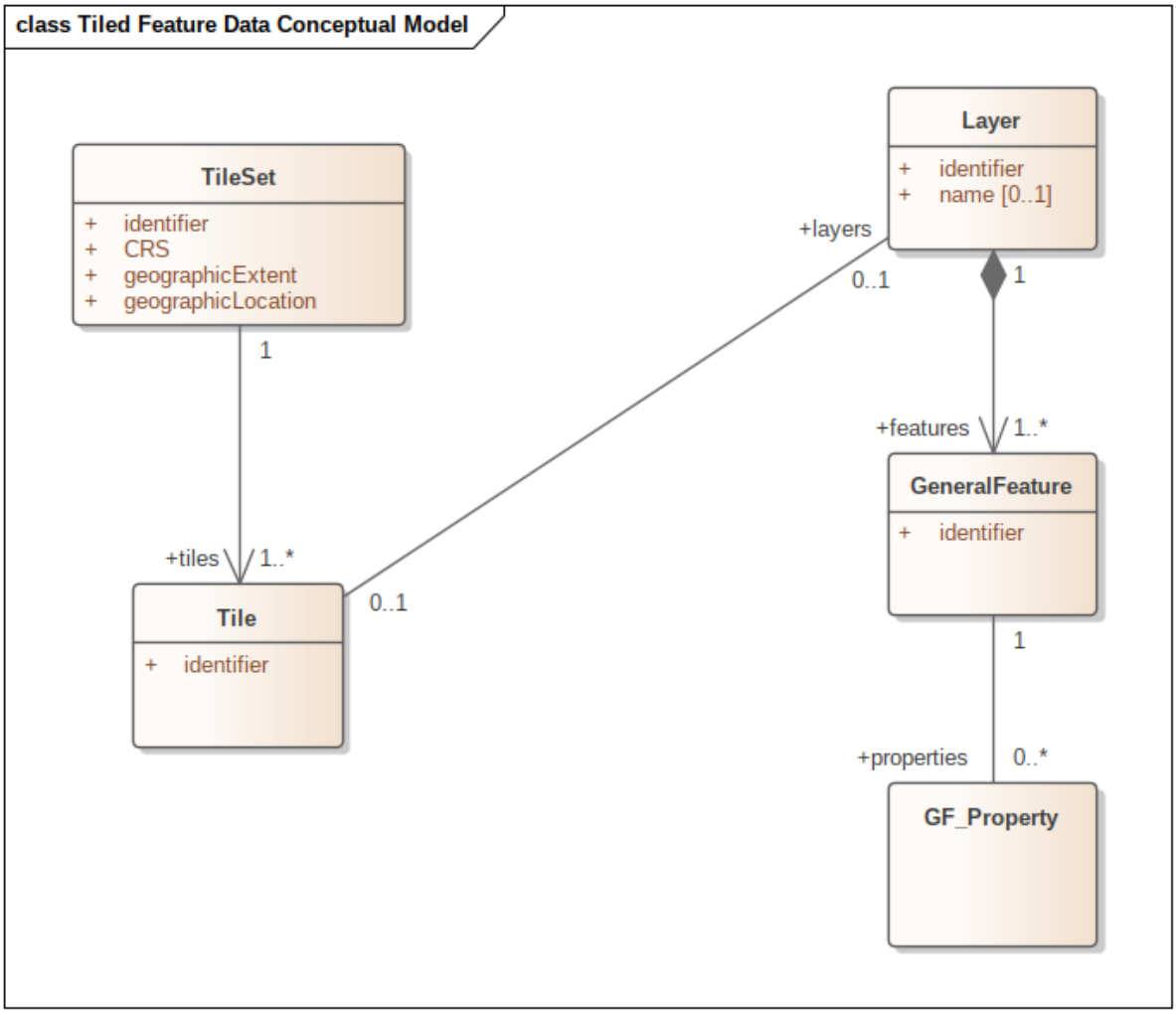

Figure 7. Vector Tiles Conceptual Model

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 excepting the `TileSet` element. Subsequent versions of the Tile Set Metadata Model saw the `Tile`, `General Feature` and `General Feature Properties` 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 of a tile set. It contains all the elements required to describe the main characteristics of a tile set. The conceptual model covering Tile Set metadata is summarized in the Unified Markup Language (UML) class diagram of Figure 8.

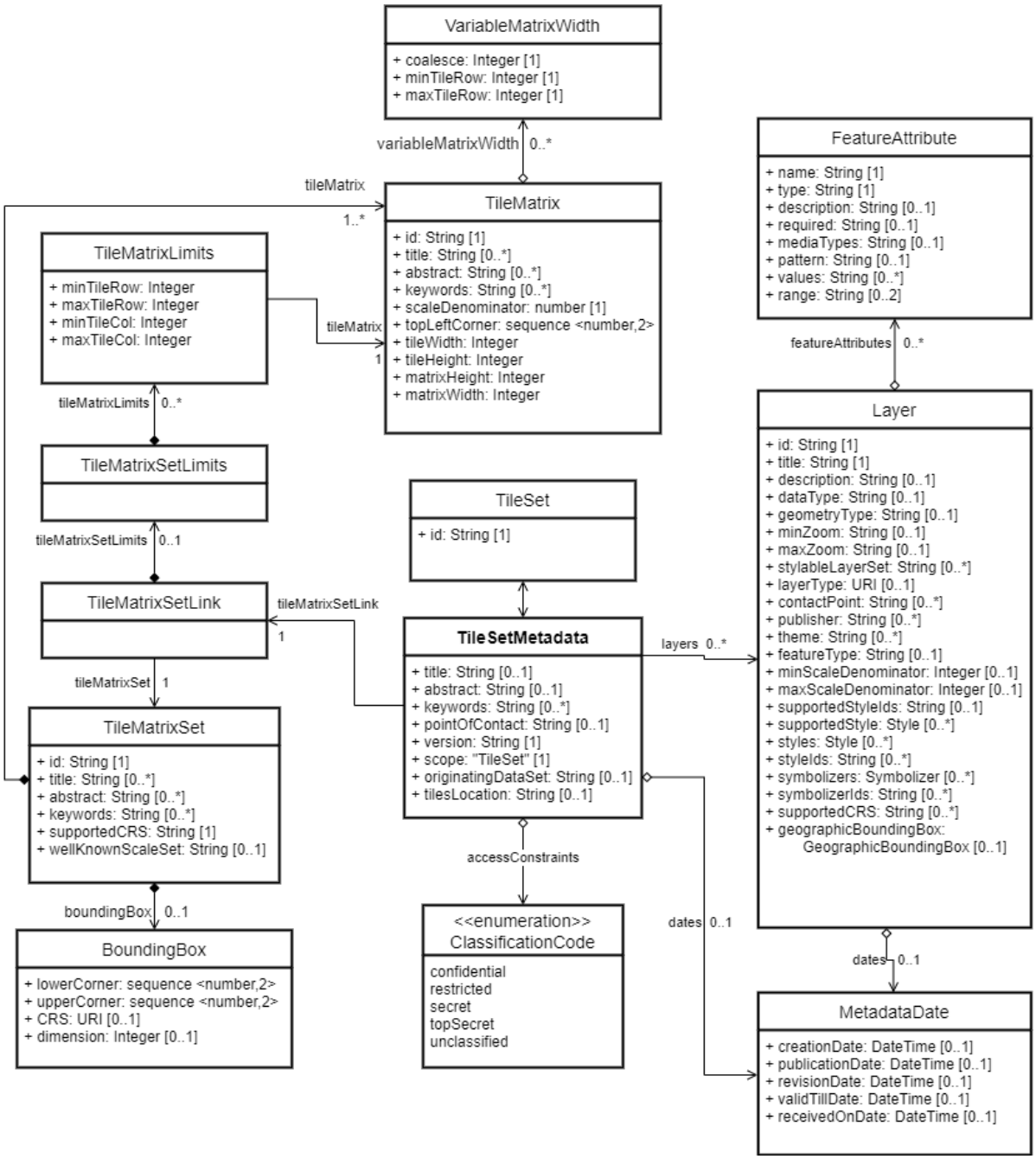

Figure 8. TileSet metadata conceptual model

The model can be succinctly divided in 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`, represented the tiling scheme of the tile set as defined by the OGC Tile Matrix Set Standard.
- The `Layers` and `FeatureAttributes`, the latter comprising the actual feature properties that could be found within the Tile Set.

7.2. Elements

7.2.1. TileSet

A Tile Set is a definition of how tiles are organized. It contains a definition of the geographic extent and geographic location as well as a CRS. One tile set contains one or many tiles.

This class in this model is not defined to its extent in order to avoid including unnecessary information.

7.2.2. TileSetMetadata

The Tile Set Metadata describes the Tile Set with a simple metadata entry, as well as some structural information.

At the core of the Tile Set Metadata the following information is enumerated:

- Id: the Tile Set identifier, same as the TileSet class.
- Title: a title for the Tile Set, suitable for indicating the Tile Set usage in a Tile Set listing.
- Abstract: a longer description of the Tile Set, including all information needed to facilitate understanding, such as applicability, classification methods, reference to norms and whatever other information the user should be aware of before using the Tile Set.
- Keywords: a list of well-known keywords that would help locating the Tile Set in a catalog.
- 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: a version number for the Tile Set.
- Scope: with a fixed value of Tile Set.
- AccessConstraints: an indication about the availability of the Tile Set that the user with access to the Tile Set need to be aware of before using or redistributing the Tile Set in question. Possible values are confidential, restricted, secret, topSecret and unclassified.

A Tile Set can contain multiple date registries of the following types:

- Creation: the timestamp when the Tile Set was first produced.
- Publication: the timestamp when the Tile Set was first made available to the users.
- Revision: the timestamp of the last Tile Set change/revision.
- ValidTill: the 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).
- ReceivedOn: when the Tile Set was received from an external provider.

7.2.3. TileMatrix

A Tile Matrix is a tiling scheme defined as a grid coverage. cite:[Maso2019]

7.2.4. TileMatrixSet

A Tile Matrix Set is a tiling scheme composed of a collection of tile matrices, optimized for a particular scale and identified by a tile matrix identifier. cite:[Maso2019]

Table 1. Structure of TileMatrixSet

Attribute	**Definition**

Table 2. Structure of TileMatrixSet

Attribute	**Definition**
identifier	Tile matrix set identifier
title	Title of this tile matrix set, normally used for display to a human
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
boundingBox	Minimum bounding rectangle surrounding the tile matrix set, in the supported CRS b
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

7.2.5. TileMatrixSetLink

A Tile Matrix Set Link is a container of both a Tile Matrix Set and, if existant, the limits defined to it by a TileMatrixSetLimits element. cite:[Maso2019]

7.2.6. TileMatrixSetLimits

Tile Matrix Set Limits 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.

7.2.7. TileMatrixLimits

Tile Matrix Limits 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.

7.2.8. BoundingBox

Minimum bounding rectangle surrounding the tile matrix set, in the supported CRS.

7.2.9. Layer

A Layer is a set of geographic objects of the same type.

7.2.10. FeatureAttribute

A Feature Attribute element contains attributes that can be found in at least one feature belonging to the layer the FeatureAttribute element belongs to.

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 2

Table 3. 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
TileMatrixSet and TileMatrix	Tiles API	/tileMatrixSets/{tileMatrixSetId}/	
		/collections/{collectionId}/tiles/tileMatrixSets/{tileMatrixSetId}/	
TileMatrixLimits	Tiles API	/collections/{collectionId}/tiles/	

TileSet Metadata Element	Origin API	Path	Comments
Collection	No API	No path	Exclusive for Collection 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 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 2.

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.

Two TIE types are contemplated in this section:

- TileJSON Metadata TIEs: consisting on importing Tiles Metadata from client providers, stored as TileJSON encodings.
- Tile Set Metadata TIEs: consisting on generating and accessing Tile Set Metadata out of a particular generated tile set.

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 3. 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 4.

Table 4. 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
GeoSolutions D102 Tiles API	X	X		X	X
interactive instruments D101 Tiles API	X	X	X	X	X
Terranodo D100 Tiles API	X	X	X	X	X

Table 5. 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. Ecere

8.1.3. Skymantics

8.1.4. Terranodo

8.2. Tile Set Metadata TIEs

8.2.1. GeoSolutions

GeoSolutions developed a WebGIS applications based on the Open Source Framework MapStore [https://mapstore.geo-solutions.it/] aimed to show a complete workflow were 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 9) on the client and then by clicking on a download button (Figure 10) 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`).

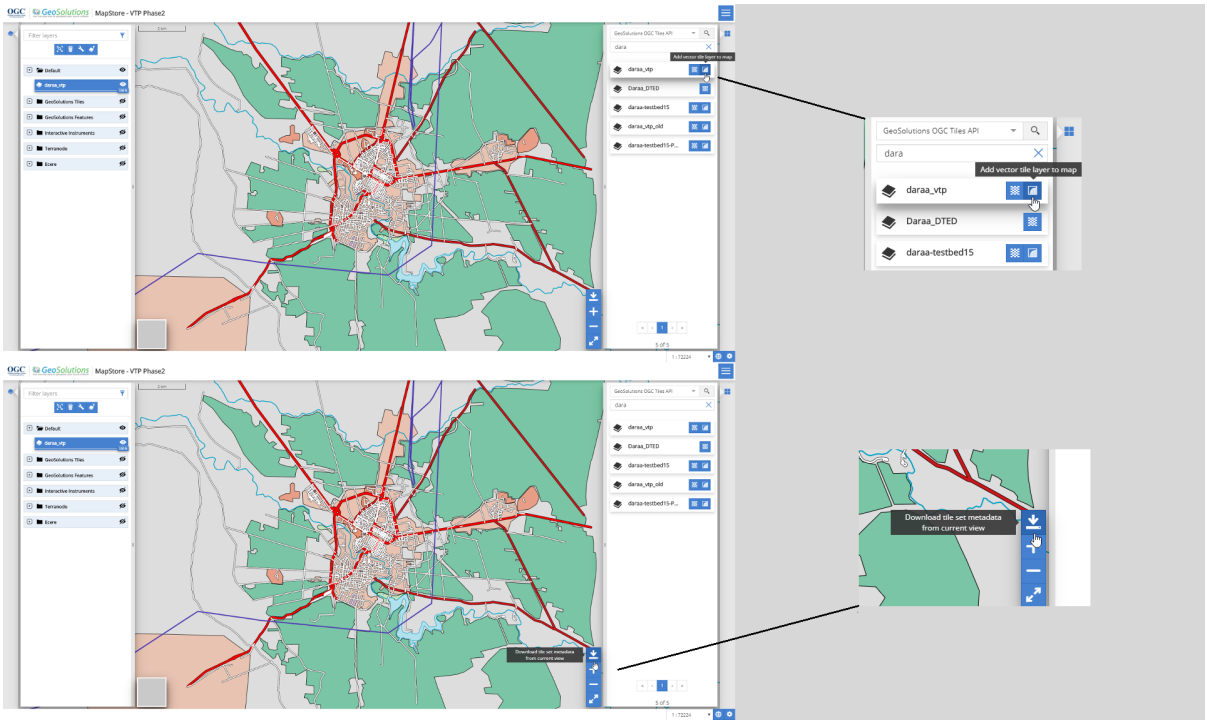

Figure 9. GeoSolutions - MapStore web client - steps to open tile set metadata download form: 1 add collection from an OGC Tiles API and 2 enable the download metadata plugin

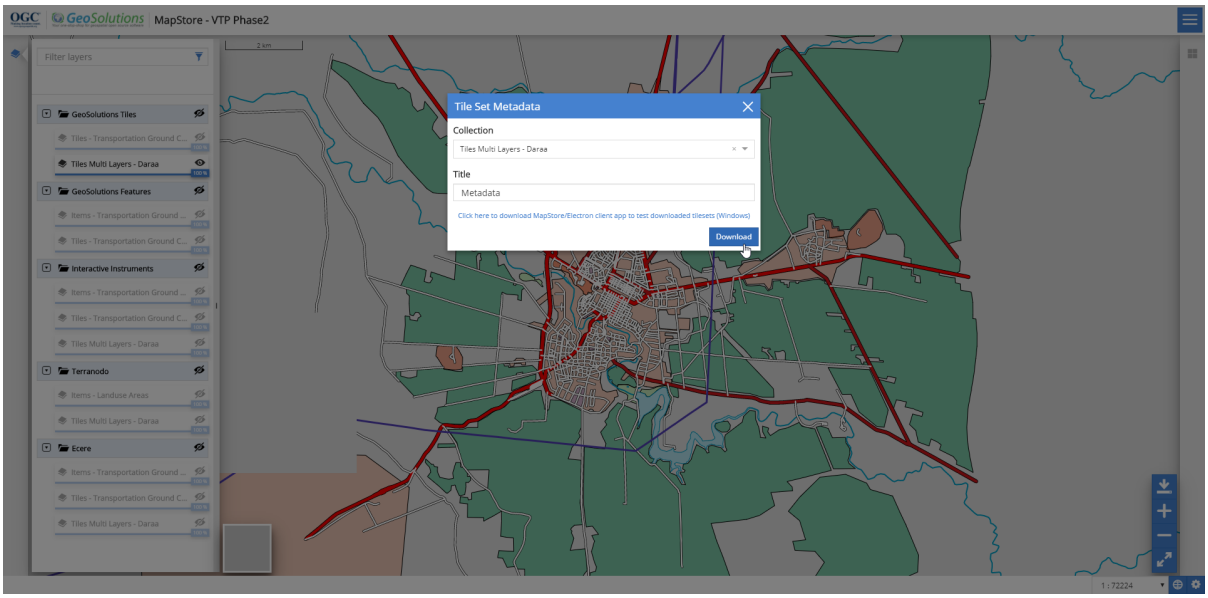

Figure 10. GeoSolutions - MapStore web client - display the tile set metadata download form

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 12, a button allowed the end user to visualize the Tile Set Metadata in a tree structure.

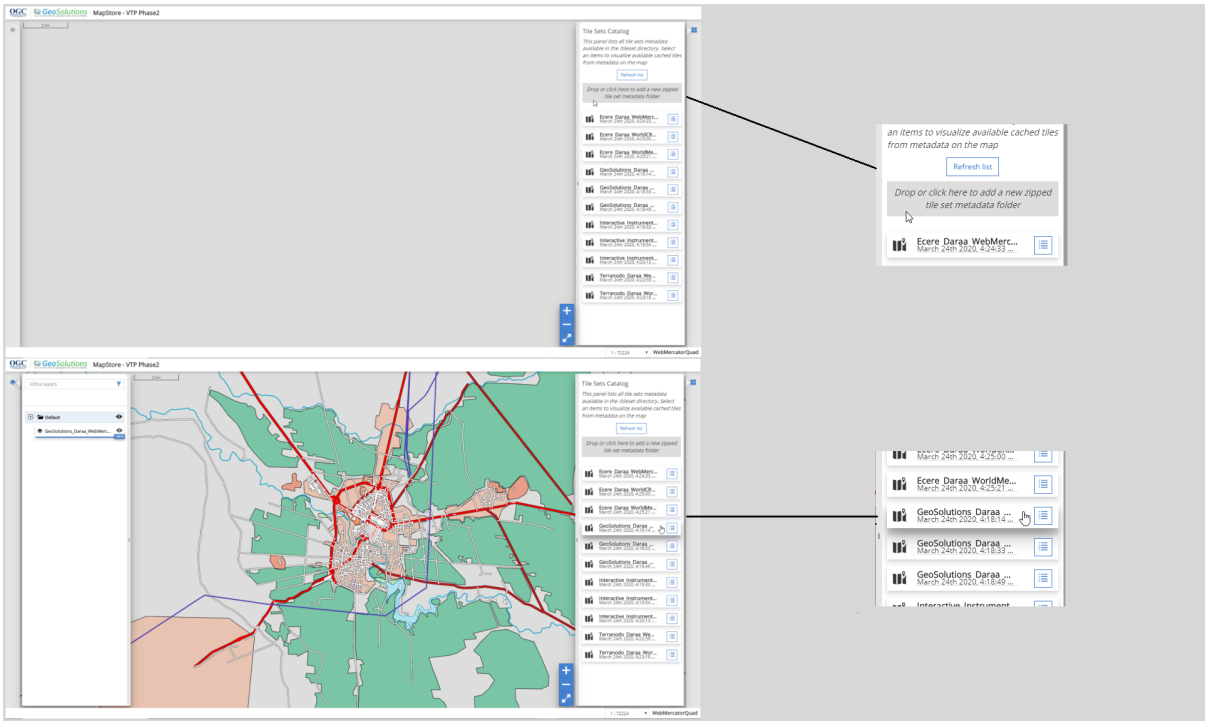

Figure 11. GeoSolutions - MapStore Electron client - steps to read and display tile set metadata zip file: 1 drag and drop zip folder on the grey field and 2 click on the item in the list representing the new tile set

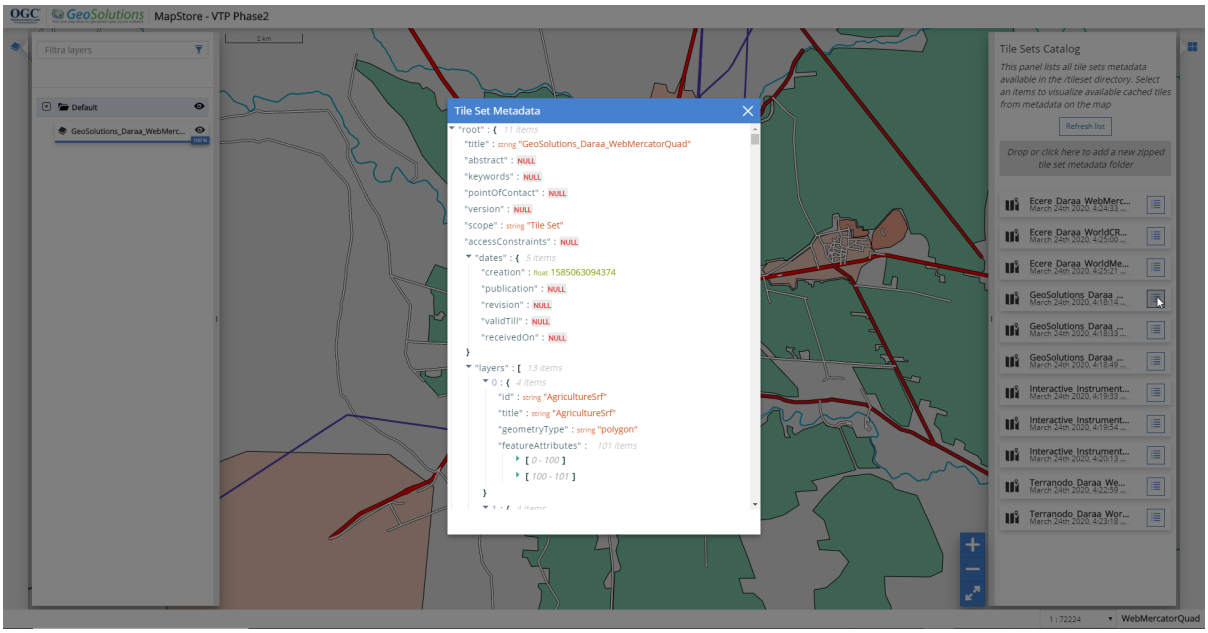

Figure 12. GeoSolutions - MapStore Electron client - display tile set metadata json generated in the zip folder

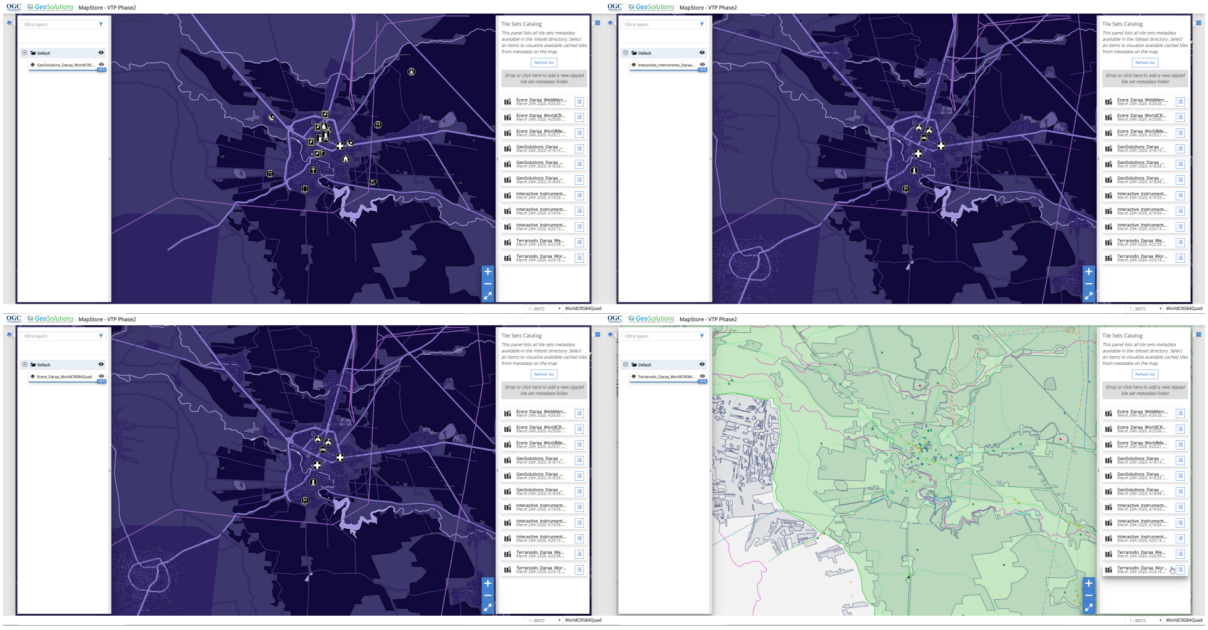

Figure 13. GeoSolutions - MapStore Electron client - display tile sets generated in WorldCRS84Quad from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

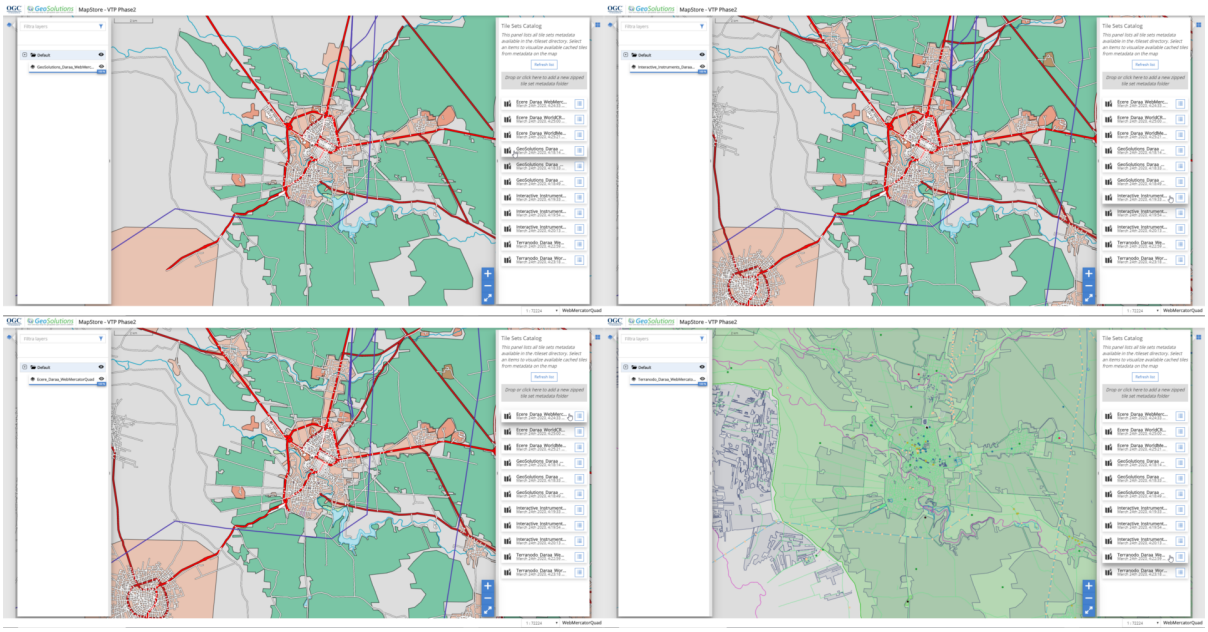

Figure 14. GeoSolutions - MapStore Electron client - display tile sets generated in WebMercatorQuad from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

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

Use of TileJSON metadata

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 10
- 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

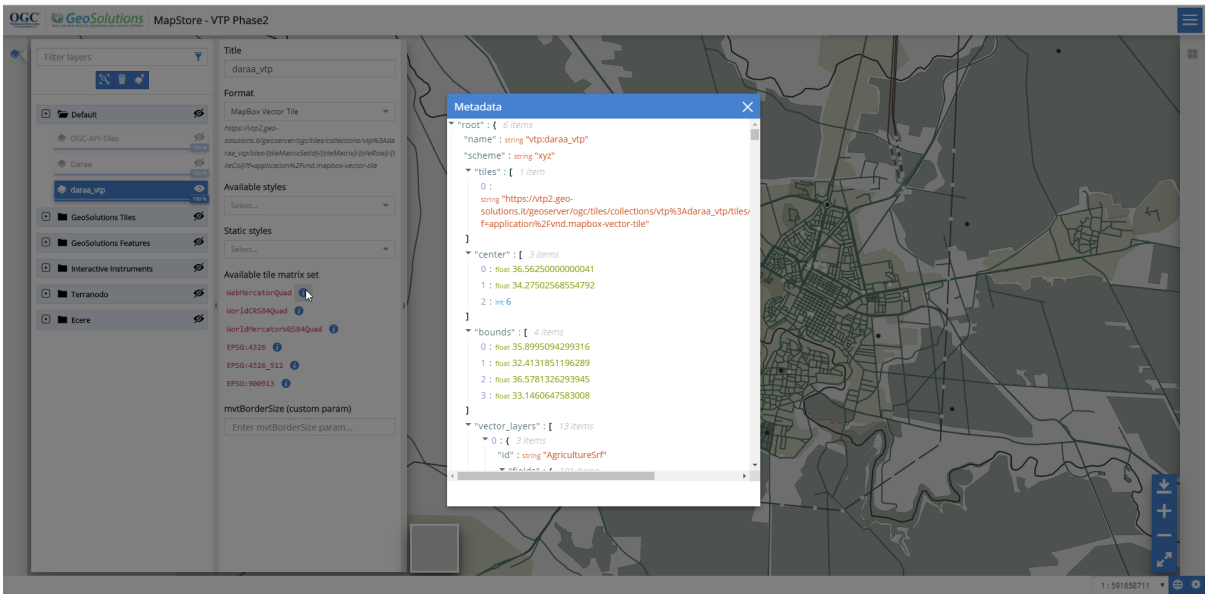

Figure 15. GeoSolutions - MapStore web client - display information from tilejson metadata in layer settings when available

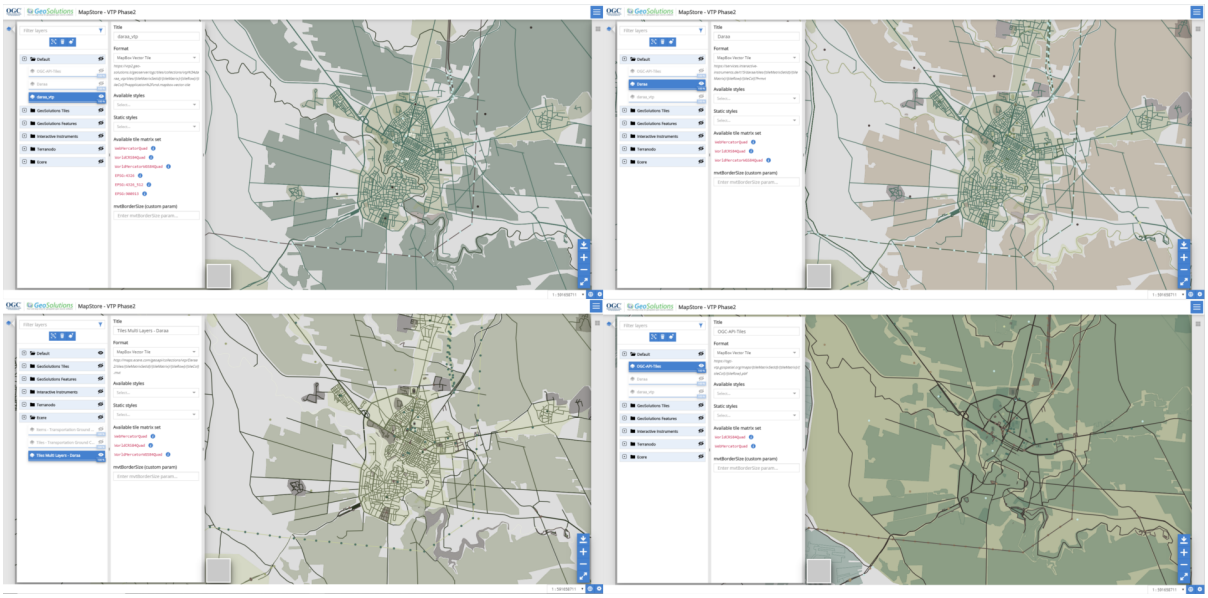

Figure 16. GeoSolutions - MapStore web client - generate on the fly mapbox style based on the vector_layers information of tilejson metadata from different OGC API Tiles servers: GeoSolutions (top left), Interactive Instruments (top right), Ecere (bottom left) and Terranodo (bottom right)

8.2.2. Image Matters

Image Matters developed a GeoPackage Producer that implemented the proposed TileSet Metadata Model in a GeoPackage. As described by GeoPackage Loading, TileSet metadata is required to populate a GeoPackage properly.

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`

Once these operations occurred, the GeoPackage producer populated the GeoPackage with the appropriate information. Through the GeoPackage Producer, Image Matters TIEd successfully with each of the four vendors.

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 exact definition of what a `cache` means in computer science, i.e. *a hardware or software component that stores data so that future requests for that data can be served faster*. Caches are extensively used in software applications due to the benefits they offer. This can generate confusion as some software applications might actually store tiles in a cache in order to access those tiles faster, therefore creating a tile cache as defined by computer science terminology.

After internal discussions, consensus was reached on dropping the use of the term `Tile Cache` in favor of using `Tile Set`. The adoption of this new term was not completely successful during the Pilot, for the reasons described in the Findings section of this report.

9.2. Location of Tile Set Metadata

Ideally, any data regarding geographical elements should be accessed by one of the APIs of OGC. As new elements are created to expand the functionalities of GIS, new resources would be created on one or several APIs in order to access these new elements.

A considerable amount of data included in the Tile Set Metadata Model was already available in OGC APIs:

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

The remainder metadata not currently available in any API of OGC consisted in the main `TileSetMetadata` element described in the Metadata Model in Chapter 7. This was expected, as the Tile Set Metadata is a novel element introduced during the course of this Pilot and the main objective of this particular ER.

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

- Adding the resource at any level of the Records API of OGC. Rejected for being the use of Records API out of scope for this Pilot.
- Adding the resource at the root of either Tiles API or Features API. Rejected due to concerns

over compatibility with the design of the APIs of OGC.

- Adding the resource at the `TileMatrixSet` level, i.e. `/tileMatrixSet/{tileMatrixSetId}/metadata`. 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`. Rejected due to the fact that one Tile Set can be composed by 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 API of OGC; the main benefit of this approach was to prevent storing duplicated metadata information, but 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 by 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 OGC Tiles API 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 highly discussed during the course of the Pilot and the solution agreed by participants consisted on 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 V3 document.

The inability of Tiles API to support multi-layer MVTs posed a limitation for Tile Set Metadata to support tile sets containing multi-layer MVTs, thus limiting the use cases where the TSM could actually be applied to. As the proposed solution allowed Tiles API to support multi-layer MVTs, it also allowed Tile Set Metadata to support them.

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

- TileJSON provided support for multi-layer MVTs.
- TileJSON was at the time of the Pilot a de-facto standard for describing a tile set. It was supported by a considerable number of tools working with tile sets. Using TileJSON had as a benefit the fact that the APIs developed in this Pilot could immediately be used out-of-the-box with existing tools, for example 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. All APIs deployed in the pilot supported this tiling scheme.

- TileJSON 3.0.0 did not cover all of the tile set metadata, but its content was sufficient for the purposes of this Pilot. Since additional members may be added to a TileJSON document, additional information could be added in the future, where needed.
- Even though the latest version of TileJSON was 2.2.0, this version did not support layer information, which had been added in the draft for 3.0.0. 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 be 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, consensus was reached 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 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 17.

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.

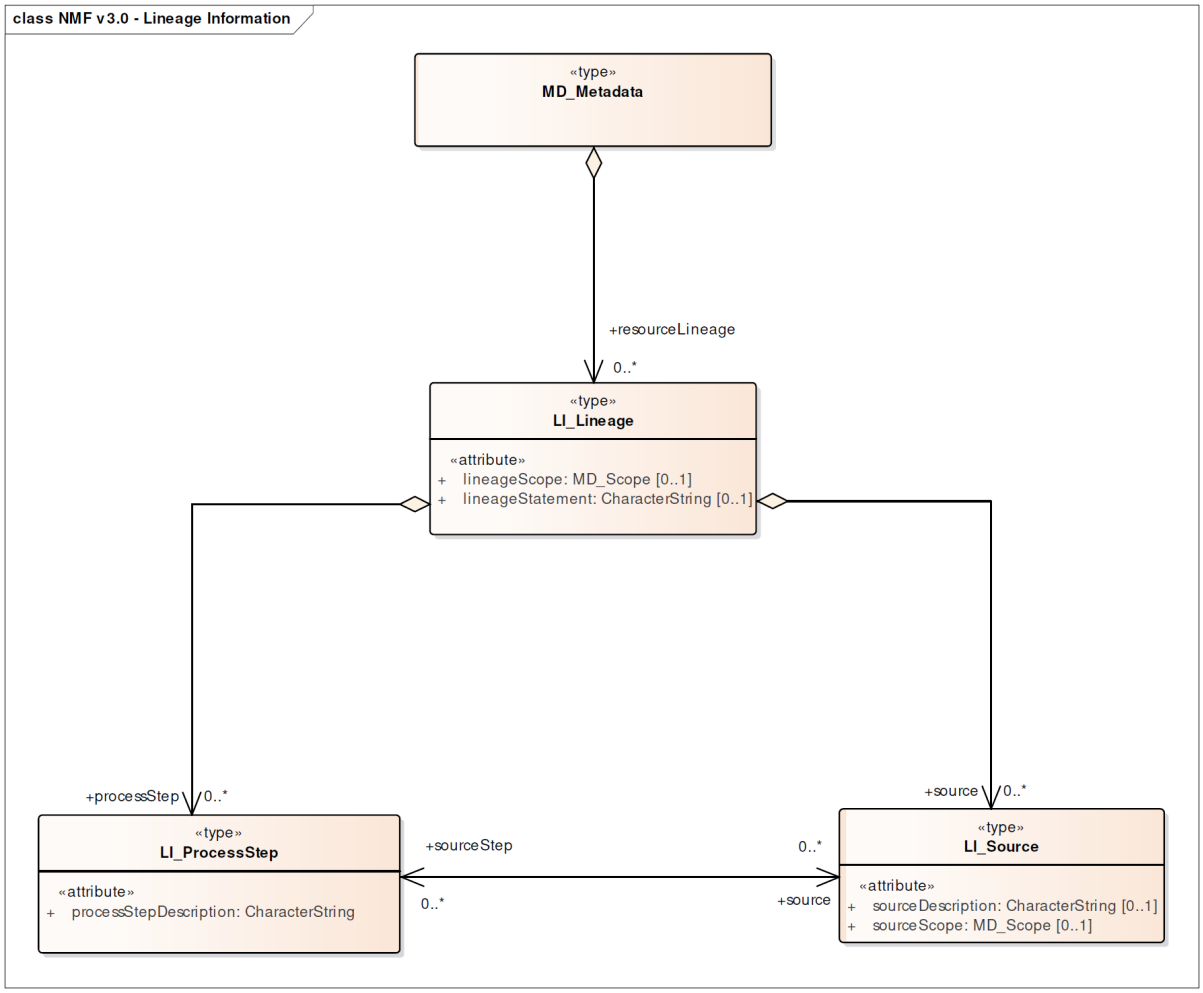

Figure 17. NMF Lineage Class Structure

9.4.3. Constraints

The Resource Constraints in the NMF, as shown on Figure 18, 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. The majority of the classes within Resource Constraints are designed to provide support for 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.

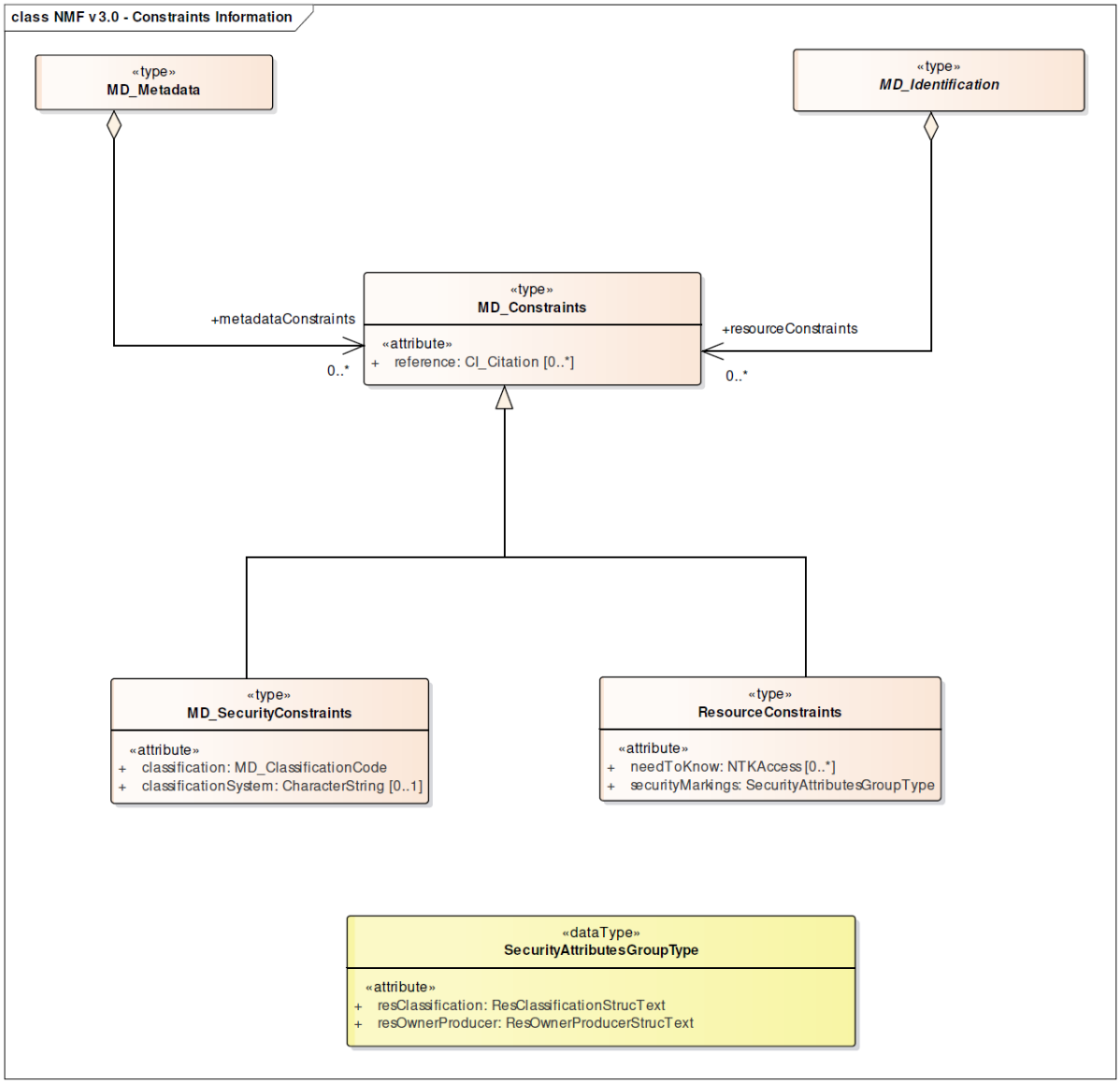

Figure 18. NMF Resource Constraints Class Structure

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*. The TMS 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.

Chapter 10. Results

The efforts carried out throughout the Pilot resulted in the following achievements:

- A Tile Set Metadata Model was agreed upon among the participants.
- 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 tile servers successfully advertised TileJSON tiles metadata and all of them were consumed by at least one client.

Chapter 11. Findings

The following are findings made by VTP2 participants in relation to the metadata aspects of the project. These findings, together with the lessons learned, will serve as a foundation for future work of tile set metadata implementations.

The draft abstract specification for the OGC Tiling Conceptual Model and Logical Model for 2-D Planar Space (OGC 19-014) identifies a generalization relationship between the Tile Set and Tile Set Metadata. The generalization relationship is shown in Figure 6. 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 therefore modeled the relationship between a Tile Set and Tile Set Metadata as a bi-directional simple association, as shown in Figure 7.

In relation to TileJSON, the participants found that the path `/tiles/{tileMatrixSetId}/metadata` used in the pilot APIs 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, for use of TileJSON in an OGC API standard the latest published version should be used, extended as necessary. It would also be important to have a registered media type for TileJSON. 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 encoding. For rendered tiles in image formats another parameter would be needed for the style used to render the tiles. For GeoPackages, the latest extensions (Vector Tiles, Tile MatrixSet, Mapbox and GeoJSON 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 all of the tile set metadata pieces very well. However, it does not have a direct representation in either the 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, 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 following are primary lessons learned by VTP2 participants 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. Before the decision was made to implement Tile Set metadata, the GeoPackage Producer 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:

The TIE demonstrated the need for including [attribute] in the [element] element. [Exact use case where this was demonstrated] * 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, Image Matters recommends 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 tiles API, a lot of that information is available from the /tiles resource, which itself also references TileMatrixSets descriptions or URIs.

Additional information not currently specified by the Tiles API is needed to describe the fields and geometry type and in the case of multi-layer 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 at the 'collection' level, as this information applies the same whether are dealing with tiles or not. It would work especially well for multi-layer tiles if hierarchical collections are supported.

Towards the end of VTP2, participants agreed to relay 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

This Pilot explored in depth the metadata requirements for tile sets and specifically their requirements when transitioning from online to offline environments.

- Some elements discussed 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 Records API of OGC 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 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. Standarizing a common metadata encoding

Push a standard based on TileJSON?

13.1.2. Adding Styles to Downloaded Tile Sets

The direct encoding of that tileset metadata in JSON, as implemented by the work of GeoSolutions, is a great 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 on 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 therefore denying the offline capability of working with styles. The 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 complete 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 6. 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

Annex C: Bibliography

bibliography::[]