OGC Testbed-14

Symbology Engineering Report

Table of Contents

Publication Date: 2018-03-15

Approval Date: 2018-12-13

Submission Date: 2018-11-28

Reference number of this document: OGC 18-029

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D029

Category: Public Engineering Report

Editor: Sara Saeedi

Title: OGC Testbed-14: Symbology Engineering Report

OGC Engineering Report

COPYRIGHT

WARNING

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

LICENSE AGREEMENT

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

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

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

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

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

indicate compliance with any LICENSOR standards or specifications.

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

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

Chapter 1. Summary

The portrayal and visualization of geospatial information is a critical task for facilitating decision making, situational awareness, and spatial analysis. However, despite its importance, various local, national, and international agencies continue to use different symbols and terminology for the same event, feature, or entity. This approach prevents interoperability from being extended to the semantic level, which in turn makes it difficult to share, reuse, and mediate unambiguous portrayal information between agencies.

This Engineering Report (ER) captures the requirements, solutions, models, and implementations of the Open Geospatial Consortium (OGC) Testbed-14 Portrayal thread. This effort leverages the work of the Portrayal Ontology development and the Semantic Portrayal Service conducted during Testbed 10, 11, 12 and 13. Thus far the emphasis for developing the portrayal ontologies (Testbeds 12 and 13) has been on modeling and representing portrayal information for feature data. The objective of Testbed-14 is to extend the portrayal ontology to accommodate more complex symbols (*e.g.*, composite symbols) and to provide clear recommendations on how to best proceed with portrayal information encodings.

1.1. Requirements and Research Motivation

A detailed description of the requirements that have been addressed – alongside the fundamental questions of our motivation for addressing this topic – is documented in Chapters 5 to 9 of this ER.

1.2. Prior-After Comparison

The Styled Layer Descriptor (SLD) specification was introduced in 2001 as a part of version 1.1.0 of the Web Map Service (WMS) Implementation Specification (OGC 01-047r2). Currently available in version 1.1, the OpenGIS Styled Layer Descriptor Profile of the Web Map Service Implementation Specification was published in 2007 with the aim of extracting the symbology instructions and turning them into the dedicated Symbology Encoding standard (SE 1.1). Consequently, while the SLD profile remains closely related to the WMS service, the symbology instructions can now be used by any styling software component.

Over the past ten years, SLD and SE have not been enthusiastically adopted although they remain the underlying method for styling layers for many geospatial software products. Furthermore, visualization tools have taken symbology requirements far beyond those envisioned in 2001, not only because 2.5D (2.5-Dimensional) and 3D (3-Dimensional) data have become common, but also because these tools are available to a broader community of map makers taking over and pushing the boundaries of 2D (2-Dimensional) visualization.

The objective of Testbed-13 was to identify and complete the gaps in the latest version of the portrayal ontology defined in Testbed-12, complete the implementation of the Semantic Portrayal Service by adding rendering capabilities and perform a demonstration of the portrayal service that showcases the benefits of the proposed semantic-based approach. The Portrayal Concept Development Study (CDS) [http://docs.opengeospatial.org/per/17-094r1.html] undertook the following activities: assessing the current state of feature portrayal, establishing a long-term vision for portrayal standardization and providing a roadmap for capabilities identified in the vision [1].

Testbed-14 has extended the portrayal ontology to accommodate more complex symbols (*e.g.*, composite symbols) and provide recommendations on how to best proceed with portrayal information encodings. One motto used for Testbed-14 for portrayal was "the time for convergence" or "on the path to convergence." However, the end of the path has not been reached, whether for Testbed-14 or for the previous testbeds which only gathered the logical and relevant pieces about portrayal interoperability points being pursued which were in line with the CDS [http://docs.opengeospatial.org/per/17-094r1.html].

As planned, Testbed-14 began an assembly of many visions on various requirements from linkable portrayal information (inherited from previous testbeds) to multiple-pass rendering symbolizer composition. Visions, requirements, expectations and experiences came from work done in previous testbeds, in the SLD/SE Standards Working Group (SWG) and from existing encoding formats used by geospatial cloud platforms and server solutions.

1.3. Recommendations for Future Work

With the experience gained, and analysis of existing encoding formats, the Testbed-14 team was able to extract useful lessons for improving the portrayal conceptual model, mainly as a set of ontologies and micro-theories that had already been built by previous testbeds. Nonetheless, this is just a start. For example, considering CartoCSS [https://cartocss.readthedocs.io/en/latest/] or GeoCSS opens up so many possible and relevant visual properties that another iteration is required to refine the details of each implementation (*e.g.*, labeling properties to control deconfliction and optimize label rendering). Some of these issues are effectively part of a basic profile, while others are part of specific profiles.

Furthermore, supporting multiple community symbology use cases using the same data encodings is required for both military and non-military responders to the same incident or event. It is recommended in the way forward that a feature be added to the Portrayal Registry construct that enables a federation of sharing or getting styles and support for a React [https://reactjs.org/] (JavaScript library) framework.

More importantly, some key issues are highlighted for consideration for the **general Open OGC roadmap for portrayal**. There are two possible directions it can take:

1. Using the rich material from all these testbeds and stabilizing the conceptual work with a default encoding for a core (please refer to OGC 18-067) and a basic profile for 2D vectors (or even a super simple profile as discussed in this testbed). Perhaps this may be internal work that can be carried out by a SWG or finalizing work (which expects the specification of standards) for a future testbed.
2. Preventing the internal standardization work within the current working groups and continuing with more testbeds according to the future deliverables. Before returning to the standardization work that is focusing on the definition or consolidation of the conceptual model, exploration of raster and 3D (Augmented Reality, *etc.*) could be very useful for an extensible core standard.

Moreover, there is a requirement to support the usage of multiple symbology sets against the same data encoding. The recommended use case for a future testbed would be along the lines of Counter-IED (improvised explosive devices) event where both military and non-military responders are

responding to the same incident. In this case, military responders would use Mil-Std-2525 or Allied Procedural Publication 6 (APP-6) whereas civilian first responders would use a symbology set such as ANSI 451.

1.3.1. Recommended Future Tasks

The following tasks can be considered for the future work:

Conceptual model, abstract specification, ontology

There is a requirement to define the "conceptual model" term for clarity with regards to the use of the term for future testbeds, use cases, and standards documents about portrayal. Note that some recent OGC discussions relating to the definition of a conceptual model slip into specific details about implementation. For other OGC documents, conceptual models are an abstraction (used to represent abstract ideas). There are some examples of a conceptual model as an abstraction with no specified implementation details or encoding formats, such as Simple Features Architecture, GeoSciML (Geoscience Markup Language), and OWS (OGC Web Services) Context. There may also be confusion regarding the relationship of "conceptual model" and OGC "abstract specs." See the tentative definition in the Terms and definitions section of this ER and the note in Chapter 5 that also discusses the ontological approach.

There can also be further discussions as to whether a core conceptual model is required to define symbology and other extensions to it. However, without a conceptual model, there will be no agreement on semantics (terms and definitions). A conceptual model is helpful for the following tasks:

- Facilitating collaboration between all team members;
- Helping to update, change, and reuse the model easily;
- Representing the abstract concepts and explaining basic entities and their relationships using both visual and written forms of diagrams;
- Defining relevant terms; and
- Minimizing the likelihood of incomplete, unclear, inconsistent, and wrong scopes, requirements, and validation.

From modular specifications to best practices

The idea may arise that portrayal standards stemming from the current process might lead to complex tools and concepts only for specialized users. This point is especially valid when comparing the OGC approach with current geospatial cloud platforms or web server solutions. These are often implementations which do not refer to anything else apart from the usual business needs from the "web mapping market." However, remember that the targeted audiences for OGC portrayal standards are Information Technology (IT) teams in charge of the implementations that require interoperable portrayal systems and the indirect final users whose lives are made easier by interoperable features between the systems deployed on the market. This case is very similar to the original Web Mapping use case that led to the development of the OGC WMS standard, which has been adopted by the International Organization for Standardization (ISO) as ISO 19128:2005. Even if implementations hide the underlying complexity from the final users, it is important to favor the adoption of these portrayal standards by the IT strategist and developer teams. In other words, it

should also make their lives easier. Modularity may help to favor such an adoption. If modularity is not an essential requirement for the above platforms and solutions, it is for the design of standards if their widespread adoption is desired and also to address interoperability requirements.

With modularity in mind, the following tasks are anticipated for the relevant SWGs to discuss and define portrayal standards:

- First, the currently defined core that has gone to public comment (Symbology Conceptual Core Model: OGC 18-067 [http://www.opengeospatial.org/pressroom/pressreleases/2893]) must be evaluated in terms of the Testbed-13/14 findings and recommendations.
- Then, using that core, define extensions that provide additional details using the main extension principles proposed in OGC 18-067 [http://www.opengeospatial.org/pressroom/pressreleases/2893] (also notice the Testbed-14 reverse engineering section (Lessons from CARTO and Mapbox underlying encoding formats) which is a methodological attempt to extract some natural extensions from the underlying model of Cartographic Cascading Style Sheets (CartoCSS); it should be continued).
- Next, these extensions have to be documented in relation to requirement classes and requirement identification (as defined in the OGC modular spec OGC 08-131r3 [https://portal.opengeospatial.org/files/?artifact_id=34762]) and illustrated in OGC 18-067 [http://www.opengeospatial.org/pressroom/pressreleases/2893] appendix.
- Finally, provided that the three previous points have been reached, the conceptual definition of the various profiles and/or extensions can be documented accordingly.

Moreover, we may notice that Field [2] points out that the current demand from internet users (not in the discipline of cartography) is for quantity, not quality, of symbology. To counteract this situation or evolution, and to favor the adoption of OGC portrayal standards to be considered standards of reference for the market, best practices (BPs) should support those portrayal standards. These BPs can highlight the importance of cartographic design and its relationship with portrayal standards. That is in the OGC defined best practices documents containing a discussion of best practices related to the use and/or implementation of an adopted OGC document for release to the public.

Rendering sequence and ordering considerations

Ordering (also called *Z-order*) is a sorting capability of overlapping graphics, such as polygons or lines in a vector dataset. For example, as it is mentioned in chapter 9, Mapnik uses the painter's algorithm when drawing objects, meaning that items are drawn from bottom to top, with items defined earlier in the source layer list being drawn first. Consequently, stylesheet developers should consider what features should appear over other features, such as roads over land uses, points of interest over buildings, and so on.

There are more complex situations, such as "spaghetti" road junctions, where multiple highways may intersect other multiple highways. A map should accurately portray the vertical topology of overlapping roads. This topic needs to be further developed concerning the following issues:

- To what extent ordering considerations (and abilities) have to be included in the basic profile, given that the elaboration of an OpenStreetMap (OSM) style is highly dependent on such abilities, but is not presumably the basic use case (instead, some of these abilities are part of an

advanced profile).

- The ability to sort features in the same layer by their vertical topology requires the attributes that define a feature's "relative" or "absolute" vertical position in the source data. More descriptions on the relative or absolute vertical position are discussed in chapter 9.

Default values

Default values remain a topic that needs future work, as raised in the section Default values and color definition in Chapter 5, which also relates to the definition of a first and default encoding.

Raster symbology requirements

Testbed-13 recommended work on raster symbology. Based on those discussions, Testbed-14 suggests a new generation of "Symbology Encoding" for raster. This is in part because the Symbology Encoding Implementation Specification 1.1.0 [http://portal.opengeospatial.org/files/?artifact_id=16700] discussed both 2D vectors and also raster. Accordingly, the next step would be to evaluate the raster abilities of the conceptual model which would be *CoverageStyle* with a specific symbolizer, according to a clear definition of the related raster data model(s) for future testbeds.

3D symbology requirements

Another future direction is to keep the testbed work oriented towards vectors (2D or 3D); in other words, future testbeds can be dedicated to symbology for 3D requirements. Confronting the conceptual model with the 3D use case is essential for identifying 3D requirements and other specific extensions. Being clear about the idea of "confrontation" will challenge the initial global structure and its extensibility.

From a reference encoding to translators

Testbed-14's effort in writing an encoding specification document that is compliant with the conceptual specification document can be extended to including a reference encoding of the conceptual model. Following this, a couple of translators or encoders from an implementation (Mapbox, CartoCSS, Geoserver, SLD/SE, *etc.*) can be created for that reference encoding.

1.3.2. Recommended Future Deliverables

Based on the results and discussions made for Testbed-14, the following deliverables are suggested for deployment for testing and demonstrating the above functionalities and for documenting the results.

1.3.3. Recommended Future Engineering Reports

- Default reference encoding format specification for the vector 2D basic profile (based on propositions from Testbed-14).
- Extending the conceptual model with the idea of reusable "symbolizer templates" (in relation to the *ParametrizedSymbolizer* of OGC 09-016 [http://portal.opengeospatial.org/files/?artifact_id=33515] and considering sharing principles from Testbed-13).
- Defining raster symbology abilities as extensions (and package for one or more dedicated profiles).

- Defining 3D symbology abilities as extensions (and package for one or more dedicated profiles).

1.3.4. Recommended Future Components

- Rendering engine compliant with the default reference encoding format (with the expectation that it is compliant with a finalized conceptual model).
- Controlling the rendering engine with a map-building user interface.

1.4. Document Contributor Contact Points

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

Contacts

Name	Organization
Sara Saeedi (Editor and Contributor)	University of Calgary
Olivier Ertz	Media Engineering Institute, HES-SO // University of Applied Sciences Western Switzerland
Thibaud Chassin	Territorial Engineering Institute, HES-SO // University of Applied Sciences Western Switzerland
Stefano Bovio	GeoSolutions S.A.S.
Simone Giannecchini	GeoSolutions S.A.S.
Andrea Aime	GeoSolutions S.A.S.
Gate Jantaraweragul	ImageMatters LLC
Jeff Yutzler	ImageMatters LLC
Jérôme St-Louis	Ecere
James Badger	University of Calgary
Soroush Ojagh	University of Calgary
Steve Liang	University of Calgary
Carl Reed	Carl Reed and Associates
Greg Buehler	OGC

1.5. Foreword

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

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

Chapter 2. References

The following normative documents are referenced in this document.

NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards. All other references are listed in the bibliography.

- ISO: ISO 19101, Geographic information — Reference model [https://www.iso.org/standard/69325.html]
- ISO: ISO 19119, Geographic information — Services [https://www.iso.org/standard/59221.html]
- ISO: ISO 19109, Geographic information — Rules for application schema [https://www.iso.org/standard/59193.html]
- ISO: ISO/IEC 19763, Information technology — Metamodel framework for interoperability [http://metadata-standards.org/19763/index.html]
- ISO: ISO 19117, Geographic information — Portrayal [https://www.iso.org/standard/46226.html]
- OGC: OGC 06-121r9, OGC® Web services common standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 08-131r3, OGC® The specification model — A standard for modular specifications [https://portal.opengeospatial.org/files/?artifact_id=34762]
- OGC: OGC 18-067, OGC® Symbology conceptual core model [http://www.opengeospatial.org/pressroom/pressreleases/2893]
- OGC: OGC 05-077, OGC® Symbology encoding implementation specification, Version 1.1.0 [http://portal.opengeospatial.org/files/?artifact_id=16700]
- OGC: OGC 05-078r4, OpenGIS® Styled layer descriptor profile of the web map service implementation specification, Version 1.1.0, 2006 [http://portal.opengeospatial.org/files/?artifact_id=22364]
- OGC: OGC 05-077r4, OpenGIS® Symbology Encoding implementation specification (SE 1.1), Version 1.1.0, 2006 [http://portal.opengeospatial.org/files/?artifact_id=16700].
- OGC: OGC Portrayal Concept Development Study [http://docs.opengeospatial.org/per/17-094r1.html]

Chapter 3. Terms and Definitions

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

- CARTO

```
CARTO is a Software as a Service cloud computing platform that provides GIS and web
mapping tools for display in a web browser.
```

- CartoCSS

```
It is an open-source Mapnik stylesheet which is similar to Cascading Style Sheets
(CSS) syntax inspired by Cascadenik and developed by MapBox to render static maps.
The reference parser is written in JavaScript and optimized for large stylesheets.
```

- Conceptual Model (CM)

```
ISO defines a "conceptual model" as "a model that defines concepts of a universe of
discourse" in https://www.iso.org/standard/69325.html[ISO 19101].
In the field of computer science, CM is a particular case of a general conceptual
model which is also called a domain model.
In this ER, CM is an abstract model to represent the abstract concepts, relevant
term, basic entities, their behavior (or attributes) and their relationships used
by domain experts.
A CM is explicitly chosen to be independent of a specific design, encoding or
implementation concerns.
```

- Feature

```
Representation of some real-world object or phenomenon, _e.g._ a building, a river,
or a person.
A feature may or may not have geometric aspects.
```

- GeoDataBase

```
A geodatabase is an Esri spatial data format to store GIS information in one large
file,
which can contain multiple points, polygon, and/or polyline layers.
It is an organization of multiple shapefiles in multiple folders.
```

- GeoJSON

```
GeoJSON is an open standard format designed for representing simple geographical
features, along with their non-spatial attributes. It is based on JSON, the
JavaScript
Object Notation. The features include points, line strings, polygons, and multi-
part collections of these types (source: https://tools.ietf.org/html/rfc7946).
```

- GeoServer

```
It is an open-source server written in Java that allows users to share, process and
edit geospatial data.
It publishes any major spatial data source using open standards and renders
hundreds to thousands of vector/raster map layers (source: http://geoserver.com).
```

- Interoperability

```
The capability to communicate, execute programs, or transfer data among various
functional units in a manner that requires the user to have little or no knowledge
of the unique characteristics of those units (source:
https://www.iso.org/standard/59221.html[ISO 19119]).
```

- HTML Canvas

```
The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of
2D shapes and bitmap images.
It is a raster-based, low-level, procedural model that updates a bitmap in a
browser.
```

- Layer

```
The basic unit of geographic information that may be requested as a map from a
server.
```

- Map

```
A pictorial representation of geographic data.
```

- Mapbox

```
An open-source mapping platform for custom designed maps. The technology is based
on Node.js, Mapnik, GDAL, and Leaflet.
```

- Mapbox GL

```
Mapbox GL is a suite of open-source libraries for embedding highly customizable and
responsive client-side maps in Web, mobile, and desktop applications
(https://docs.mapbox.com/help/glossary/mapbox-gl/).
```

- MapCSS

```
MapCSS is a CSS-like language for map stylesheets. MapCSS is used to define
stylesheets for map rendering in the https://www.openstreetmap.org/#map=3/71.34/-
96.82[OpenStreetMap] project.
```

- Mapnik

```
Mapnik is an open source mapping toolkit for desktop- and server-based map
rendering, written in C++. It supports a variety of geospatial data formats and
provides flexible styling options for designing many different kinds of maps.
```

- Model

```
Abstraction of some aspects of a universe of discourse
https://www.iso.org/standard/59193.html[ISO 19109].
```

- Ontology

```
A formal specification of concrete or abstract things, and the relationships among
them, in a prescribed domain of knowledge http://metadata-
standards.org/19763/index.html[ISO/IEC 19763].
```

- Portrayal

```
Portrayal presentation of information to humans
https://www.iso.org/standard/46226.html[ISO 19117].
```

- Semantic interoperability

```
The aspect of interoperability that assures that the content is understood in the
same way in both systems, including by those humans interacting with the systems in
a given context.
```

- Shapefile

```
A Shapefile is an Esri vector data storage format for storing the location, shape,
and attributes of geographic features. It is stored as a set of related files and
contains one feature class.
```

- Style

```
Determines the appearance of geographic data
```

- SVG

```
Scalable Vector Graphics (SVG) is an XML-based vector image format for two-
dimensional graphics with support for interactivity and animation. The SVG
specification is an open standard developed by the World Wide Web Consortium (W3C)
since 1999.
```

- SVG Tiny 1.2

```
The Tiny profile of SVG 1.2 consists of all of the features defined within this
specification which is widely adopted in the cellphones industry.
```

- Symbol

```
A bitmap or vector image that is used to indicate an object or a particular
property on a map.
```

- Symbolizer

```
It defines the visual representation of geographic objects.
```

- Symbology Encoding

```
It specifies the format of a map-styling language that can be applied to digital
feature and coverage data to produce geo-referenced maps with user-defined styling.
```

- Z-order

```
'Z-order' is an ordering of overlapping two-dimensional objects, such as windows
in a stacking window manager, shapes in a vector graphics editor, or objects in a
3D application (source: Foley et al. (1987)).
```

3.1. 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.

- **2D** 2-Dimensional
- **2.5D** 2.5-Dimensional
- **3D** 3-Dimensional
- **AGC** Army Geospatial Center
- **API** Application Program Interface
- **BLOB** Binary Large OBject
- **CartoCSS** Cartographic Cascading Style Sheets
- **CDS** Concept Development Study
- **CMSS** Cascading Map Style Sheets
- **CRS** Coordinate Reference System
- **CSS** Cascading Style Sheets
- **DISA** Defense Information Systems Agency
- **EPSG** European Petroleum Survey Group
- **Esri** Environmental Systems Research Institute
- **FGDC** Federal Geographic Data Committee
- **FME** Feature Manipulation Engine
- **GeoJSON** Geospatial JavaScript Object Notation
- **GeoSciML** Geoscience Markup Language
- **GIS** Geographic Information System
- **GL** Graphic Library
- **GML** Geography Markup Language
- **IGN** Institut National de l'information geographique et forestiere
- **IT** Information Technology
- **JSON** JavaScript Object Notation
- **MIL-STD-2525** Military Standard 2525
- **NAPSG** National Alliance for Public Safety GIS
- **NAS** NSG Application Schema
- **NGA** National Geospatial-Intelligence Agency
- **NSG** National System for Geospatial Intelligence
- **OGC** Open Geospatial Consortium
- **OSM** Open Street Map

- **OWL** Ontology Web Language
- **OWS** OGC Web Services
- **PNG** Portable Network Graphics
- **POI** Points of Interest
- **QGIS** Quantum Geographic Information System
- **RDF** Resource Description Framework
- **SDP** Spatial Data Pilot
- **SE** Symbology Encoding
- **SIDC** Symbol Identification Codes
- **SLD** Style Layer Descriptor
- **SPARQL** SPARQL Protocol and RDF Query URI Uniform Resource Identifier
- **SQL** Structured Query Language
- **SVG** Scalable Vector Graphics
- **SWG** Standard Working Group
- **TIE** Technology Integration Experiment
- **TDS** Topographic Data Store
- **URI** Unique Resource Identifier
- **URL** Uniform Resource Locator
- **W3C** World Wide Web Consortium
- **WKT** Well Known Text
- **WMS** Web Map Service
- **WMTS** Web Map Tile Service
- **XML** eXtensible Markup Language

Chapter 4. Overview

This Engineering Report (ER) discusses the following sections:

Section 5 introduces the background on "D160 Portrayal Ontology" and extends the portrayal ontology developed in previous testbeds and describes it as a "Portrayal Conceptual Model" to represent a common reference to drive the implementations.

Section 6 describes the demonstration scenarios, technology integration experiments, datasets, guidelines and requirements set by the sponsors.

Section 7 presents the solution developed in this testbed to address "D163 WMTS with Portrayal Support" and implements a Web Map Tile Service (WMTS) instance based on the portrayal conceptual model. In this section, SLD and Cascading Style Sheets (CSS) encodings have been used for styling vector data, and *GeoServer* has been used for rendering.

Section 8 analyses two other encoding formats for styling vector data and rendering them with two platforms based on free and open source solutions: *CARTO online builder* (CartoCSS code generator) and *Mapbox studio* (data oriented with JavaScript Object Notation (JSON) filtering).

Section 9 presents the solution developed in this testbed to address "D162 WMS with Portrayal Support" and implements a WMS instance based on the portrayal conceptual model. In this section, the *CartoCSS* encoding has been used for styling vector data and *Mapnik* has been used for rendering.

Section 10 summarizes the findings, design approaches, and workflows used for "D161 GeoPackage with Portrayal Support" to implement and support the GeoPackage feature portrayal on mobile devices.

Section 11 provides a summary of the main findings and discusses links to forward-looking recommendations.

The following figure (Figure 1) describes the portrayal work items executed in Testbed-14 and how they are mapped to the ER sections.

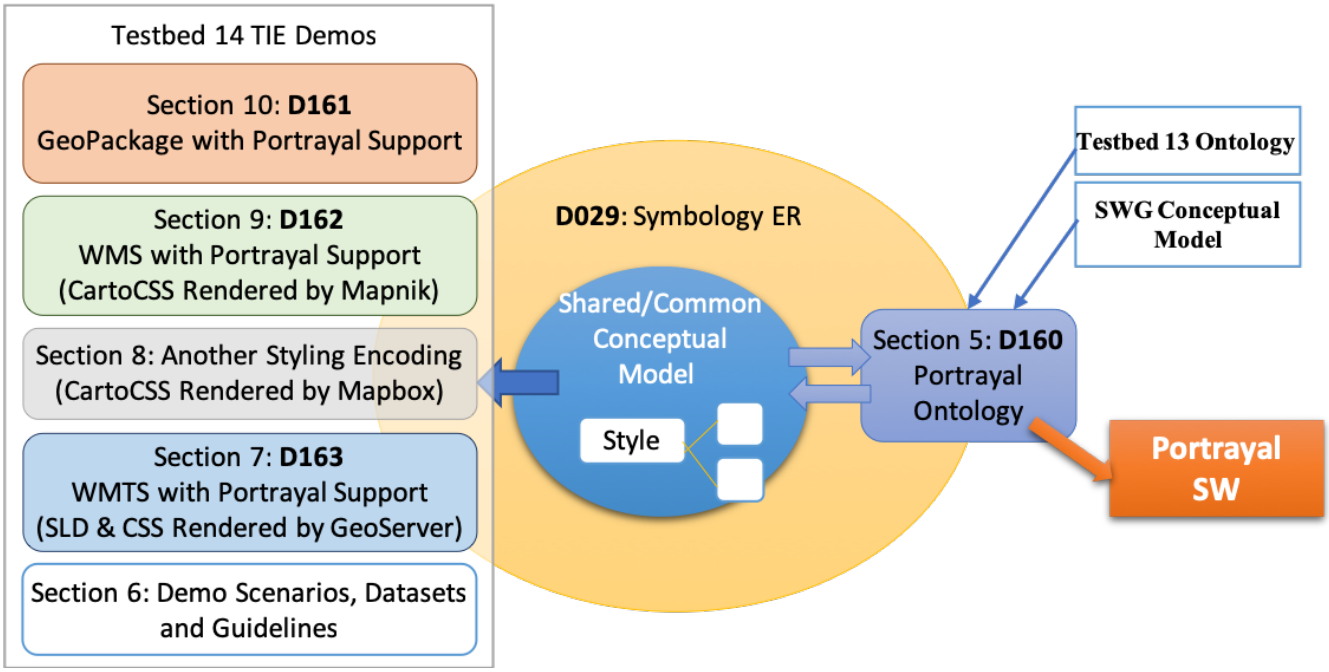


Figure 1. Testbed-14 portrayal thread work items

"Appendix A" presents the portrayal conceptual model expression constructs: Primary Expressions, Identifiers, Operators and Functions which are referred to in Section 5 of this ER.

"Appendix B" provides code snippets that illustrate the encoding of the conceptual styling model as a GNOSIS CMSS (Cascading Map Style Sheets) file which is described in Section 5 of this ER and shall help to implement similar technology.

"Appendix C" and "Appendix D" provide style encodings in GeoCSS and CartoCSS that illustrates the demonstration scenario. These style encodings are described in Section 7 and 9 of this ER and will help to implement similar encoding languages.

"Appendix E" shows the GeoPackage JSON-based Style Encoding for the styles derived from the source encoding that illustrates the demo scenario. This style encoding is described in Section 10 of this ER.

Chapter 5. Portrayal Conceptual Model

This section summarizes the findings, design approaches and changes in work item D160 related to portrayal ontologies.

5.1. Background

The formalization of portrayal ontologies started in OGC Testbed-10, where the focus was on representing point-based symbologies related to Disaster and Emergency Management. The initial implementation of the Semantic Portrayal Service during the OGC Testbed-11 focused on defining styles, portrayal rules, point-based symbols and graphics to enable a Web Processing Service (WPS) to produce an SLD document [3, 4]. The initial ontology was heavily based on the ISO 19117, Geographic Information - Portrayal standard [https://www.iso.org/standard/46226.html].

However, the results from the implementation of the style renderers and graphic ontology development during Testbed 12 proved that ISO 19117 was mostly designed for runtime implementation (for example, use of a portrayal function) rather than for a declarative approach. It was concluded that the OGC 05-077r4, OpenGIS Symbology Encoding Implementation Specification (SE 1.1) [http://portal.opengeospatial.org/files/?artifact_id=16700] standard provides a declarative approach based on eXtensible Markup Language (XML) encoding that is better aligned with modern Application Program Interface (API) renderers (Java Canvas, HTML Canvas, Scalable Vector Graphics (SVG), MapCSS, Environmental Systems Research Institute (Esri) Map Renderer, *etc.*). An update of the portrayal ontologies was then done by introducing a symbolizer microtheory aligned with SE 1.1 [http://portal.opengeospatial.org/files/?artifact_id=16700] and graphic ontology based on SVG constructs. The scope of the portrayal ontologies was limited to vector feature-based representation.

In Testbed-13, the gap between the current portrayal standards (SLD and SE) and the portrayal ontologies developed during Testbed-12 was only identified in the vector data. For this analysis, a round trip conversion from these standards to Linked Data Representation and vice versa was conducted to show that portrayal ontologies were at least as expressive and able to support rendering tasks.

This chapter raises new findings and recommendations that are especially useful for the OGC SLD/SE SWG [http://www.opengeospatial.org/projects/groups/sldse1.2swg] to complement the material on Portrayal Ontology provided by Testbed-13 that targeted the Geosemantics DWG. The focus of Testbed-13 on the subject was on the ability to "share, reuse and mediate unambiguous portrayal information between agencies". The goal then was to make the portrayal ontologies being developed during Testbed 12 as expressive as the current portrayal standards (especially Symbology Encoding, aka SE) and to test its application to Linked Data representation. Therefore, concern for identification was key, allowing for referenceable global unique identifiers to represent the different portrayal information, and making it easier to reuse and link to other artifacts expressed in the other portrayal documents.

Thus, we understand that the motivation of Testbed-13 participants for using "ontology technologies" (*e.g.*, OWL (Ontology Web Language) and RDF) is related to the strong need to "share information on the web" (according to Colin Atkinson, Models versus Ontologies - What is the Difference and where does it Matter? [http://www.cs.bham.ac.uk/~bxb/news/Colin.pdf]). In contrast, the

work for Testbed-14 is dedicated to extending the portrayal conceptual framework and favors the use of "modeling technologies" based on the Unified Modeling Language (UML) for an intuitive and expressive graphical notation of a conceptual styling model. It is relevant because it better reveals some considerations related to rendering aspects – especially when it is about the composition of symbolizers, rendering order and the important requirement to visualize the same cartographic image from one rendering system to another. This does not mean that the formal semantic model (ontology), the set of classes and properties, and the diagrams and tables of Testbed-13 are no longer valid. On the contrary, Testbed-14 takes a simplified view of the Testbed-13 materials in order to better focus on its goals. For instance, *SymbolSet* or *PortrayalRuleList* (among others) continue to be important for the purpose of sharing information with referenceable global unique identifiers but become less important for (re)focusing on for Testbed-14.

On a side note, ontology technologies used in Testbed-13 come with a default encoding, which whilst nice to have, must be put into perspective with the idea of "one conceptual model/many encodings" that might require an "encoding neutral" conceptual framework. It is up to the related SWGs to decide whether to use the "ontology technologies" developed by previous testbeds that would be beneficial for their ability to "share, reuse and mediate unambiguous portrayal information between agencies". Behind the scenes, the hypothesis for Testbed-14 is as follows:

- given a styling document `X` encoded in conformance with the conceptual model, and the related rendering engine producing map `X`;
- given a styling document `Y` encoded in conformance with conceptual model, and the related rendering engine producing map `Y`;
- then, map `X` is "very similar" to map `Y` with a narrow preservation of the cartographic message.

NOTE

A conceptual model in the field of computer science is a special case of a general conceptual model. To distinguish from other types of models, it is also known as a domain model. Conceptual modeling should not be confused with other modeling disciplines such as data modeling, logical modeling and physical modeling. The conceptual model is explicitly chosen to be independent of design or implementation concerns, for example, concurrency or data storage. The aim of a conceptual model is to express the meaning of terms and concepts used by domain experts to discuss the problem, and to find the correct relationships between different concepts. The conceptual model attempts to clarify the meaning of various, usually ambiguous terms, and ensure that problems with different interpretations of the terms and concepts cannot occur. Such differing interpretations could easily cause confusion among stakeholders, especially those responsible for designing and implementing a solution, where the conceptual model provides a key artifact of business understanding and clarity. Once the domain concepts have been modeled, the model becomes a stable basis for subsequent development of applications in the domain. The concepts of the conceptual model can be mapped into physical design or implementation constructs using either manual or automated code generation approaches.

As a consequence, in line with the above note, the goals described in the following are about a portrayal conceptual model with the idea that it be mapped into implementation constructs that are here encoding formats.

5.2. Goals on Portrayal Conceptual Model for Testbed-14

The goals for the work on Testbed-14 D160 reflect common portrayal conceptual framework concerns:

- Update the conceptual model taking into consideration lessons learned from existing; rendering engine implementations and encoding formats such as, CARTO CartoCSS, GeoServer GeoCSS, MapBox GL (Graphic Library) Style, and GNOSIS CMSS;
- Emphasize expressiveness of the conceptual model in relation to symbol composition;
- Make recommendations for developing the ideas of (1) one conceptual model, many encoding formats; (2) core/extensions organization; and (3) basic profile definition.

As Testbed-14 is an extension of Testbed-13, both share top-level elements, namely, a declarative approach for the continuation of Symbology Encoding (SE) and the scope limited to vector based (feature-based) representation.

The intent was also to explore the idea of reusable "symbolizer templates" in relation to past ideas such as the *ParametrizedSymbolizer* of OGC 09-016 [http://portal.opengeospatial.org/files/?artifact_id=33515]. Due to time constraints, this ER is unable to delve deeper into this subject except to point out that the implementations studied do not show any obvious progress in this direction.

Chapter 8, which brings some lessons from CARTO and Mapbox Style underlying encoding formats, and CMSS encoding format appendix of this ER provide some details to support the below findings and recommendations.

5.3. About the Symbolizer Concept

SE 1.1 has a symbolizer to describe how a feature appears on a map and Testbed12 has introduced a symbolizer microtheory as well. Previously Testbed-11 was only able to control point-based symbols and graphics. CartoCSS also uses the term "Symbolizer", but it carries a different meaning.

SE defines four symbolizers (for vector-based features) which are: Line, Polygon, Point and Text. The *PolygonSymbolizer* describes how a polygon geometry type feature appears on a map, based on styling parameters to both stroke the polygon outline AND fill it. Whereas the CartoCSS Polygon Symbolizer only allows control over filling the polygon, there is a Line symbolizer for controlling the outline of the polygon, and thus by extension everything with a linear part in its geometry to be drawn. This is applicable also for line type geometry.

Following the CartoCSS approach:

- The Polygon symbolizer is used to Fill the area part of a geometric shape, specifically the interior of a polygon;
- The Line symbolizer is used to Stroke the linear part of a geometric shape, which is the outline of a polygon or a line;
- The Point and Marker symbolizers are for marking a point geometric shape which may result from the transformation of a polygon or line into a point (*e.g.*, centroid or vertices);

- The Text symbolizer is for labeling point geometric shapes which may result from a similar transformation.

Therefore, the Testbed-14 conceptual styling model offers four categories of symbolizers, to Fill, Stroke, Mark and Label. The model defines the four classes as abstract concepts representing points of extension to define concrete ways to Fill, Stroke, Mark and Label.

An interesting point to note is that OGC 09-016, OWS-6 Symbology Encoding (SE) Changes ER [http://portal.opengeospatial.org/files/?artifact_id=33515] suggested renaming *PolygonSymbolizer* to *AreaSymbolizer*, probably with the idea in mind of dedicating it to control how to fill the area of a geometric shape. However, the OGC 09-016 [http://portal.opengeospatial.org/files/?artifact_id=33515] did not elaborate on the idea (*i.e.*, the *AreaSymbolizer* still described how to stroke AND how to fill).

Also, a symbolizer may or may not apply depending on the geometric type, so it is up to the standard to specify the default behavior for applying each type of symbolizer to each type of geometric shape. For example, "What happens when a Fill symbolizer is applied to a point geometric shape?", or "What happens when a Mark symbolizer is applied to a multipoint geometric shape?"

TIP	Our recommendation is to disable any default behavior for the application of a Fill symbolizer for line or point geometric types, and to do the same with a Stroke symbolizer for point geometric types.

NOTE	The Text symbolizer is the only symbolizer that can move the top label to avoid overlapping, whereas the Marker symbolizer always shows a marker at the exact visualization point, and may thus hide, but not move, the marker when overlapping occurs.

5.4. A Symbolizer to Compose

Testbed-13 introduced the concept of the *CompositeSymbolizer* which is especially, but not only, useful when a geometric shape has to be drawn multiple times with the same symbolizers (*e.g.*, marking a point geometry twice). Given the multiple cardinality from Rule to Symbolizer, having some styling parameters of the same type of symbolizer multiple times in a Rule does not mean drawing multiple times. Rather, the overriding mechanism happens in such instances (see next section). Thus, a symbolizer to control the rendering of a composition is required to finely control the drawing of composite symbols. The model conceptually reveals the importance of the ordering of the components of the composition. Also, each component has a name and pass type to define its membership for a particular symbolizer. The default type is a simple pass which means a composition case in which each feature is drawn multiple times in the same layer's rendering pass.

As an example of encoding using CartoCSS, a simple pass composition with two mark symbolizers is presented:

- Each styling parameter is prefixed with a symbolizer name to define its membership in a particular symbolizer (using the separator slash);
- The natural sequence of instructions defines the order of rendering (from top to bottom; with *marksymbolizer1* first, then *marksymbolizer2*)

```
#station {
  marksymbolizer1/marker-width: 40;
  marksymbolizer1/marker-type: ellipse;
  marksymbolizer1/marker-fill: #000000;
  marksymbolizer1/marker-allow-overlap: true;

  marksymbolizer2/marker-width: 30;
  marksymbolizer2/marker-type: arrow;
  marksymbolizer2/marker-fill: #FFFFFF;
  marksymbolizer2/marker-allow-overlap: true;
  marksymbolizer2/marker-transform: rotate(-45);
}
```

The second pass type is said to be a multiple-pass, which refers to a composition case in which each layer is drawn multiple times in separate rendering passes.

TIP

This ability is quite common in cartography software:

- CartoCSS introduces it;
- GeoServer GeoCSS uses a `z-index` property, which maps back to multiple SE 1.1 [http://portal.opengeospatial.org/files/?artifact_id=16700] *-FeatureTypeStyle*, each one painted as a separate rendering pass;
- It is also found in QGIS (Quantum Geographic Information System) under the name "symbol levels".

As an example of encoding using CartoCSS, a multiple-pass composition with three stroke symbolizers (old symbology of the French Institut National de linformation geographique et forestiere (IGN) "irregularly maintained narrow road") is presented:

- Each symbolizer has a prefix name starting with a double colon to identify the group of styling parameters affected by a new pass symbolizer (in CartoCSS, the separator slash stands for simple pass compositions, whilst the double colon stands for multiple-pass compositions);
- The natural sequence of instructions defines the order of rendering – from top to bottom, with a pass for *strokeleft* first, followed by a pass for *strokerightdash*, before finally a pass for *strokemiddle*.

```
#ignroads {
  ::strokeleft {
    line-color: #000000;
    line-width: 2;
    line-offset: -5;
  }

  ::strokerightdash {
    line-color: #000000;
    line-width: 2;
    line-dasharray: 15,10;
    line-offset: 5;
  }

  ::strokemiddle {
    line-color: #FFFFFF;
    line-width: 10;
  }
}
```

Note that a style has a default rendering pass applying all the defined rules according to the rule interpretation algorithm. Thus, each insertion of a multiple-pass composition into a rule allows control over other extra relevant passes. As illustrated by the above example, multiple-pass composition is especially useful for drawing connected roads (also called casing, or roads with a border).

Note that the line casing use case, for example, could be considered on its own for its specific purpose (with properties specific to casing for the PenStroke). The rationale to define line casing specifically without resorting to multiple passes is two-fold. First, in hardware accelerated real-time visualization clients, passes can be very costly, and specific use cases can benefit from such a "fast-track" symbolizer, which do not require the level of flexibility that can only be defined through multiple passes. Secondly, there is a conceptual advantage to having the model correspond more directly to how someone thinks about the style being described, in terms of expressiveness, ease of making modifications, facilitation of conversion between encodings, but sometimes it can also facilitate the renderer's task to produce the desired result. For line casing in particular, the actual width of the casing is what is specified, as opposed to specifying a width larger than the inner line in order to get the desired thickness for the casing. This also enables specifying the casing in pixel sizes, while the size of the line itself is specified in real-world units. This in turn enables better visual in 3D views where the ratio between pixels and real-world distance varies considerably.

Certain situations may require these symbolizers intended for a specific purpose, either for performance reasons or when it becomes too complex to compose a specific symbol (*e.g.*, Military Standard 2525 for military map symbols or windbarbs for meteorologists). Cascadenik (Cascadenik implements cascading stylesheets for Mapnik) does the casing by extending its stroke symbolizer with outline parameters (see https://github.com/mapnik/Cascadenik/wiki/Dictionary#outline). The same applies to specific parameters for adding a center line (https://github.com/mapnik/Cascadenik/wiki/Dictionary#inline). Nevertheless, according to Tom MacWright, Cascadenik now supports multiple-pass drawings, which are called attachments (see https://macwright.org/2012/11/02/css-for-

maps.html).

5.5. About the Importance of Ordering

The model reveals some ordering constraints that have an impact on the final visual aspect of the symbols:

- **Style → Rule is {ordered}**

A Style is built on a list of styling rules. The order of the rules matters in relation to the overriding mechanism, *i.e.*, the order in which visual properties are being applied, so that if the same visual property is defined again from a rule later in the list, it is overridden by the new value. This means that repeating a rule with different values for the same visual properties does not result in different things being rendered the second time round.

With an example of encoding using CartoCSS considering both situations #1 and #2, when both rules with a rule condition have the same specificity, a feature that falls into both of these rule conditions does not look like the same thing.

Situation #1

```
#layer {
  line-color:#3E7BB6;
  line-width:3;
}

#layer[name = 'highway'] {
  line-width:6;
  line-dasharray: 10,5;
}

#layer[num = 10] {
  line-width: 1;
}
```

Situation #2

```
#layer {
  line-color:#3E7BB6;
  line-width:3;
}

#layer[num = 10] {
  line-width: 1;
}

#layer[name = 'highway'] {
  line-width:6;
  line-dasharray: 10,5;
}
```

Notice that this also applies to the Testbed-13 *PortrayalRuleList* concept as it is a list with a certain order.

- **Symbol → Symbolizer is {ordered}**

Generally, CSS adds styles to a document model, whereas for maps, we tend to use the painter's model which gives importance to ordering. This affects how ordering in the language works. Indeed, from the example of encoding with CartoCSS, both rules below do not produce the same results (in other words, `fill + stroke <> stroke + fill`):

```
#layer {
  line-width: 6;
  polygon-fill: #aec;
  polygon-opacity: 0.8;
}

#layer {
  polygon-fill: #aec;
  polygon-opacity: 0.8;
  line-width: 6;
}
```

Note that we may also use the SVG paint order (see https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/paint-order) by defining normal behavior with the fill painted first, then the stroke, then the markers with the labels always on top (even having their own rendering pass). Specific parameters would then be required to specify a different paint order.

- **Compose → Symbolizer is {ordered}**

When doing composition, the order is obviously important, with the rendering pass being simple or multiple. Using opacity to simulate the visibility of a component behind the scene is not a feasible option. Instead, we can play with the ordering. In the above #station example, the marksymbolizer1 is drawn before marksymbolizer2, with the intent of having the first one in background.

5.5.1. Future Work on Ordering

GeoServer also recognizes another kind of ordering issue called the *sortBy Z-ordering* (see http://docs.geoserver.org/stable/en/user/styling/sld/extensions/z-order/syntax.html). This has an impact on the ordering of the flow of features coming from the layer to visualize. The idea of such parameters for a rule is similar to the `ORDER BY` part of an Structured Query Language (SQL) query. Such an ability is relevant for different use cases:

- The OpenStreetMap dataset extensively uses a *Z_order* attribute to model the above/below relationships between elements in the real world;
- In thematic mapping, when proportional symbol representation is used, *e.g.*, on a population attribute, features may need to be sorted by the specific attribute controlling the variation of the symbol size (avoiding the problem of bigger symbols hiding smaller ones).

QGIS introduces a layer rendering ability called "Control feature rendering order". CARTO with CartoCSS goes further, allowing for control through an explicit preliminary step of using a complete SQL query to select the features to render, before reordering them. The OSM Carto Editing Guidelines (https://ircama.github.io/osm-carto-tutorials/editing-guidelines/) clearly show the complexity of this preliminary step for preparation of the geodata flow.

The following questions needs to be answered:

- How desirable is it to have abilities in the cartographic language itself to control the ordering?
- Is it a separate responsibility not belonging to the styling conceptual model?

Due to time constraints, these types of ordering issues will require further investigations in future work.

5.6. Rule Interpretation with Cascading

Given the popularity, power and simplicity of the World Wide Web Consortium (W3C) Cascading Style Sheets (CSS) as a language for adding style (*e.g.*, fonts, colors, and spacing) to Web documents, it is worth considering some of the underlying CSS concepts for inspiring a language for the presentation of geospatial data. Specifically, the fundamental concept of cascading that is used by CartoCSS and Mapbox styles as well as GL Style, GeoCSS, and Cascadenik. Cascading may actually reduce the number of rules for complex styles such as the "Circumpolar Thermokarst Landscape" from the Arctic SDP (Spatial Data Pilot) project (see http://ogc.standardstracker.org/show_request.cgi?id=519).

Currently, SE 1.1 interprets multiple rules for rendering features multiple times. The idea of cascading is different, rather than overriding one rule over another. The cascading focuses on how something should be styled and refers to an algorithm finding the styling parameters to apply for each feature (from http://docs.geoserver.org/stable/en/user/styling/css/cascading.html):

- By locating all the rules with rule conditions matching the current feature;
- Sorting them by specificity (*e.g.*, number of attributes in a rule condition), from less specific to more specific;
- Having more specific rules that add to and override styling parameters with less specific rules.

Finally, depending on the feature attributes, a specific rendering rule is built for the feature which mixes all the applicable rules. By setting the common bits with less specific rules, and overriding them by specifying the exceptions to the norm with more specific rules, the core of the algorithm allows for the preparation of succinct style sheets for otherwise very complex rule sets.

Consequently, rather than the SE 1.1 [http://portal.opengeospatial.org/files/?artifact_id=16700] *ElseFilter* concept, one can simply have a first Rule without any Filter. Note that the overriding follows the composition prefixes.

Lastly, cascading for maps is often associated with nesting for better cartographic code structuring. However, we suggest that this ability not be part of the conceptual view, but instead be a responsibility of the encoding. As an example using CartoCSS, this encoding with nesting is similar to the above situation #1.

```
#layer {
  line-color:#3E7BB6;
  line-width:3;

  [name = 'highway'] {
    line-width:6;
,line-dasharray: 10,5;
  }

  [num = 10] {
  ,line-width: 1;
  }
}
```

Hence, one could think that cascading has only benefits and that cascading is the core of rule interpretation. Nonetheless, we may moderate such a radical position. Indeed, having cascading at the core could lead to a concept difficult to understand:

- Cartographers who are not coming from a web background find it difficult to relate to cascading overriding;
- The related concept of specificity to compute (inherited from W3C CSS [https://www.w3.org/standards/webdesign/htmlcss]) can be complex and users may have difficulty "to run it in their heads", as (see https://docs.geoserver.org/stable/en/user/styling/css/cascading.html)

We may add the fact that automatic legend generation then becomes complex, as all possible cases must be generated, and some might not be relevant to the cartography at hand. Also, it is difficult to control order of symbol appearance in the legend (probably the reason why separate tooling to do legends are used in some setups).

Therefore, we may find it instead easier to have a simplified version of CSS with no cascading, but with eventual rule nesting for expressing localized overrides (Cascadenik like, also implemented by GeoCSS). Such interesting contributions in terms of formatting may solve most of the above issues.

As a consequence, while cascading is powerful, its usage is probably best left as an option for the cartographer to choose from- or provide a way to turn the cascading feature off.

5.7. Styling Model

This section describes some details of the styling model elaborated during Testbed-14 (Figure 2). This model complements previous Testbed results and therefore, it is important to be read in relation to the ontologies described especially in the Testbed-13 Portrayal ER [5]. In other words, some elements of understanding are not repeated in this section and, they are described with Testbed-13 (*e.g.*, tables).

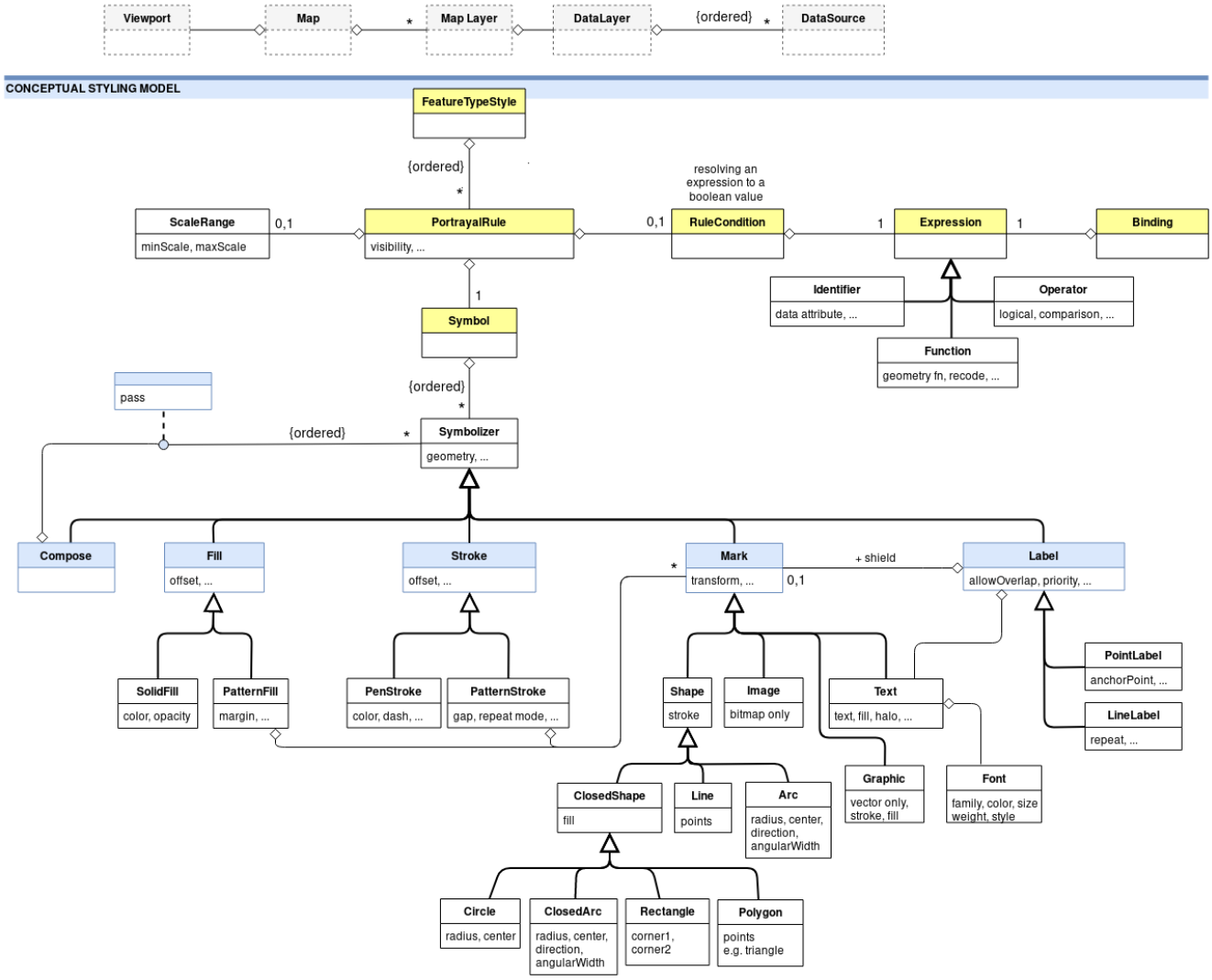


Figure 2. UML diagram of Testbed-14 styling model

5.7.1. FeatureTypeStyle Modeling and Vector-based (Feature-based) Representation

A *FeatureTypeStyle* describes the styling instructions to apply on features coming from a DataLayer so as to render a MapLayer as an output. The input is a DataLayer built of an ordered list of DataSource each containing a single feature type. As an example, given *DataSource A* has linear feature for roads, *DataSource B* has polygon features for urban areas, *DataSource C* has point features for gas stations, and *DataLayer Z* built of `{B, A, C}` gives the rendering engine to first process *B*, then *A*, then *C*.

This is in line with what is mentioned in Testbed-13 (§ 6.3.5):

```
"The OGC SLD standard defines the concept of "layer" as a collection of features
that can be potentially of various mixed feature types."
```

and also:

```
"The WMS interface uses the LAYER parameter to reference named layers as in the
example parameter: `LAYERS = Rivers, Roads, Houses`".
```

NOTE	The ordering of the list has importance; this is the behavior of MapBox (old Mapbox Studio for desktop) and CARTO (online editor) with CartoCSS.

Also, SE 1.1 [http://portal.opengeospatial.org/files/?artifact_id=16700] defines the *FeatureTypeStyle* concept as the styling that is to be applied to a single feature type. The *FeatureTypeName* property allows identifying the specific feature type that the feature-type style is for. In other words, the idea for Testbed-13 and Testbed-14 is that a *FeatureTypeStyle* is associated to a specific feature type that has to be identified in the case of a DataLayer with many feature types (in "Table 4" of the Testbed-13 Portrayal ER [5], this is presented by the featureType table attribute).

In consequence, the present styling model shares a common understanding of the Style concept being a central concept and especially the *FeatureTypeStyle* concept being a style applied for a specific FeatureType. Similar to Testbed-13, the scope of the Testbed-14 is limited to vector-based (feature-based) representation.

5.7.2. PortrayalRule

The main difference with Testbed-13 is related to the interpretation by the rendering engine of a set of rules (see the Testbed-13 Portrayal ER [5] chapter *Rule interpretation with cascading*).

Table 1. Properties of PortrayalRule

Name	Definition	Data type and value	Multiplicity
ruleCondition	An Optional set of Selectors which determine whether the rule should be applied or not	RuleCondition	0..1
scaleRange	Scale range to which the rule applies	ScaleRange	0..1
visibility			0..1
symbol	The symbol associated with the rule	Symbol	0..1

TIP	*Z-order* may be added as an optional integer typed property to specify the **visual rendering priority** of elements within a single layer or the ordering of the layers themselves

5.7.3. ScaleRange

The *ScaleRange* concept is just an extraction of the properties related to the special filter allowing to restrict the styling of a *PortrayalRule* to a certain range of map scale.

Table 2. Properties of ScaleRange

Name	Definition	Data type and value	Multiplicity
minScale	The minimum scale denominator to which the rule applies	Integer	0..1
maxScale	The maximum scale denominator to which the rule applies	Integer	0..1

5.7.4. RuleCondition

RuleCondition is similar to the concept presented in the Testbed-13 Portrayal ER [5]. Like ScaleRange, it is a concept used by PortrayalRule to "define conditions of application of associated symbolizers" (from OGC 08-067, Symbology Conceptual Core Model [http://www.opengeospatial.org/pressroom/pressreleases/2893]), hereby related to feature attributes. In other words, it is to identify a subset of features from a collection of features whose property values satisfy a set of logically connected predicates (constraint according to the Testbed-13 Portrayal ER [5]). If the attribute values of a feature satisfy all the predicates, then that feature is considered to be part of the resulting subset a given Symbol.

Table 3. Properties of RuleCondition

Name	Definition	Data type and value	Multiplicity
expression	Constraint defining the condition of application of the rule in relation to attributes of the feature type data model (similar to the WHERE clause of a SQL statement)	Expression	1

Discussion: Whether to specify the *scaleRange* separately from the *ruleCondition* is still being debated. In an earlier version of the conceptual model, the min and max scale range specifiers were simply defined as comparison expressions between constant numerical values, and the view scale denominator, that generic expressions allowed to define (see also section on Expressions below). The advantage is that the *ruleCondition* then completely encompasses whether a rule should be applied or not. A potential reason for separating it is to somehow highlight the scale condition as opposed to other selectors, but the encoding could still opt for doing this while keeping it within the *ruleCondition* for the conceptual model. The concept of expressions is to have a consistent generic approach, which would apply as well to rules conditional on time range for example; or allow the comparison of attribute values in comparison with the current view scale.

5.7.5. Expression and Binding

The concept of "Expression" allows different types of values, both static and dynamic, that can

represent feature attributes, geometry, layer identifier, properties of the current view such as scale, visualization time, and more. In addition to primary expression constructs, more complex expressions can be built through operators evaluating a relation between operands, performing arithmetic, function calls and more. Depending on the expressions complexity supported, expressions can offer a level of flexibility almost comparable to a miniature programming language.

Testbed-13 adopted an approach with expressions as literals and associated to a standard URI (Unique Resource Identifier) for the query language of the expression (OGC Filter or SPARQL (a recursive acronym for SPARQL Protocol and Resource Description Framework Query Language)). Also, Testbed-13 introduced the idea of "Binding" which is a good conceptual definition of the binding to symbolizer attributes.

In the conceptual model introduced for Testbed-14, expressions are used both for specifying the conditions for a rule to apply, as well as for specifying any value for a given style or symbolizer property. When wondering if a symbolizer attribute can be manipulated, it is generally just a matter of time to find a use case to confirm that yes, it has to be (notice it is also the approach of desktop tools like QGIS).

Moving away from an "encoding approach", the model may better try to specify the extent of the abilities. We can distinguish between different types of primary expression constructs: Literals (textual, numeric, lists), Identifiers, Variables (application controlled), Operations (with Function calls as a particular type of operation) (see the Conceptual Model Expressions appendix of this ER).

While pre-built functions used in portrayal are required (text formatting functions, transformation functions already available in SLD/SE such as recode, categorize, interpolate, *etc.*); the expression model stays extensible, allowing the definition of advanced portrayal profiles with an enriched set of functions (geometry manipulation functions like start/end of line, *etc.*). See also GeoServer filter function reference: http://docs.geoserver.org/latest/en/user/filter/function_reference.html

5.7.6. Symbol and Symbolizer Extension Points

Introducing a Symbol concept between the *PortrayalRule* and Symbolizer concepts, as suggested by the Testbed-13 Portrayal ER [5] symbolizer microtheory, makes sense within the context of the http://docs.opengeospatial.org/per/17-045.html#*a_2_1_4_portrayalruleset[Testbed-13 Portrayal ER] (B.12.4.4): "Allow the retrieval of a Symbol instance in the frame of a Semantic Portrayal Registry. As a consequence, it is the Symbol that holds the various Symbolizers to draw to create a cartographic representation (and not the Rule as it is for SE 1.1). That is why the cardinality of a _PortrayalRule* has one mandatory Symbol."

In other words, a Symbol instance "describes how a feature is to appear on a map" utilizing the concept of Symbolizer which details the various styling information to apply given a drawing order.

Testbed-13 states that a "Symbolizer is obtained by specifying one of the sub-classes of Symbolizer and then supplying parameters to override its default behavior". Testbed-14 keeps as a base this vision of the Symbolizer concept coming from SE 1.1 as a top class of all the possible types of symbolizers and provides properties that are inherited by all. Also, all these types of symbolizers also share the multiple common cardinalities with the Symbol concept.

Nonetheless, in Testbed-14 the possible types are seen as categories of symbolizers, adding an intermediate level of specialization. For instance, as opposed to SE 1.1 se:PolygonSymbolizer, the Fill class inherits from Symbolizer but is also defined as the parent class of all the different and conceivable types of area filling abilities. As a consequence, a container does not define how to draw the linear border of an area which rather a concern of one Stroke class specialization (versus se:PolygonSymbolizer which did include this information). Notice, this may call into question the logic of separation between Symbolizer ontology and Graphic ontology defined in Testbed-13.

Finally, there are two levels of extension points: Either at the Symbolizer top level to define a new category of symbolizers (*e.g.*, might be for raster data) or at the intermediate level to define specializations belonging to a given category because of a common styling intent (Fill is to fill, Stroke is to stroke, Label is to label with deconfliction, *etc.*).

As a consequence, from the Testbed-13 Portrayal ER [5] symbolizer microtheory, while we keep the Symbolizer concept definition (name, title, description, unit of measure, list of bindings to apply), we redefine the sub-classes as presented in the following section.

Fill Extension Point

The *Fill* abstract class is the parent class of all the different and conceivable types of area filling abilities (*e.g.*, to draw a polygon-typed feature geometry or any shape with an area to fill).

Table 4. Properties of Fill class

Name	Definition	Data type and value	Multiplicity
geometry	Used to specify the geometric attribute from the data model associated with the symbolizer or process each geometry before drawing (e.g. ST_Buffer)	Geometry	0..1
labelObstacle	If true the drawn area will be considered an obstacle for labels so no label will overlap it	Boolean	0..1

Fill behavior according to geometry type (may be redefined by Fill sub-types):

- (multi)point feature: does not apply.
- (multi)line feature: does not apply.
- (multi)polygon feature: relative to the inside surface of the polygon (all polygons when multiple geometries).

Stroke Extension Point

The *Stroke* abstract class is the parent class of all the different and conceivable types of linear drawing abilities (*e.g.*, to draw a line-typed feature geometry or any geometric line).

Table 5. Properties of Stroke class

Name	Definition	Data type and value	Multiplicity
geometry	Used to specify the geometric attribute from the data model associated with the symbolizer or process each geometry before drawing (e.g. ST_EndPoint)	Geometry	0..1
offset	Offset distance between original geometry and the drawn line stays equal	Decimal	0..1
labelObstacle	If true the drawn line will be considered an obstacle for labels so no label will overlap it	Boolean	0..1

Stroke behavior according to geometry type (may be redefined by Stroke sub-types):

- (multi)point feature: does not apply.
- (multi)line feature: relative to the line (all lines when multiple geometries).
- (multi)polygon feature: relative to the outline of the polygon (all polygons when multiple geometries).

Mark Extension Point

The *Mark* abstract class is the parent class of all the different and conceivable types of point marking/drawing abilities (*e.g.*, to draw a point-typed feature geometry or to hang a mark on a geometric point).

Table 6. Properties of Mark class

Name	Definition	Data type and value	Multiplicity
geometry	Used to specify the geometric attribute from the data model associated with the symbolizer or process each geometry before drawing (e.g. ST_GeometricMedian)	Geometry	0..1
transform	Defines a list of transform definitions that are applied (among Translate / Scale / Rotate according to SVG transform definitions)	Transform	0/*
opacity	Defines the level of opacity to apply when rendering the mark	Decimal	0..1

Name	Definition	Data type and value	Multiplicity
labelObstacle	If true the drawn point symbol will be considered an obstacle for labels so no label will overlap it	Boolean	0..1

Mark behavior according to geometry type (may be redefined by Mark sub-types):

- (multi)point feature: relative to the point (all points when multiple geometries)
- (multi)line feature: relative to a centroid of the geometry or any similar representative point (set of centroids when multiple geometries)
- (multi)polygon feature: same as previous

Label Extension Point

The *Label* abstract class is the parent class of all the different and conceivable types of feature labeling abilities. One main ability is related to the control of deconfliction resolution to avoid the overlapping of labels between them and with other features (see also labelObstacle properties of the other symbolizers). In other words, using Label symbolizers may hide the labeling of some geometries in some circumstances, whereas using Mark symbolizers guarantees the display of the marking of all geometries to be rendered.

Table 7. Properties of Label class

Name	Definition	Data type and value	Multiplicity
geometry	Used to specify the geometric attribute from the data model associated with the symbolizer or process each geometry before drawing (e.g. ST_GeometricMedian)	Geometry	0..1
deconflict	If true the labelling is performed according to a deconfliction algorithm	Boolean	0..1
priority	Specifies an expression to use in determining which features to prefer if there are labelling conflicts	Expression	0..1
textElement	Defines the visual properties of the label to write on the map	Text	1
shield	A mark to display behind the label such as a highway shield	Mark	0..1

Label behavior according to geometry type is similar to the Mark behavior (may be redefined by Label sub-types).

5.7.7. Composition of Symbolizers

The idea of symbolizers composition allows drawing each feature several times according to different symbolizers to create a symbol in itself. Often, it is used to compose two descriptions of the same symbolizer type. As an example, the Harmonica index is built of two Mark symbolizers, a triangle over a rectangle (https://www.bruitparif.fr/l-indice-harmonica). Another example, to draw an alternation of a dash (PenStroke) and symbol (PatternStroke) on a line feature (https://docs.geoserver.org/stable/en/user/styling/css/cookbook/line.html#alternating-symbols-with-dash-offsets).

Notice that there is a given order which is essential to consider, a different order producing a different rendering.

Table 8. Properties of symbolizers composition

Name	Definition	Data type and value	Multiplicity
items	List of symbolizers applied according to a given order (each feature is drawn multiple times even for repeating several times a same type of symbolizer)	Symbolizer	2/*
compositing	Composition operator (e.g. src-over / dest-out / dest-over)	String	0..1

For each item, a **pass level** may be used to define the belonging of the Symbolizer to a separate rendering pass. This means each layer is drawn multiple times (versus each feature is drawn several times). For more details, please refer to section "A symbolizer to compose".

Table 9. Properties of Pass

Name	Definition	Data type and value	Multiplicity
pass	Define the belonging of the Symbolizer to a rendering pass (zero is the default rendering pass)	Integer	0..1

5.7.8. Styling Model Fill Extensions

SolidFill

The *SolidFill* class is a concrete implementation of the Fill symbolizer class and allows to formulate an area filling according to color and its opacity.

Table 10. Properties of the SolidFill class

Name	Definition	Data type and value	Multiplicity
color	Represents a color in a given color space	Color	0..1

Name	Definition	Data type and value	Multiplicity
opacity	Represents a percent of opacity	Decimal	0..1

PatternFill

The *PatternFill* class is a concrete implementation of the Fill symbolizer class. This class repeats a mark according to a rectangular tiling pattern over an area (*i.e.*, mosaic). A mark can be defined very informally as "a little picture". The appearance of the mark is defined according to the related mark sub-type to be used.

Table 11. Properties of the PatternFill class

Name	Definition	Data type and value	Multiplicity
mark	The mark to repeat within the area	Mark	1
tileGapX	Define the X gap of an empty space to put between two consecutive tile	Decimal	0..1
tileGapY	Define the Y gap of an empty space to put between two consecutive tile	Decimal	0..1

5.7.9. Styling Model Stroke Extensions

PenStroke

The *PenStroke* class is a concrete implementation of the Stroke class. This class defines drawing a line analogously to how a pen is used with ink. That is to say by filling the area formed by the thickness of the line with a color and optionally considering a pattern of dashes and gaps.

Table 12. Properties of the PenStroke class

Name	Definition	Data type and value	Multiplicity
color	Represents a color in a given color space	Color	0..1
opacity	Represents a percent of opacity	Decimal	0..1
width	Thickness of the line which gives form to an area to fill with the given color	Decimal	0..1
linecap	Similar to the SVG definition https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linecap	String	0..1

Name	Definition	Data type and value	Multiplicity
linejoin	Similar to the SVG definition https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-linejoin	String	0..1
dashArray	Defining the pattern of dashes and gaps used to draw a linear part	List of Decimal	0..1
dashOffset	Defining an offset on the rendering of the associated dash array (shifting the start of the dash array computation)	Decimal	0..1

We may also introduce a StippleStroke case to fill the area formed by the thickness of the line with the repetition of a Mark, similar to PatternFill (to be done, evaluate the relevancy of the term stipple introduced by OGC 09-016 [http://portal.opengeospatial.org/files/?artifact_id=33515]).

PatternStroke

The *PatternStroke* class is a concrete implementation of the Stroke class. This class is about one graphic repeated along a line. This is very similar to the SE 1.1 definition of GraphicStroke. The Length property specifies the linear length to reserve along the line for one of the repeated marks to plot. By default, the linear length is equal to the Mark natural length (which depends on the view box of the Mark). This has the effect to perfectly juxtapose the repeated marks all along the line. Zero-length value means the mark is not repeated within all the available linear space. Notice that a single Mark is not plotted if the size of the available linear space smaller than the Length.

Table 13. Properties of the PatternStroke class

Name	Definition	Data type and value	Multiplicity
mark	The mark to plot along the line	Graphic data type	1
length	Linear length to reserve along the line to plot a single mark (by default it is the mark natural length)	Decimal	0..1

NOTE A renderer MAY apply some aesthetic embellishments like trying to bend a graphic around corners or avoid a graphic to be cut at the start or at the end.

5.7.10. Styling Model Mark Extensions

On many points, the conceptual model defines these mark extensions in line with the Testbed-13 Portrayal ER [5] Graphic Object Hierarchy.

Image

The *Image* class is a concrete implementation of the Mark class. This is the simplest point marking ability, by referencing a bitmap image resource (*e.g.*, portable network graphics (PNG)).

Table 14. Properties of the Image class

Name	Definition	Data type and value	Multiplicity
resource	Reference to resource from which a bitmap image can be obtained	String	1
format	Expected content format of a successful fetch (MIME type)	String	1
width	Indicates the width of the image	Decimal	0..1
height	Indicates the height of the image	Decimal	0..1

Graphic

The *Graphic* class is a concrete implementation of the Mark class. This class allows the referencing of a vector image resource (*e.g.*, in particular SVG). The main difference from the Image class is related to the optional ability to override the styling properties of the linear and area parts inside the graphic. By default the properties defined inside are used and 100% black if no inside properties are defined.

Table 15. Properties of the Graphic class

Name	Definition	Data type and value	Multiplicity
resource	Reference to resource from which a graphic can be obtained	String	1
format	Expected content format of a successful fetch (MIME type)	String	1
width	Indicates the width of the graphic	Decimal	0..1
height	Indicates the height of the graphic	Decimal	0..1
stroke	Override the styling properties of the linear part inside the graphic	PenStroke	0..1
fill	Override the styling properties of the area part inside the graphic	SolidFill	0..1

The overriding of the stroke and fill styling properties is inspired from CartoCSS which makes this difference between what is called the *Point* and the *Markers* symbolizers. The latter offers *marker-line-color*, *marker-line-width*, *marker-line-opacity*, *marker-fill*, *marker-fill-opacity* visual properties as *PenStroke* and *SolidFill* offers in this conceptual model.

Shape

The *Shape* abstract class extends the Mark class to allow to define a geometric figure to use a customized shape marker. There is one common property to all markers of this type, the stroke so as to draw the linear part of the shape, being it an open or closed shape.

Table 16. Properties of the Shape class

Name	Definition	Data type and value	Multiplicity
`stroke`	`The stroke visual properties to use to draw the linear part of the shape`	`PenStroke`	`1`

Notice, as described in Testbed-13, we consider the following sub-types of Shape: ClosedShape, Line and Arc. The intent here is to conceptually define these sub-types, whereas Testbed-13 proposed an "encoding approach" to describe the geometry of the shape (*e.g.*, Scalable Vector Graphics (SVG) path definition or Well Known Text (WKT), Geography Markup Language (GML)):

- **Line** concrete sub-type is defined according to a list of connected points that creates straight lines (in comparison to SVG, it is similar to the polyline element, https://developer.mozilla.org/en-US/docs/Web/SVG/Element/polyline).
- **Arc** concrete sub-type is defined according to X/Y radius, center, direction and angular range (*e.g.*, as defined by https://www.telerik.com/kendo-angular-ui/components/drawing/api/geometry/Arc/).
- **ClosedShape** abstract sub-type: see below.

We may argue it is important to keep the SE 1.1 well-known names, but such an ability is similar to referencing a registry of predefined shapes to avoid specifying the points forming a little square every time such a simple shape is used. Precisely, a portrayal registry is the kind of ability that is described in Testbed-13.

ClosedShape

The *ClosedShape* abstract class extends the Shape class as one of the types of Marker which allows to use a customized and especially closed shape marker to hang on the reference point to draw. Therefore, there is one more common property to all markers of this type, the fill so as to draw the area part of the shape, because it is closed.

Table 17. Properties of the ClosedShape class

Name	Definition	Data type and value	Multiplicity
`fill`	`The fill visual properties used to draw the area part of the shape`	`SolidFill`	`1`

The intent here is to define conceptually only the concrete sub-types to describe closed shapes (in comparison to Testbed-13, Circle and Polygon are concerned, whereas Line is rather a "simple" Shape):

- **Circle**: may be defined according to center and radius

- **Rectangle**: may be defined according to two corner points
- **Polygon**: may be defined according to a list of connected points and joining start and end points
- **ClosedArc**: may be defined similar to the arc definition while joining start and end points

Text

The *Text* class is a concrete implementation of the Mark class. It allows to defines a mark consisting of a text built of one or many characters. It may be used to reference a specific glyph in a font.

Table 18. Properties of the Text class

Name	Definition	Data type and value	Multiplicity
`textLabel`	`Text label associated with the symbolizer`	`Expression`	`1`
`font`	`Font visual properties`	`Font`	`1`
`halo`	`Halo visual properties applied to the background of font glyphs (color or radius)`	`Halo`	`0..1`

Shortly, Font and Halo definition is as follow:

- **Font**: may be defined according to font family, color, font style, weight and size
- **Halo**: may be defined according to radius and color

5.7.11. Styling Model Label Extensions

The extensions described in this section are inspired by the OGC Engineering Report OGC 09-016 [http://portal.opengeospatial.org/files/?artifact_id=33515] [6] which defines labeling specializations on the base of a shared Label abstract class.

PointLabel

The *PointLabel* class is a concrete implementation of the Label class. This class defines specific visual properties to draw a text label relative to a point placement:

- for 0-dimensional geometry, it is the geometry itself;
- for 1 or 2-dimensional geometry the semantic is to compute and use a representative interior central point as the point placement (*i.e.*, guaranteed to lie on the line or the surface interior).

Notice that for a collection of geometries the label is just repeated at each part.

Table 19. Properties of the PointLabel class

Name	Definition	Data type and value	Multiplicity
transform	Defines a list of transform definitions that are applied (among Translate / Scale / Rotate and according to SVG transform definitions)	Transform	0/*
minDistance	The minimum distance between the next text label (defines a kind of exclusion zone)	Decimal	0..1

LineLabel

The *LineLabel* class is a concrete implementation of the Label class. It defines specific visual properties to draw a text label relative to a line. In particular, it draws an aligned text label relative to a placement line:

- for a 1-dimensional geometry, it is the line itself
- for a 2-dimensional geometry, it is a line following the shape of the geometry

Table 20. Properties of the LineLabel class

Name	Definition	Data type and value	Multiplicity
offset	Makes the label be parallel to the line at the given distance (by default the label is displayed so as the line crosses the center point of the label)	Decimal	0..1
repeat	If true then repeat labels by inserting the given gap values between each label along a line	Boolean	0..1
gap	Defines the gap value between each label repeated along a line (only relevant when the repeat property is set true)	Decimal	0..1
initialGap	How far away the first label will be drawn relative to the start of the rendering line (only relevant when the repeat property is set true)	Decimal	0..1

5.7.12. Default Values and Color Definition

For the conceptual model presented in this document, default values still need to be considered, also in relation to the specification of different encodings. In particular, all visual properties with

0..1 multiplicity may have a default value (*e.g.*, what is the default color for a PenStroke?). Pushing the "one conceptual model, many encodings" further, it would be interesting to allow each encoding specification to define its default values.

Please note that CartoCSS needs at least one visual property to be set to activate the symbolizer (*e.g.*, setting only *line-width = 2* implies *line-color = black*, whereas setting only *line-color = red* implies *line-width = 1*).

Moreover, concerning the definition of colors, Testbed-13 suggested using a *CSSColorLiteral* datatype that uses the same syntax as CSS and SVG. This may be questioned given a more conceptual definition (*e.g.*, tuple of numbers according to a color model).

Also, we should consider the possibility that an encoding can define its own named colors. This may be acceptable as soon as the encoding specification takes care to document the converting into the root definition of a color, that is a tuple of numbers according to a color model.

5.8. Towards a Basic Profile

The above conceptual model has the objective of defining the core structure and organization of a styling model with explicit extension points. In contrast, although the intent for Testbed-14 was not to define an exhaustive set of symbolizer abilities, those described above might be a good set to become a basic profile, including especially composition abilities.

We may wonder if a "super-basic" profile would be interesting for the elaboration of really simple cases. That profile would be set only with one specialization of Fill (SolidFill), one of Stroke (PenStroke), one of Mark (Image) and one of Label (PointLabel) and no composition.

Also, we may think to other profiles, for instance in relation to thematic mapping. Indeed, while choropleth and proportional symbols are made possible by the basic profile defined above (and the availability of transformation functions like categorize and interpolate defined by SE), some advanced abilities are useful.

We may think to specific Fill sub-types:

- HatchedFill: https://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/Core_xsd.html#HatchedFill
- DensityFill: https://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/Thematic_xsd.html#DensityFill
- DotMapFill: https://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/Thematic_xsd.html#DotMapFill

Or even specific Mark sub-types:

- PieChart: https://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/Thematic_xsd.html#PieChart
- AxisChart: https://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/Thematic_xsd.html#AxisChart

Finally, we may wonder if the ability of sharing identifiers introduced by Testbed-13 has to considered part of the basic profile.

5.9. Towards Encoding Specifications

The idea for Testbed-14 was not to define a complete default encoding, but rather to start the definition of good practices to elaborate an encoding specification document.

The main point to underline here is related to the organizing of such a document. Indeed, given a conceptual model which describes styling abilities with the definition of explicit requirements for each table and rendering characteristics, an encoding shall describe a syntax mapping for all the related classes and properties in the conceptual model.

For instance, considering the below Requirement related to composition abilities:

- Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/Compose
- Requirement Txt: Implementations shall support the encoding of the table properties and rendering characteristics of the Compose class and meet all of the tabulated constraints and notes.

The **GeoCSS syntax mapping** would explain how all properties may have multiple values separated by commas. The specification document would import content explained in the documentation with some relevant examples (https://docs.geoserver.org/stable/en/user/styling/css/multivalueprops.html).

The **CartoCSS syntax mapping** would explain how it is possible to create multiple symbols of the same type using named instances. The specification document would import content explained in the documentation with some relevant examples (https://tilemill-project.github.io/tilemill/docs/guides/symbol-drawing-order/).

Chapter 6. Portrayal Demonstration

This section summarizes the workflows used for Portrayal Thread demonstration and Technology Integration Experiments (TIEs).

NOTE

The Goal of the Portrayal Challenge

Given the same data and symbols, provide the same (or as similar as possible) portrayal in each user's computing environment.

The Portrayal demo tries to link symbols and features, and help different users quickly understand the Daraa region in their computing environment (Figure 3). Interoperable geospatial portrayal still is challenging across multiple environments such as:

- Web/mobile/desktop (computing)
- Server-side, client-side (rendering)
- Connected/disconnected/intermittent/limited (communications)
- Aerial/imagery/hillshade/gray/night backgrounds (display)

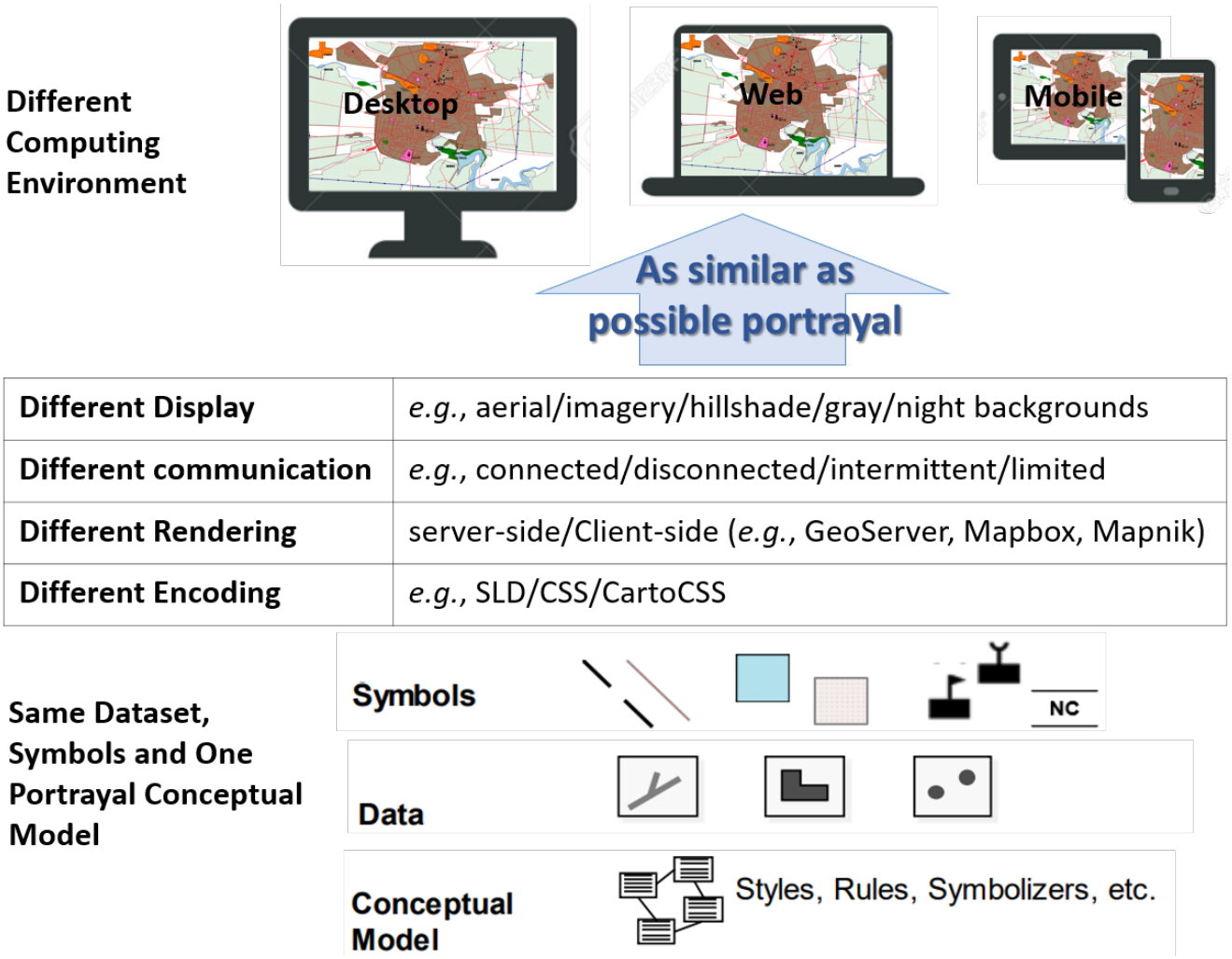


Figure 3. Portrayal Challenge: Given the same data and symbols, provide the same (or as similar as possible) portrayal in each user's computing environment

6.1. Daraa Test Dataset

To demonstrate the robustness and flexibility of the portrayal model to accommodate different data models and formats, we have implemented and tested the following formats and protocols to build map layers.

6.1.1. Shapefile

The Shapefile format is a legacy data format for sharing 2D vector data between geospatial software systems. Esri developed the Shapefile format as an open specification for data interoperability among Esri and other geographic information system (GIS) software products. The Shapefile format can spatially describe vector features: points, lines, and polygons, representing simple feature data. A geospatial dataset encoded in Shapefile format is composed of multiple files. However, many services distribute Shapefiles in a ZIP compressed file form so that they can be bound together as a coherent set of files. For this reason, the implementation of the portrayal service supports creation of a Shapefile that is wrapped in a ZIP file that can be accessed remotely via a simple uniform resource locator (URL) for the demonstration (Figure 4). These data files are available in the OGC Portal.

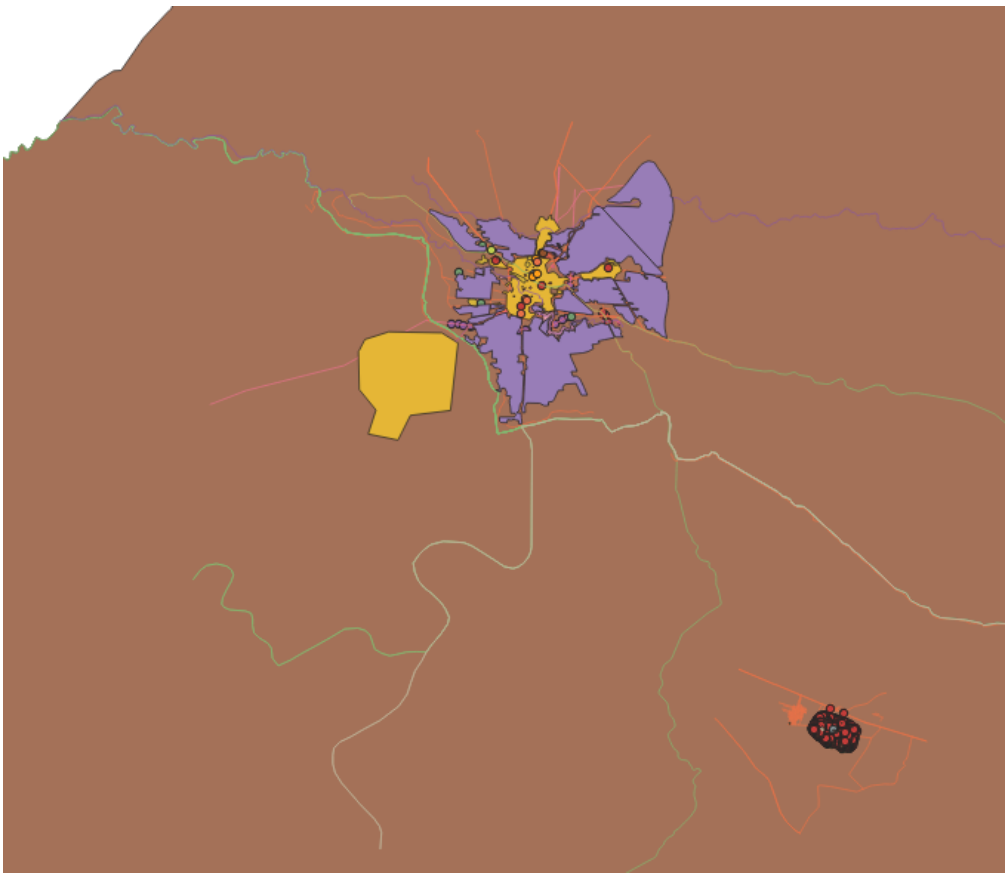


Figure 4. Shapefiles imported into QGIS (Quantum Geographic Information System) with default random styles applied

6.2. Sample Symbol Sets

The styles in this document were provided to the OGC Testbed-14 effort for informational purposes only. They are a reflection of the geospatial data and portrayal work performed by National Geospatial-Intelligence Agency (NGA), Army Geospatial Center (AGC), Institut National de

l'information geographique et forestiere (IGN), Defence Information Systems Agency (DISA), and Federal Geographic Data Committee (FGDC). The data is OpenStreetMap (OSM) over Daraa, Syria converted into the NGA Topographic Data Store (TDS) data model, and provided to OGC in support of Testbed13 and 14. The styles and portrayal description for the feature types were derived from NAPSG (National Alliance for Public Safety GIS), DISA, IGN, and AGC. The expressions were prepared by AGC by cross walking symbols with NGA TDS OpenStreetMap (OSM) feature types over Daraa, Syria. AGC assessed the NGA data to ensure the majority of symbols presented have feature geometries in the Daraa, Syria NGA TDS OSM data set. In some cases, Topographic Points or Points of Interest (POI) geometries may need to be generated if the symbol (such as a Dam, MIL-STD military standard 2525 or Emergency Management POI) is desired to be used during TIEs.

The styles also included symbols for Points of Interests. The POIs section recognizes that custom portrayals of vector data are a common use case in many communities, and emerging OGC Portrayal Conceptual Models and Encodings need to be able to support renderings of POI symbols (such as NAPSG). This section also provides a placeholder for military standard 2525 (MIL-STD-2525) symbols for testing. Simple SVG examples are provided, and Testbed-14 should acquire at least one complex symbol from the MIL-STD-2525D for TIE testing.

The following items pertain to applying portrayal descriptions for feature data:

- Line Weights and Point/Graphic sizes: Unless otherwise noted, all descriptions containing stroke-width, size, *etc.* are in meters.
- Expressions are as follows:
  - FCODE - Feature Code
  - Feature Class - Profile of the NAS (NSG (National System for Geospatial Intelligence) Application Schema) that combines features into a collective feature class
  - Expression - The attribute expression that maps directly to the symbolized element
- Drawing Order of symbols with multiple components, is depicted by the order presented in each of the symbol sections.
- SVGs for Topographic Points/Graphic, Graphic Fill, and Graphic Stroke are in an associated Zip folder.
- SVGs for Points of Interest (MIL-STD-2525D) are available at https://spatialillusions.com/https://github.com/missioncommand/mil-sym-js/issues
- Symbols for Emergency Management are available for download at https://napsg-web.s3.amazonaws.com/symbology/index.html#/subcat?Incident

6.2.1. Composite Symbols and MIL-STD-2525

MIL-STD-2525 symbology is often used as overlays for situational awareness and has been implemented in a variety of systems in different ways. The current version is MIL-STD-2525D, however most systems are still using 2525C. Symbol Identification Codes (SIDC) are numeric codes that uniquely identify the elements needed to build a MIL-STD-2525D compliant symbol.

MIL-STD-2525D uses a component-based approach to symbology to provide thousands of symbol combinations from a limited set of symbol components made up of frames, icons, and modifiers. In most military systems, the SIDC codes are passed to the system with location information and the

system is designed to interpret the code and render the appropriate symbols as an overlay.

The goal of using MIL-STD-2525 symbology is to portray the provided Scaled Vector Graphics (SVG) and document how the component-based approach can be incorporated into the models/methods under development within Testbed-14. For this purpose, the following steps need to be considered:

- Use the individual components of the symbol to create a new composite symbol based on attributes within the data.
- Use the Angle of Orientation Attribute to create an appropriate rotation of the Antipersonnel Mine with Directional Effects, the angle of orientation should maintain its orientation with the north when the map rotates.
- Symbols such as Emergency Operation and Organization or Group should rotate with the map so the people are upright and the triangle points up.

To simplify the task of rendering these symbols, a shapefile is provided using portions of the SIDC code to provide information to use as expressions for composite symbols made up of frames and icons or modifiers and to rotate symbols based on an angle of orientation.

6.3. Sample Encodings

There will be multiple encodings, (*e.g.*, Cascading Style Sheets (CSS), SLD, and CartoCSS) to identify and assess alternatives to the OGC Style Layered Descriptions/Symbol Encoding (SLD/SE) standard as implemented in OGC GeoPackage and OGC WMS/WMTS.

6.4. Demonstration Workflow

This section discusses the portrayal thread Technology Integrated Experiments (TIEs) conducted in Testbed-14.

Web Map Service

For the demonstration, we performed a TIE with the Web Map Service 1.1.0 from the University of Calgary. For this deliverable, a straightforward WMS implementation was provided that supports rendering the conceptual model from D160. A public repository was provided on GitHub with a Node.js implementation of a WMS version 1.1.1 (implementing GetCapabilities and GetMap). It uses Mapnik to render images, and the stylesheet is defined in CartoCSS and compiled to Mapnik XML.

Web Map Tile Service

This work item contributed to:

- Experience from implementing the rendering engine of GeoServer (both raster and vector)
- Experience in extending SLD in GeoServer to account for more sophisticated styling
- Experience in developing and maintaining the GeoCSS styling encoding in GeoServer

The objectives:

- Help with setting the right path to go beyond SLD -→ contribute to the conceptual model

- Promoting GeoCSS and our extension to SLD to be taken into account for the point above
- Push farther limits of our rendering engine by looking at other exciting styling approaches such as CartoCSS and Mapbox Styles

GeoPackage

This last experiment is to implement GeoPackage with Portrayal Support. This effort was conducted by Image Matters LLC to support GeoPackage feature portrayal on mobile devices.

6.4.1. TIE Components and Result

The following table provides an overview of the components involved in the portrayal TIEs. All of the TIEs were successful.

Table 21. Involved TIE Components

Deliverable	Component Name and Description	Usecase	Success	More Information
D160	Portrayal Ontology to specify a conceptual framework for portrayal data (e.g. symbols for vectors and portrayal rules)	Extend the portrayal ontology to represent composite symbols	Yes	Section 5
D161	GeoPackage with Portrayal Support	Visualizing Daraa GeoPackage feature portrayal on mobile devices	Yes	Section 10
D162	WMS with Portrayal Support	Visualizing and rendering Daraa features based on the conceptual model using a simple WMS	Yes	Section 9
D163	WMTS with Portrayal Support	Visualizing and rendering Daraa features based on the conceptual model using a WMTS	Yes	Section 7

Chapter 7. WMTS with Portrayal Support

The purpose of this chapter is to capture the work performed by GeoSolutions on styling the NSG data for Daraa and Zaatari using SLD and CSS for GeoServer.

This section covers information on how the data was ingested into PostgreSQL and GeoServer as well as on the styles' creation process itself. Both SLD and CSS styles are created as close as possible to the original guidelines and then a minimal revision is made to reuse some of the OSM styling guidelines; both styles are presented in the document.

7.1. Styling Daraa Topographic Data Store

7.1.1. Import Data Process

Some of the necessary information to style the NGA data are localized in column `ZI006_MEM` that contains OSM tags in a JavaScript object notation (JSON) snippet. The process below has been used to import such data into PostGIS and to obtain a new data structure with additional columns with the needed OSM tags information to simplify the styling work in GeoServer.

Parts of the scripts for conversion are written in JavaScript and available here [https://github.com/geosolutions-it/ogc-testbed14-styles/tree/master/scriptstestbed].

Conversion Steps

Using the extend.js [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/scriptstestbed/extend.js] script:

- Converted geodatabase file of Daraa to Geospatial JavaScript Object Notation (GeoJSON) with European Petroleum Survey Group (EPSG): `3857` projection, source projection of geodatabase file is EPSG: `4326` (saved in `./out` folder).
- Extracted OSM tag [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/scriptstestbed/extend.js#L42-L53] values from `ZI006_MEM` column and mapped to new columns, mapped and created only declared columns in configuration [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/scriptstestbed/config.json] (saved in `./extended` folder).
- Converted GeoJSON to Shapefile.

Import Steps

- Created a new database in Postgres named *daraa* with PostGIS extension.
- Imported Shapefile to Postgres and used "shp2pgsql" library to preserve the case of columns title (saved in `./shapefiles` folder).
- Added new PostGIS database store to GeoServer.
- Connected it to database *daraa* with schema public.
- Publish layer needed for styling to layer groups.
- Created and applied styles.

NOTE	import.js [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/scriptstestbed/import.js] in `./scriptows` folder does not import cases of columns title correctly, so the *shp2pgsql* library used instead with `Preserve case of column names` option enabled.

The GeoServer data directory which we provide in this GitHub repository [https://github.com/geosolutions-it/ogc-testbed14-styles] contains styles, layers, and *layergroups* to create the final map but it also needs the above data to be imported to a PostgreSQL database. Layer groups are *Daraa* for guideline style and *Daraa_review* for the reviewed one. Styles follow the same rules of layer groups, "_review" suffix indicates reviewed ones.

7.1.2. Styling Feature Classes

This section provides the detailed information regarding styles created for each feature class converted from the original File Geodatabase.

NOTE	All GeoCSS styles can be found in Appendix C document.

AgricultureSrf **Feature Class**

The style for this layer includes an applied pattern (*Green_AP77_Fill.svg*) combined with a fill solid background `#F7EDED`. Guidelines seem to require Graphic Fill and Graphic at the same time, and the current style uses only graphic fill.

In this case, it could be enough to have a single pattern and avoid multiple combinations to decrease the complexity of agriculture pattern.

Figure 5 shows a preview and both style files are available here: *AgricultureSrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/agriculturesrf.css] and *AgricultureSrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/agriculturesrf.sld].

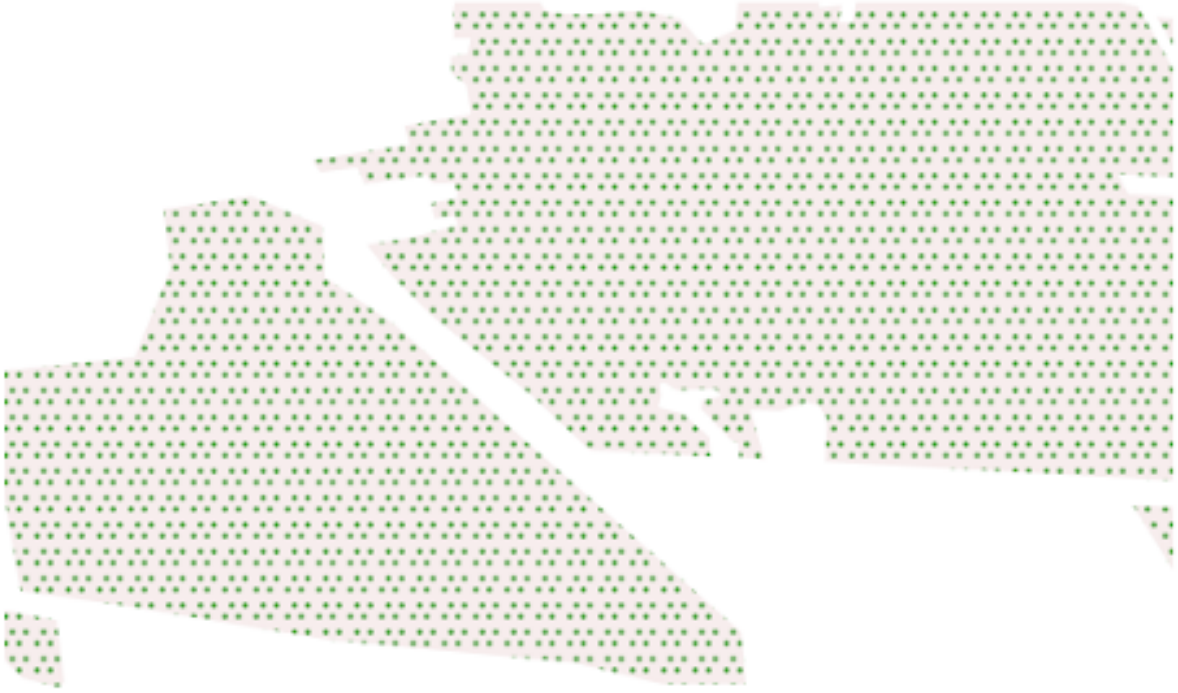


Figure 5. Style preview for guideline AgricultureSrf feature class styling

CulltureSrf Feature Class

The style for this layer includes stroke with dasharray as described in guidelines.

Figure 6 shows a preview and both style files are available here: *CulltureSrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/culturesrf.css] and *CulltureSrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/culturesrf.sld].

Figure 6. Style preview for guideline CulltureSrf feature class styling

HydrographyCrv **Feature Class**

The style for this layer includes the OSM tag waterway instead of `WCC` or `FCSUBTYPE` because they do not contain readable values to assign styles. All `FCSUBTYPE` attributes are `100314` and `WCC` values are `1`, `7`, and `999`. It seems that `1 = stream`, `7 = river`, and `999 = drain`.

Figure 7 shows a preview and both style files are available here: *HydrographyCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographycrv.css] and *HydrographyCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographycrv.sld].

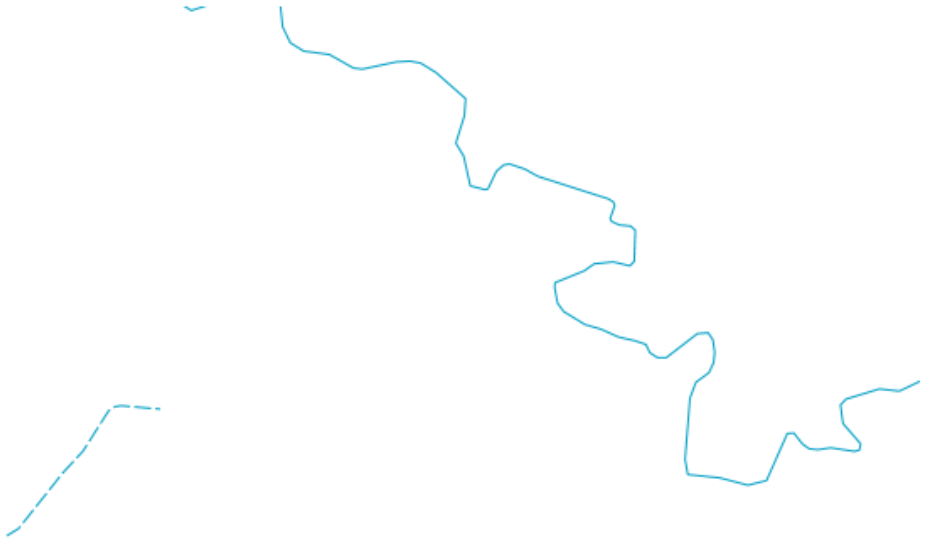


Figure 7. Style preview for guideline HydrographyCrv feature class styling

NOTE

Review

Applied on stroke, with a similar color to a lake fill surface. This change ensures visual continuity between streams lines and water surfaces at higher denominator scales.

Figure 8 shows a preview and both style files are available here: Reviewed *HydrographyCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographycrv_review.css] and Reviewed *HydrographyCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographycrv_review.sld].

Figure 8. Style preview for reviewed HydrographyCrv feature class styling

HydrographySrf Feature Class

Figure 9 shows a preview and both style files are available here: *HydrographySrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographysrf.css] and *HydrographySrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographysrf.sld].

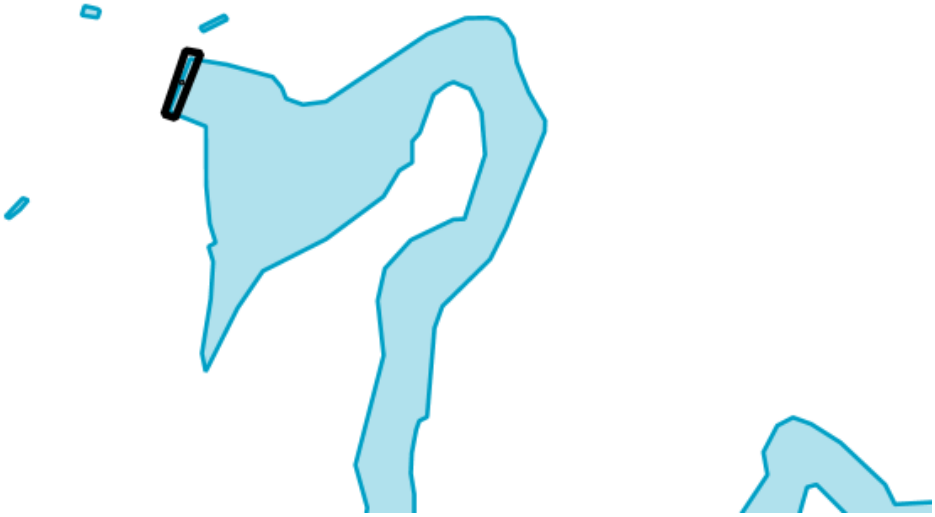


Figure 9. Style preview for guideline HydrographySrf feature class styling

NOTE

Review

Removed Dam icon because it was too small and hidden by the Dam line; instead, we could apply a fill to highlight the Dam surface. Removed stroke from the lake to blend it with the background.

Figure 10 shows a preview and both style files are available here: Reviewed *HydrographySrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographysrf_review.css] and Reviewed *HydrographySrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/hydrographysrf_review.sld].

Figure 10. Style preview for reviewed HydrographyCrv feature class styling

InformationPnt **Feature Class**

The current style uses Open Sans font family and the name is in English, taken from OSM tags as labels.

Figure 11 shows a preview and both style files are available here: *InformationPnt* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/informationpnt.css] and *InformationPnt* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/informationpnt.sld].

Figure 11. Style preview for guideline InformationPnt feature class styling

NOTE	*Review* Guidelines indicate *universal* as font family, so sans-serif font was selected as *Open Sans* to ensure readability.

RecreationSrf Feature Class

The current style simulates a polygon with a stroke that could apply an offset. It has been applied a pattern (sports_ground_wilson) between `1 : 80000` to `1 : 200000` scales.

Figure 12 shows a preview and both style files are available here: *RecreationSrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/recreationsrf.css] and *RecreationSrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/recreationsrf.sld].

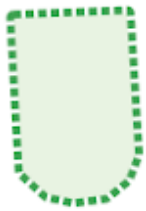


Figure 12. Style preview for guideline RecreationSrf feature class styling

SettlementSrf Feature Class

Figure 13 shows a preview and both style files are available here: *SettlementSrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/settlementsrf.css] and *SettlementSrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/settlementsrf.sld].

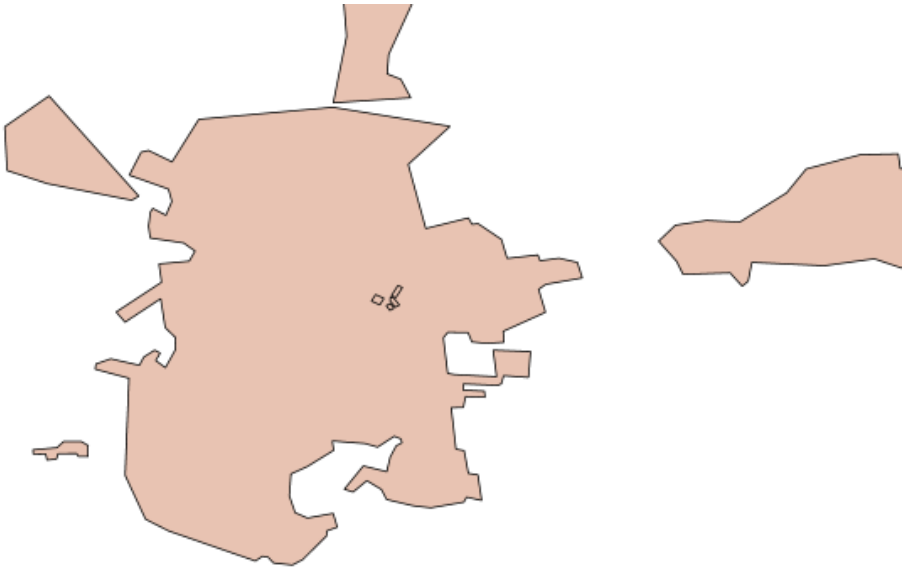


Figure 13. Style preview for guideline SettlementSrf feature class styling

NOTE	*Review* Removed stroke to blend surface with the background.

Figure 14 shows a preview and both style files are available here: Reviewed *SettlementSrf* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/settlementsrf_review.css] and Reviewed *SettlementSrf* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/settlementsrf_review.sld].

Figure 14. Style preview for reviewed SettlementSrf feature class styling

StructurePnt **Feature Class**

Used amenity tag to assign styles also for public buildings because they contain the same values of `OTH` column. Size and max/min scales denominators are not defined for a public building, so we applied a scale denominator with visibility less than `1 : 200000` and size of `70m` (greater than general buildings).

Figure 15 shows a preview and both style files are available here: *StructurePnt* CSS [https://github.com/

geosolutions-it/ogc-testbed14-styles/blob/master/styles/structurepnt.css] and *StructurePnt* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/structurepnt.sld].

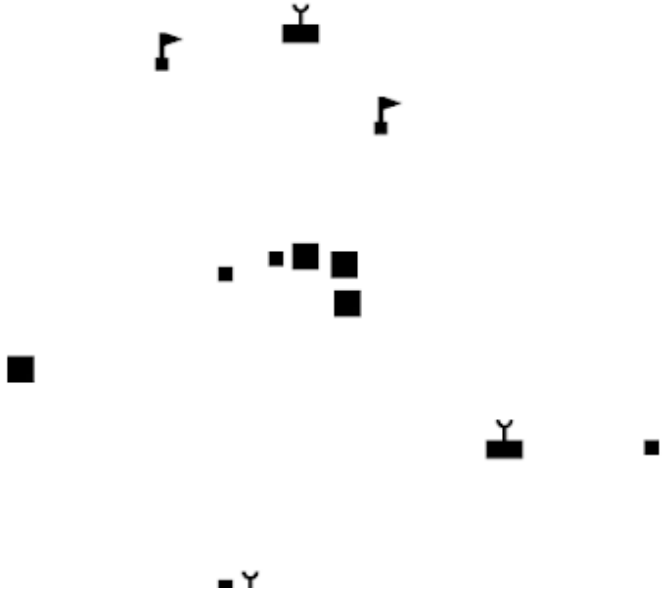


Figure 15. Style preview for guideline StructurePnt feature class styling

NOTE

- The general building doesn't seem correct with scales denominators, so we applied two ranges, scales `0 - 100000`, and scales `100000 - 200000`.
- Some icons are not visible to high denominator scales, *e.g.*, *PT_Black_Mosque.svg*.

TransportationGroundCrv **Feature Class**

Modified stroke-width of a street from `500000` to `5000` scale denominator to display line also in crowded places.

Figure 16 shows a preview and both style files are available here: *TransportationGroundCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv.css] and *TransportationGroundCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv.sld].

Figure 16. Style preview for guideline TransportationGroundCrv feature class styling

NOTE

Review

Applied a style with lighter colors to blend roads with the background and have better contrast with elements on top (*e.g.*, marker). Non-primary roads now have a stroke color brighter than settlement surface but with the same gradient of color instead of white. This changes helps to highlight not primary roads in agriculture surfaces but blend them in settlement surfaces. Added labels in English.

Figure 17 shows a preview and both style files are available here: Reviewed *TransportationGroundCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_review.css] and Reviewed *TransportationGroundCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_review.sld].

Figure 17. Style preview for reviewed TransportationGroundCrv feature class styling

NOTE

- It could be useful to have *Z-order* parameter provided from OSM, and it could help to render roads order correctly.

TransportationGroundCrv_highlight **Feature Class**

Used a double stroke and applied geometry transformations to get tickmarks at corners for bridges.

Figure 18 shows a preview and both style files are available here: *TransportationGroundCrv_highlight* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_highlight.css] and *TransportationGroundCrv_highlight* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_highlight.sld].

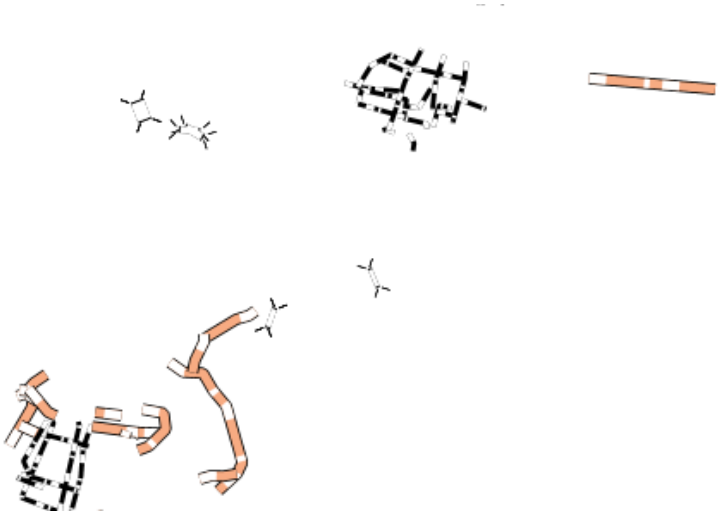


Figure 18. Style preview for guideline TransportationGroundCrv_highlight feature class styling

NOTE	*Review* Used lighter colors to highlight other element on the map, used different scales denominator. Displayed tick marks only on lower scale denominator. Here below the created styles and a preview snippet.

Figure 19 shows a preview and both style files are available here: Reviewed *TransportationGroundCrv_highlight* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_highlight_review_review.css] and Reviewed *TransportationGroundCrv_highlight* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/transportationgroundcrv_highlight_review.sld].

Figure 19. Style preview for reviewed TransportationGroundCrv_highlight feature class styling

UtilityInfrastructureCrv Feature Class

Figure 20 shows a preview and both style files are available here: *UtilityInfrastructureCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurecrv.css] and *UtilityInfrastructureCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurecrv.sld].

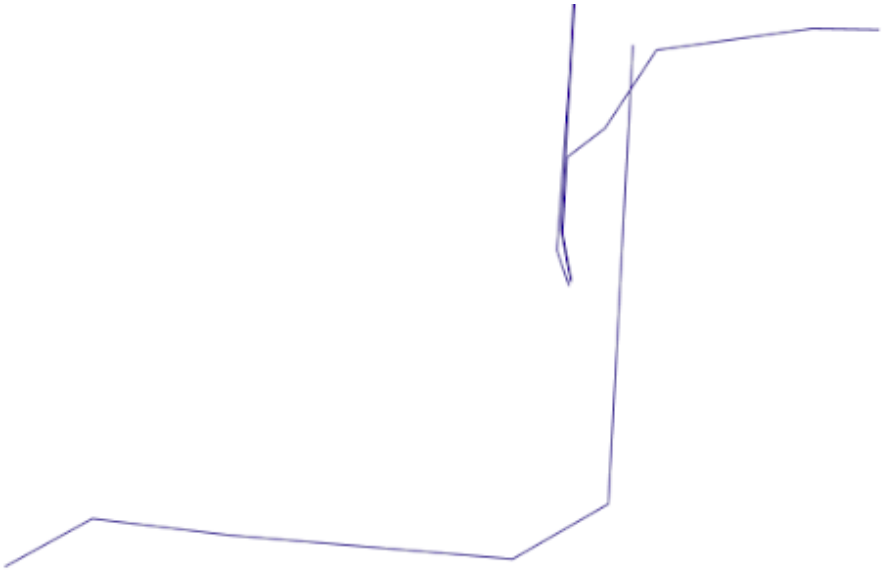


Figure 20. Style preview for Guidelines UtilityInfrastructureCrv feature class styling

NOTE	*Review* Used #333333 color for stroke to obtain a high contrast. Here below the created styles and a preview snippet.

Figure 21 shows a preview and both style files are available here: Reviewed *UtilityInfrastructureCrv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurecrv_review_review.css] and Reviewed *UtilityInfrastructureCrv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurecrv_review.sld].

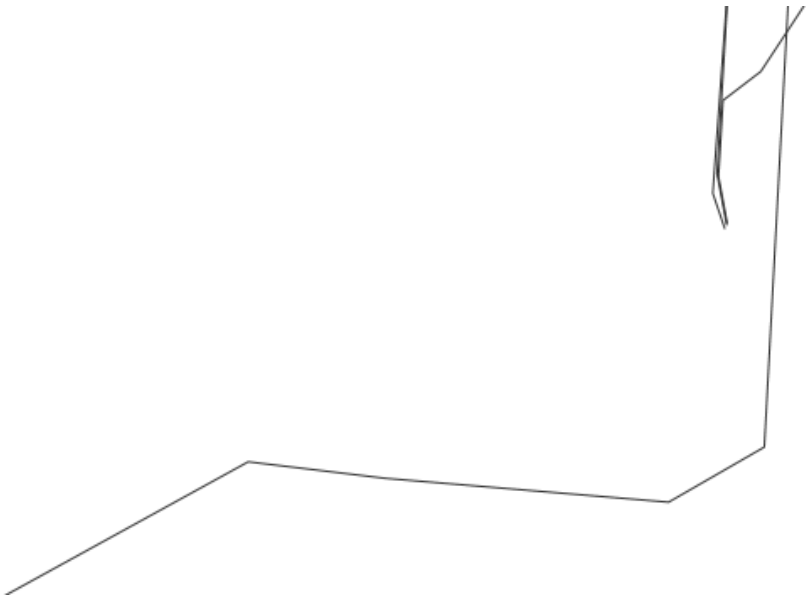


Figure 21. Style preview for reviewed UtilityInfrastructureCrv feature class styling

UtilityInfrastructurePnt **Feature Class**

We had to fix *pylon_wilson.svg*, because its position in the SVG layout is not placed correctly. This problem could happen with others markers; so, it is important to check the structure of the SVG element before using it in styles.

Figure 22 shows a preview of SVG files.

Figure 22. left original, right adjusted SVG files

We applied a different scale denominator to labels, less than `1 : 35000`.

Figure 23 shows a preview and both style files are available here: *UtilityInfrastructurePnt* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurepnt.css] and *UtilityInfrastructurePnt* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/utilityinfrastructurepnt.sld].

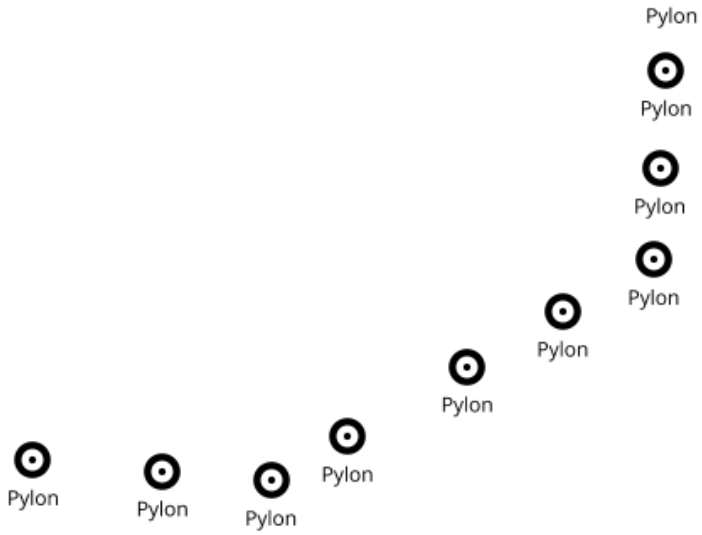


Figure 23. Style preview for guideline UtilityInfrastructurePnt feature class styling

VegetationSrfv Feature Class

Here below the created styles and a preview snippet.

Figure 24 shows a preview and both style files are available here: *VegetationSrfv* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/vegetationsrfv.css] and *VegetationSrfv* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/vegetationsrfv.sld].

Figure 24. Guidelines style preview for VegetationSrfv feature class styling

Emergency Management Symbology

Used a temporary Shapefile layer to simulate those markers. If original SVG contains an image instead of a path, it needs to be converted.

Figure 25 shows a preview of Emergency Management Symbology.

Figure 25. Guideline style preview for Emergency Management Symbology

7.1.3. Raster Symbolizer

Figure 26 shows a preview and both style files are available here: *Raster* CSS [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/landsat8.css] and *Raster* SLD [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/landsat8.sld].

Figure 26. Guideline style preview for Landsat 8 raster image

7.1.4. MIL-STD-2525 Symbolizer

Added symbol visibility only under `1 : 35000` scale.

Figure 27 shows a preview and both style files are available here: *MIL-STD-2525 CSS* [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/MIL-STD2525D.css] and *MIL-STD-2525 SLD* [https://github.com/geosolutions-it/ogc-testbed14-styles/blob/master/styles/MIL-STD2525D.sld].

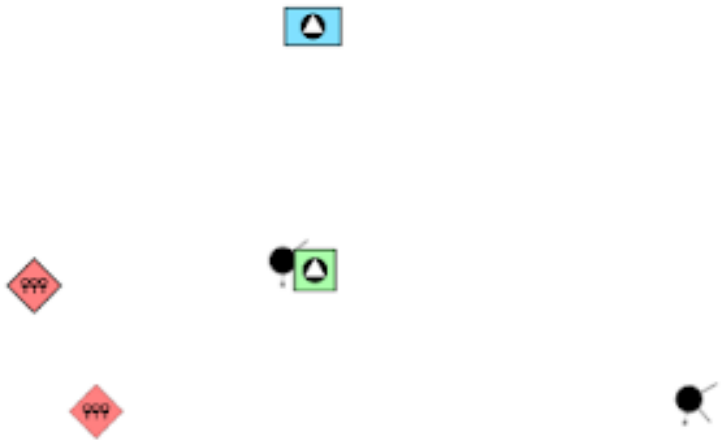


Figure 27. Guideline style preview for MIL-STD-2525 Symbology

7.1.5. All Feature Class Layers

Here below you can find the final results for both the styles closer to the original guidelines as well as the style we have revised a little to reuse some of the OSM styling guidelinees. The demo is available at this link [https://dev.mapstore.geo-solutions.it/mapstore/#/viewer/openlayers/7234].

Figure 28. A preview of the guideline style for all the layers

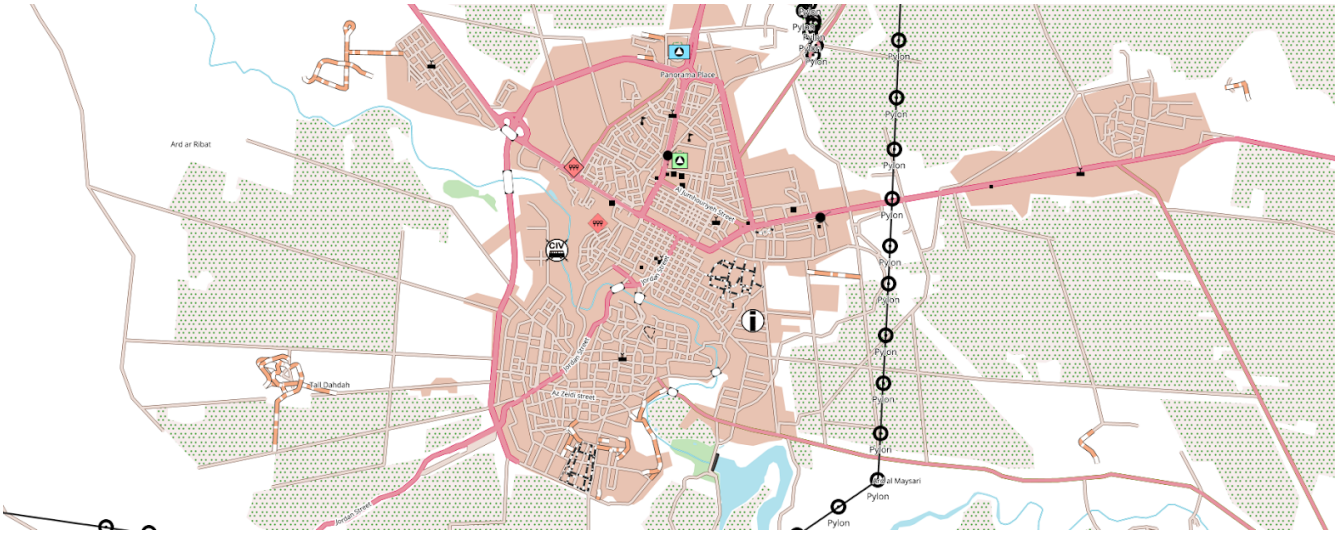


Figure 29. A preview of the reviewed version of the style

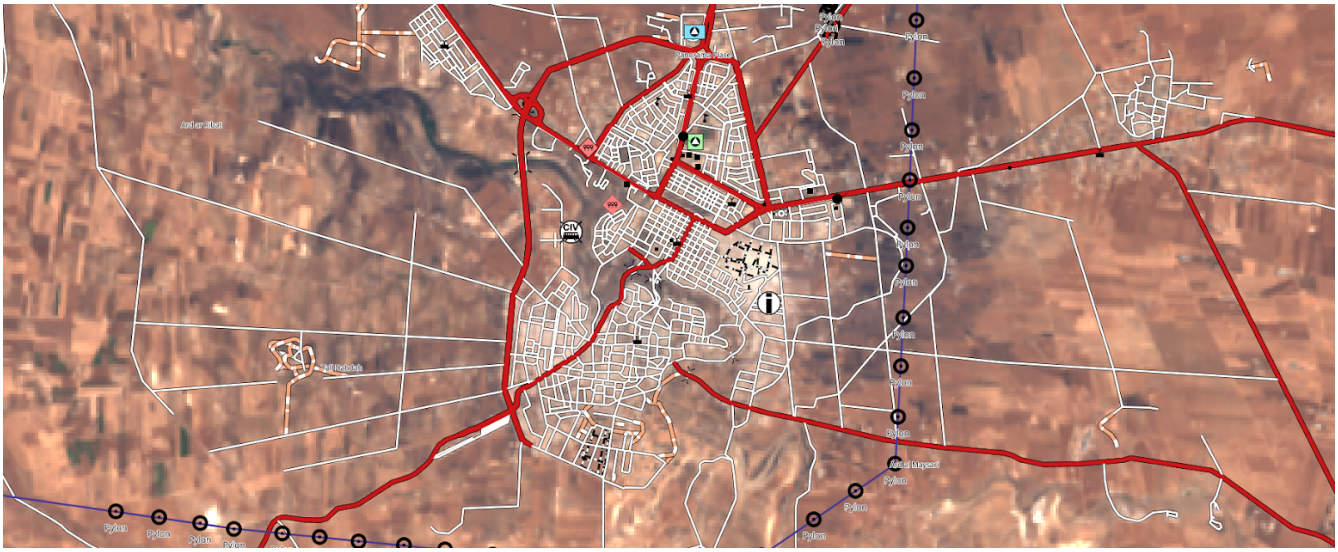


Figure 30. Overlay of roads and POIs on Landsat8 for guideline style

Figure 31. Overlay of roads and POIs on Landsat8 for guideline style

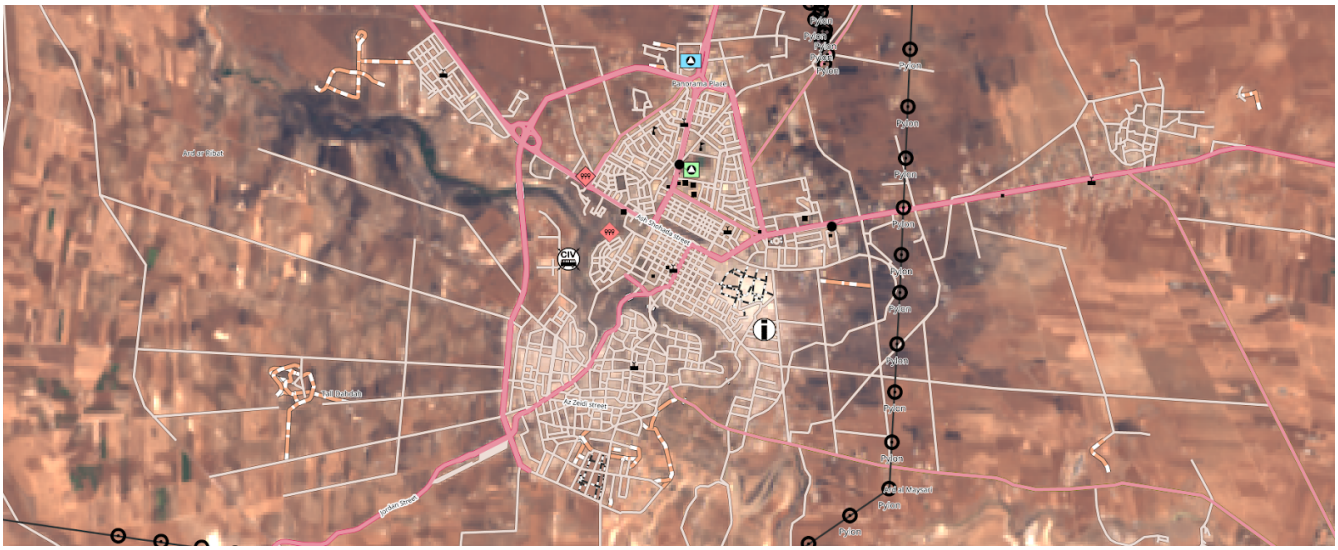


Figure 32. Overlay of roads and POIs on Landsat8 for reviewed style

Figure 33. Overlay of roads and POIs on Landsat8 for reviewed style

7.2. Bridge Style Tickmarks Challenge

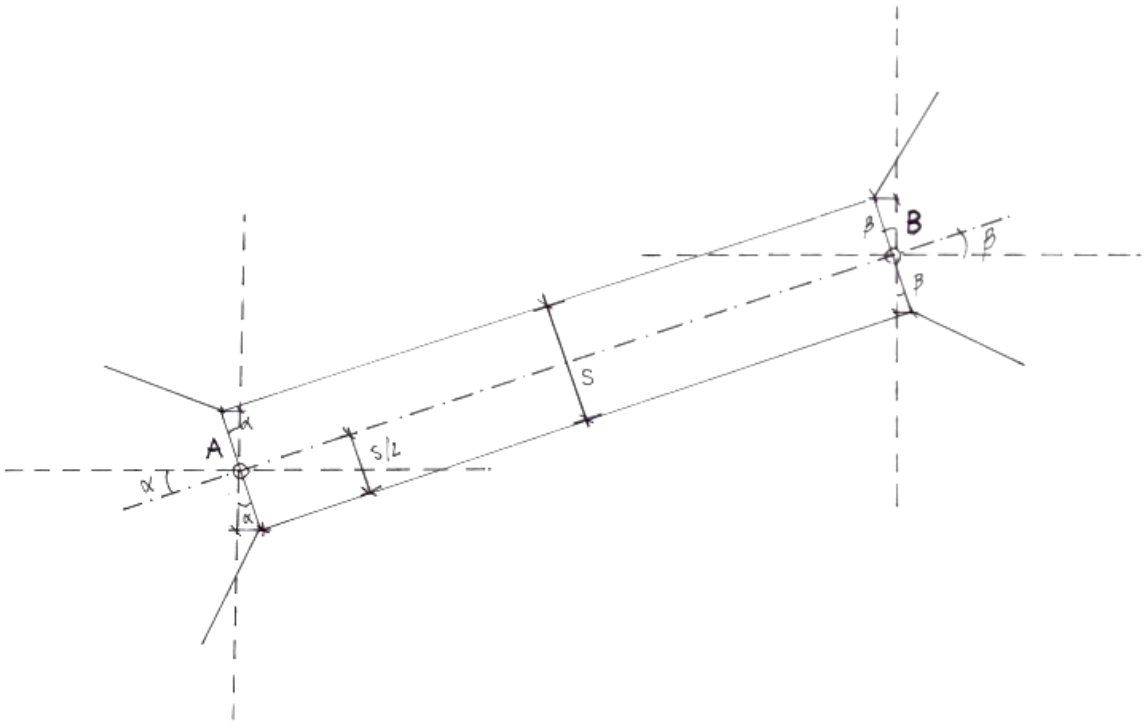


Figure 34. Bridge style tickmarks

In this figure, "**A**" is the startPoint(the_geom), "**B**" is the endPoint(the_geom), "**alpha**" is the startAngle(the_geom), "**Beta**" is the endAngle(the_geom), and "**s**" is stroke-width.

Extract the start and end points of the original line, place lines marks there:

```
mark-geometry: [startPoint(the_geom)], [endPoint(the_geom)], [startPoint(the_geom)],
[endPoint(the_geom)];
mark: symbol('shape://vertline');
```

Figure 35. Preview of extracting the start and end points of the original line and placing lines marks there

Rotate the form marks to obtain an angle of 135° using the start and end angle of the line geometry (we need to add/remove 45°).

```
mark-geometry: [startPoint(the_geom)], [endPoint(the_geom)], [startPoint(the_geom)],
[endPoint(the_geom)];
mark: symbol('shape://vertline');
mark-rotation: [startAngle(the_geom) - 45], [endAngle(the_geom) + 45], [startAngle
(the_geom) + 45], [endAngle(the_geom) - 45];
```

Figure 36. Preview of rotating the form marks

Set the anchor of the mark to top-left `(0 0)` and top-right `(0 1)`. Marks with top-left anchor will be apply on the bottom side of the line, top-right on the top side.

```
mark-geometry: [startPoint(the_geom)], [endPoint(the_geom)], [startPoint(the_geom)],
[endPoint(the_geom)];
mark: symbol('shape://vertline');
mark-rotation: [startAngle(the_geom) - 45], [endAngle(the_geom) + 45], [startAngle
(the_geom) + 45], [endAngle(the_geom) - 45];
mark-anchor: 0 0, 0 0, 0 1, 0 1;
```

Figure 37. Preview of setting the anchor of the mark to top-left and top-right

Calculate the offsets of marks with trigonometry formulas, knowing start and end angle of the line and the offset perpendicular to the line that represent the hypotenuse of the right triangle (`stroke-width / 2`)

```
mark-geometry: [startPoint(the_geom)], [endPoint(the_geom)], [startPoint(the_geom)],
[endPoint(the_geom)];
mark: symbol('shape://vertline');
mark-rotation: [startAngle(the_geom) - 45], [endAngle(the_geom) + 45], [startAngle
(the_geom) + 45], [endAngle(the_geom) - 45];
mark-anchor: 0 0, 0 0, 0 1, 0 1;
mark-offset: [10 * sin(toRadians(startAngle(the_geom)))] [-10 * cos(toRadians
(startAngle(the_geom)))], [10 * sin(toRadians(endAngle(the_geom)))] [-10 * cos
(toRadians(endAngle(the_geom)))], [-10 * sin(toRadians(startAngle(the_geom)))] [10 *
cos(toRadians(startAngle(the_geom)))], [-10 * sin(toRadians(endAngle(the_geom)))] [10
* cos(toRadians(endAngle(the_geom)))];
```

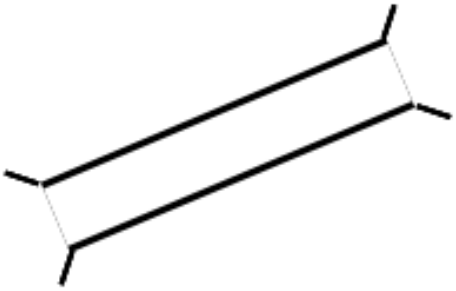


Figure 38. Preview of moved marks based on the calculated offset

Chapter 8. Lessons from CARTO and Mapbox Styles

This section includes rendering the Daraa dataset using two approaches based on free and open source solutions: CARTO online builder (CartoCSS code generator) and Mapbox studio (data oriented with JSON filtering). The final goal of this design was to compare those solutions with GeoServer to observe their common elements and their differences. The idea was not to stick as close as possible to the Daraa map guidelines given during the Testbed-14, but more about developing a meta-understanding of the solutions. A reverse-engineering of CartoCSS is done to notice how close the rendering engine is to the agreed conceptual model during the Testbed-14. This text is divided as follows: First, the challenges working with Mapbox and CartoCSS are described; then the rendering of Dara's refugee camp is described using CartoCSS and Mapbox. Then, the reconstruction of the rendering-engine of CartoCSS is described.

8.1. Challenges in Styling Feature Classes

The work done using GeoServer with GeoCSS has been reformatting to suit the requirements of the CartoCSS and Mapbox Style rendering engines. However, starting from the encoding of a solution, a few challenges have been addressed:

- The goal is to do the least modification possible on the Testbed-14 data set including the SVGs and the Shapefiles. To avoid pre-processing the data we had to find a way to use a regular expression to filter data the OSM tag of the column *ZI006_MEM*. In CartoCSS we have succeeded in applying the symbol "=~" as following:

```
[zi006_mem =~".*farmland.*"]
```

Unfortunately, this has not been solved for the Mapbox Style, using a data-driven solution.

- GeoServer uses a continuous scale to render the symbology. Unfortunately, both CartoCSS and Mapbox work on discontinuous zoom levels. To approximate the same visualization scale as GeoServer, we have used the OSM zoom levels [https://wiki.openstreetmap.org/wiki/Zoom_levels] scale which is shown in Figure 39.

Level	# Tiles	Tile width (° of longitudes)	m / pixel (on Equator)	~ Scale (on screen)	Examples of areas to represent
0	1	360	156 412	1:500 million	whole world
1	4	180	78 206	1:250 million	
2	16	90	39 103	1:150 million	subcontinental area
3	64	45	19 551	1:70 million	largest country
4	256	22.5	9 776	1:35 million	
5	1 024	11.25	4 888	1:15 million	large African country
6	4 096	5.625	2 444	1:10 million	large European country
7	16 384	2.813	1 222	1:4 million	small country, US state
8	65 536	1.406	610.984	1:2 million	
9	262 144	0.703	305.492	1:1 million	wide area, large metropolitan area
10	1 048 576	0.352	152.746	1:500 thousand	metropolitan area
11	4 194 304	0.176	76.373	1:250 thousand	city
12	16 777 216	0.088	38.187	1:150 thousand	town, or city district
13	67 108 864	0.044	19.093	1:70 thousand	village, or suburb
14	268 435 456	0.022	9.547	1:35 thousand	
15	1 073 741 824	0.011	4.773	1:15 thousand	small road
16	4 294 967 296	0.005	2.387	1:8 thousand	street
17	17 179 869 184	0.003	1.193	1:4 thousand	block, park, addresses
18	68 719 476 736	0.001	0.596	1:2 thousand	some buildings, trees
19	274 877 906 944	0.0005	0.298	1:1 thousand	local highway and crossing details

Figure 39. Correspondence table between zoom level and scale (from OSM wiki [https://wiki.openstreetmap.org/wiki/Zoom_levels])

The stroke size used in GeoServer is in ground units. Unfortunately, the Mapbox Style and CartoCSS use a pixels unit to describe their line size. To simplify the process, we manually divided the size implemented on GeoServer to be visually as close as possible. To face this problem, a formula can be used to convert ground space to pixels units as:

```
size = fct(zoom level).
```

Unfortunately, the value of the zoom cannot be picked dynamically from a parameter in CartoCSS. To change the size according to the zoom level, a "size" must be attached to each zoom level. However, it seems that TileMill and Mapnik have introduced a zoom variable with "@zoom". Using one of those design tools helps to face this issue dynamically.

8.2. Portrayal Support using CartoCSS Encoding and Carto Rendering

For the sake of making this document easy to read, each difference is described once. Only eight layers can be drawn in the same map. We can have a final rendering of the Daraa map with all the

layers.

8.2.1. *AgricultureSrf* Feature Class

Figure 40 shows a preview of the *AgricultureSrf* feature class. The CSS implementation of the Carto style file is available here: *AgricultureSrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/agriculturesrf/agriculturesrf.css].

Figure 40. AgricultureSrf layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:** The import of an SVG file is different (the SVG is smaller):
  - The polygon fill pattern command does not have any size attribute linked to the SVG file.
  - A solution can be to increase the size of the SVG symbol beforehand.

NOTE	An opacity and a gamma have been formalized to have a color more brownish.

8.2.2. *CulltureSrf* Feature Class

Figure 41 shows a preview of the *CulltureSrf* feature class. The CSS implementation of the Carto style file is available here: *CulltureSrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/culturesrf/culturesrf.css].

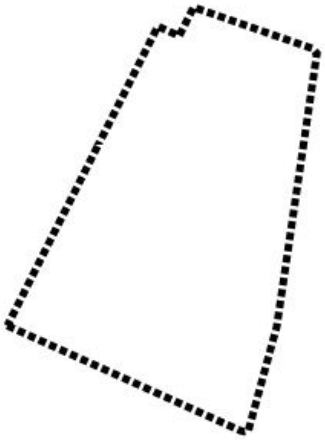


Figure 41. CultureSrf layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:**
  - With the utilization of the same stroke and dash size, we notice that the dashes are closer and smaller than the GeoServer rendering.
  - The line is stronger (*cf.* Challenges in Styling Feature Classes).

8.2.3. *HydrographyCrv* Feature Class

Figure 42 shows a preview of the *HydrographyCrv* feature class. The CSS implementation of the Carto style file is available here: *HydrographyCrv* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/hydrographycrv/hydrographycrv.css].

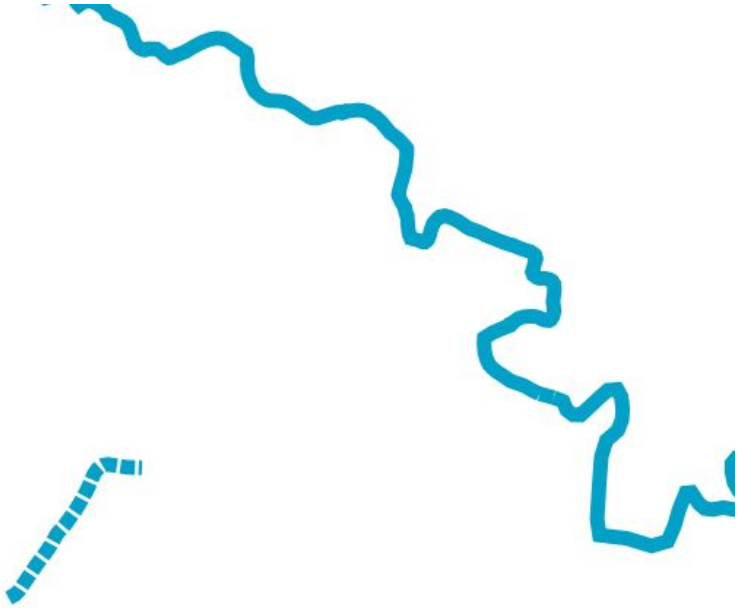


Figure 42. HydrographyCrv layer preview rendered by Carto using guideline style in CartoCSS

8.2.4. *HydrographySrf* Feature Class

Figure 43 shows a preview of the *HydrographySrf* feature class. The CSS implementation of the Carto style file is available here: *HydrographySrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/hydrographysrf/hydrographysrf.css].

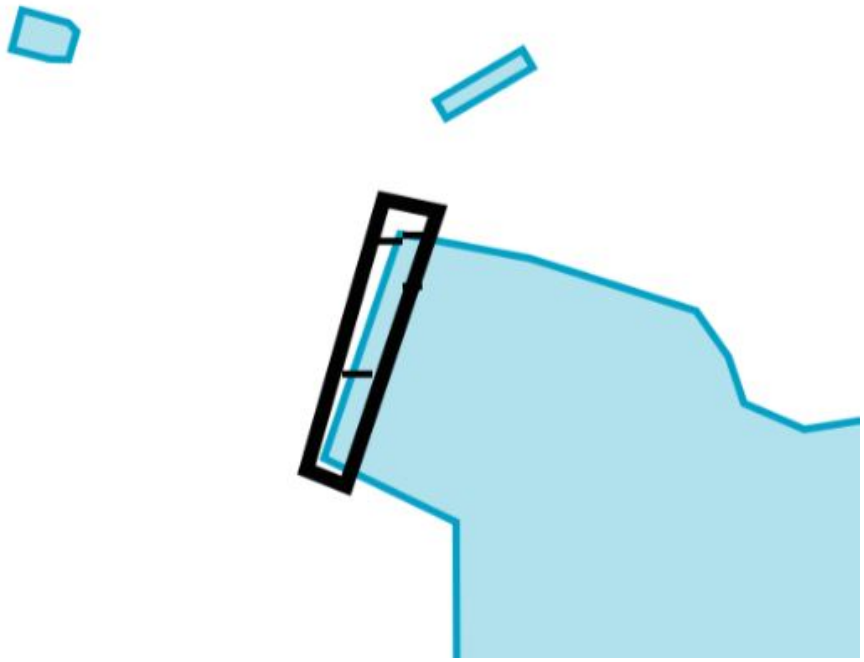


Figure 43. HydrographySrf layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:**
  - The SVG marker file import on a polygon has an unexpected behavior. We can see in the above figure that four points from the SVG import are not on the node or the arc.
  - A width size has been used on the marker geometry but if we applied a height size to get a square the symbol vanish. The handling of the width and height parameter is a bit odd for markers. If only the width (or the height) is defined, the marker is scaled up/down uniformly (ratio 1:1). Nevertheless, if height and width are described, the marker is scaled with the ratio (`height:width`).

NOTE	The idea of *Z-order* exists in CartoCSS. The concept was implemented with the passes system. During the rendering, you can select a line to render after the first pass using "::name". Here, we render all the lakes and at the second pass, we render the dam symbology.

8.2.5. *InformationPnt* Feature Class

Figure 44 shows a preview of the *InformationPnt* feature class. The CSS implementation of the Carto style file is available here: *InformationPnt* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/informationpnt/informationpnt.css].

Figure 44. InformationPnt layer preview rendered by Carto using guideline style in CartoCSS

NOTE	It is not possible to extract the name from a regular expression. We have to manually create (only 5 inputs into the data) a new column which contains the toponym.

8.2.6. *RecreationSrf* Feature Class

Figure 45 shows a preview of the *RecreationSrf* feature class. The CSS implementation of the Carto style file is available here: *RecreationSrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/recreationsrf/recreationsrf.css].

Figure 45. RecreationSrf layer preview rendered by Carto using guideline style in CartoCSS

8.2.7. *SettlementSrf* Feature Class

Figure 46 shows a small preview of the *SettlementSrf* feature class. The CSS implementation of the Carto style file is available here: *SettlementSrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/SettlementSrf/SettlementSrf.css].

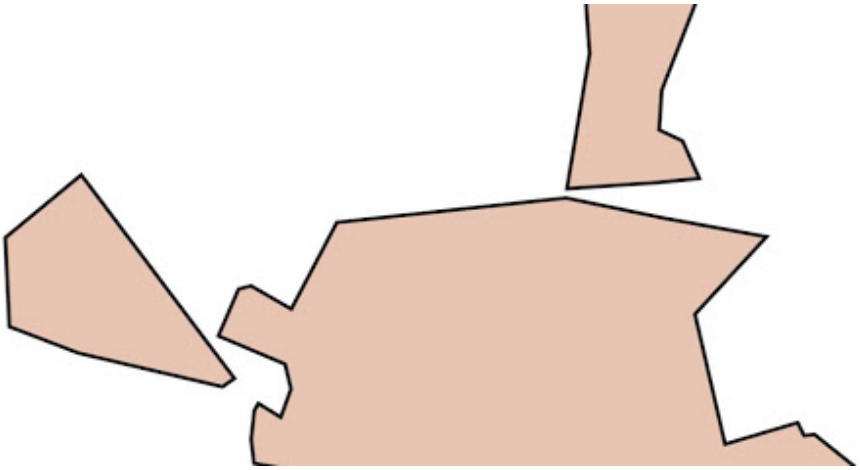


Figure 46. HydrographyCrv layer preview rendered by Carto using guideline style in CartoCSS

8.2.8. *StructurePnt* Feature Class

Figure 47 shows a small preview of the *StructurePnt* feature class. The CSS implementation of the Carto style file is available here: *StructurePnt* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/structurepnt/structurepnt.css].

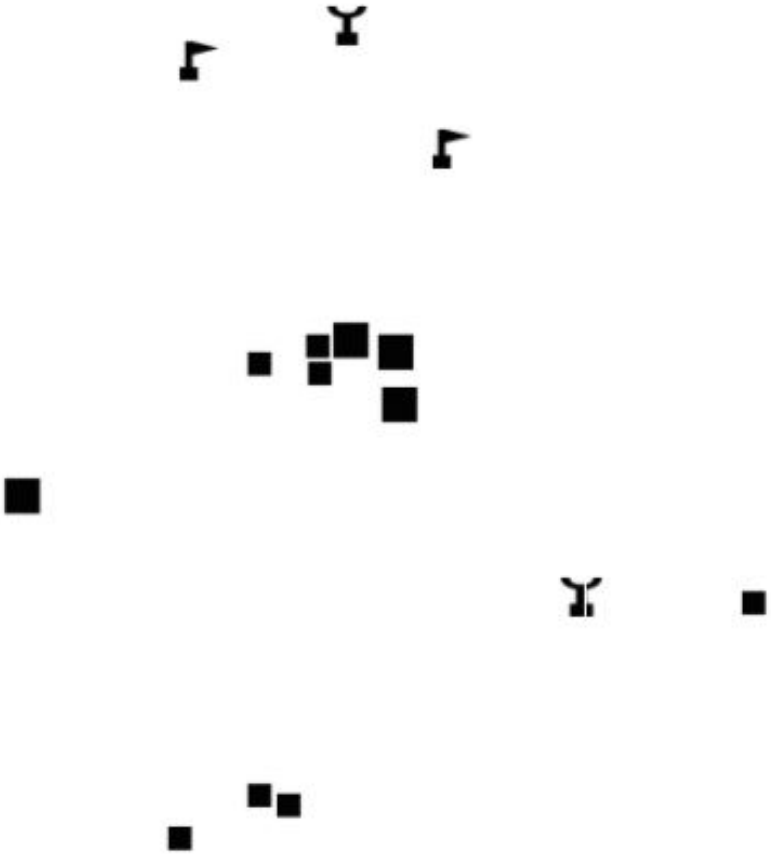


Figure 47. StructurePnt layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:**
  - The size used in GeoServer is in ground units and it does not suit the CartoCSS rendering.

Some visual modifications have been done into the SVG size.

- The top icon from the mosque and the school is not rendered. The SVG files have been modified. Those provided in the data set were not a version handle by CartoCSS.

NOTE	Only a width size has been applied, and the height size does not seem to change anything.

8.2.9. *TransportationGroundCrv* Feature Class

Figure 48 shows a small preview of the *TransportationGroundCrv* feature class. The CSS implementation of the Carto style file is available here: *TransportationGroundCrv* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/TransportationGroundCrv/TransportationGroundCrv.css].

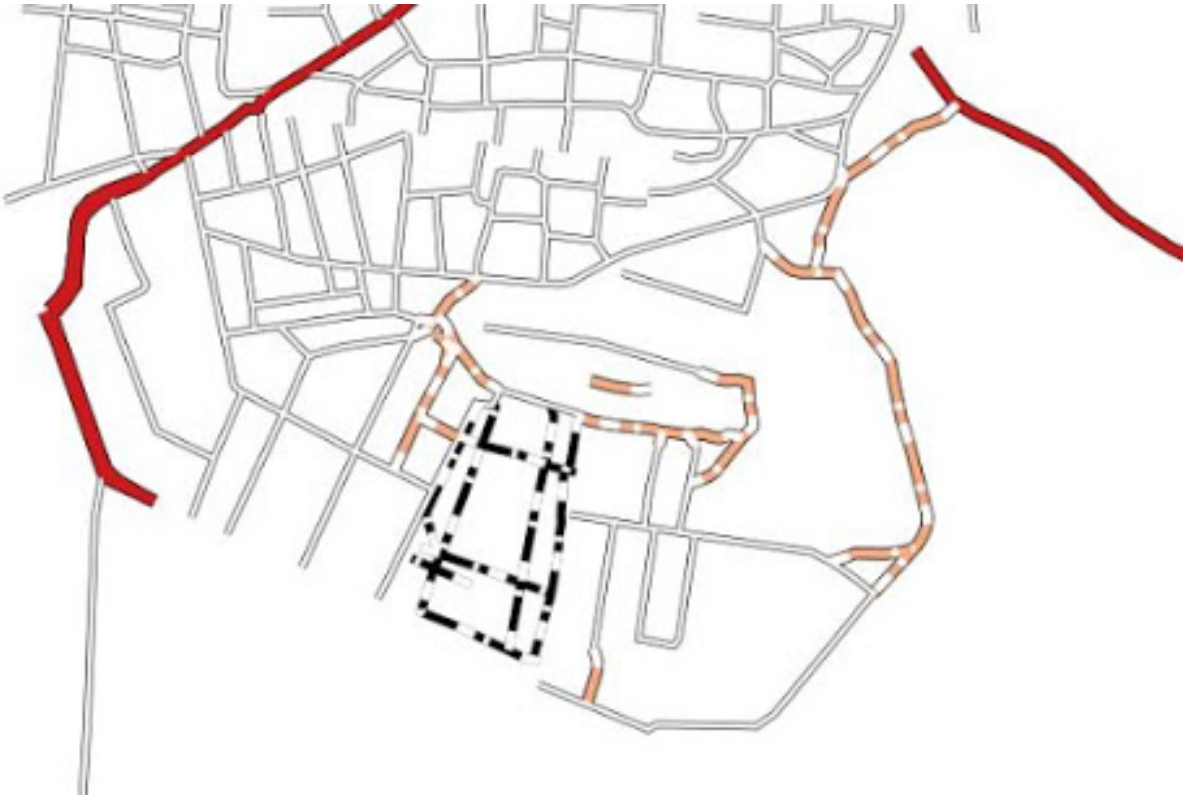


Figure 48. TransportationGroundCrv layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:**
  - A regular expression has been used for the road type separation (primary, secondary, trunk and *etc.*). GeoServer uses the RIN_ROI column. In CartoCSS, some roads are missing because of this.

8.2.10. *UtilityInfrastructureCrv* Feature Class

Figure 49 shows a small preview of the *UtilityInfrastructureCrv* feature class. The CSS implementation of the Carto style file is available here: *UtilityInfrastructureCrv* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/utilityinfrastructurecrv/utilityinfrastructurecrv.css].

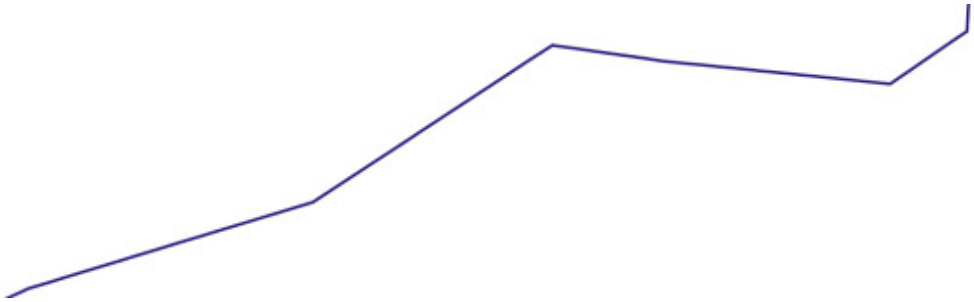


Figure 49. UtilityInfrastructureCrv layer preview rendered by Carto using guideline style in CartoCSS

8.2.11. *UtilityInfrastructurePnt* Feature Class

Figure 50 shows a small preview of the *UtilityInfrastructurePnt* feature class. The CSS implementation of the Carto style file is available here: UtilityInfrastructurePnt CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/UtilityInfrastructurePnt/UtilityInfrastructurePnt.css].

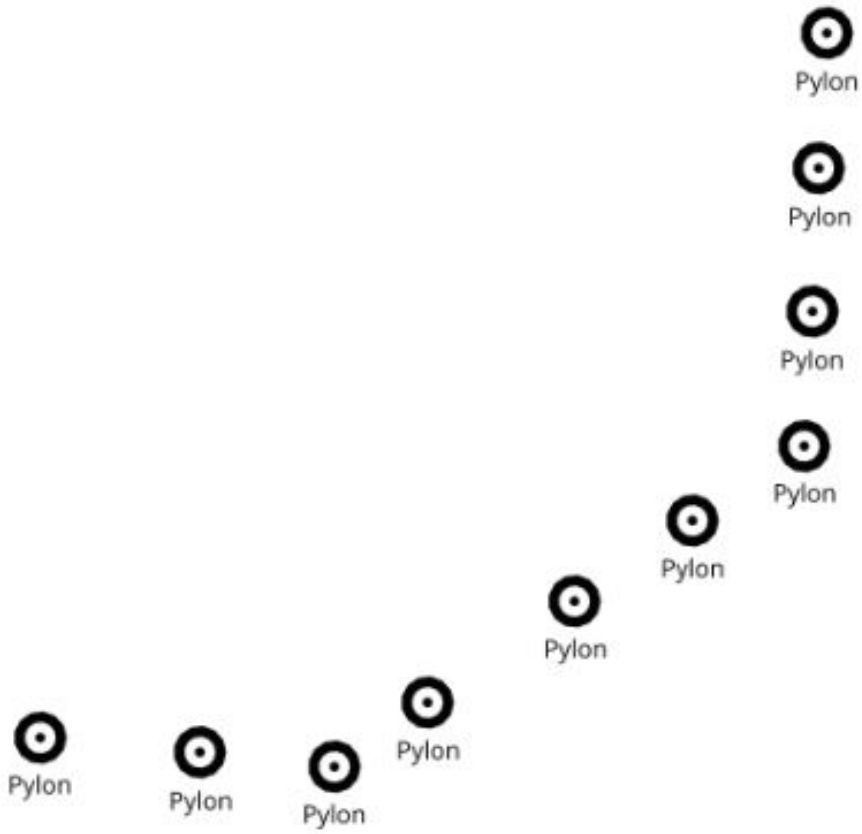


Figure 50. UtilityInfrastructurePnt layer preview rendered by Carto using guideline style in CartoCSS

- **Comparison with GeoServer:**
  - The labels have been placed below the marker via a "dy" (vertical) translation. The label's distance from the symbol change with the scale change in GeoServer. This behavior does not exist in CartoCSS.
  - In GeoServer, the "Pylon" labels are shown in a different scale than the symbol. It should then be defined as toponyms. If this is the case, placing a label below its POI is considered by the cartographic guidelines of map creation as the less meaningful position.

NOTE

The guidelines provided does not describe fully how to render the "Pylon" label. This leads to two potential understandings. In our first comprehension "Pylon" is part of a composite symbol made by the circular shape and the label. In this case, the two parts of the symbol have to be rendered together as one element, one part cannot exist without the other. Furthermore, the distance between the two elements has to be the same. We chose this understanding for our rendering. The second comprehension is to render "Pylon" label as a toponym. In that case the label is an additional information to the symbol. A toponym should follow the cartographic guidelines and be placed preferably on the top right of the symbol.

8.2.12. *VegetationSrf* Feature Class

Figure 51 shows a small preview of the *VegetationSrf* feature class. The CSS implementation of the Carto style file is available here: *VegetationSrf* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/vegetationsrf/vegetationsrf.css].

Figure 51. VegetationSrf layer preview rendered by Carto using guideline style in CartoCSS

8.2.13. *MIL-STD-2525* Feature Class

Figure 52 shows a small preview of the *MIL-STD-2525* feature class. The CSS implementation of the Carto style file is available here: *MIL-STD-2525* CartoCSS [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/CartoCSS/MIL-STD2525D/MIL-STD2525D.css].

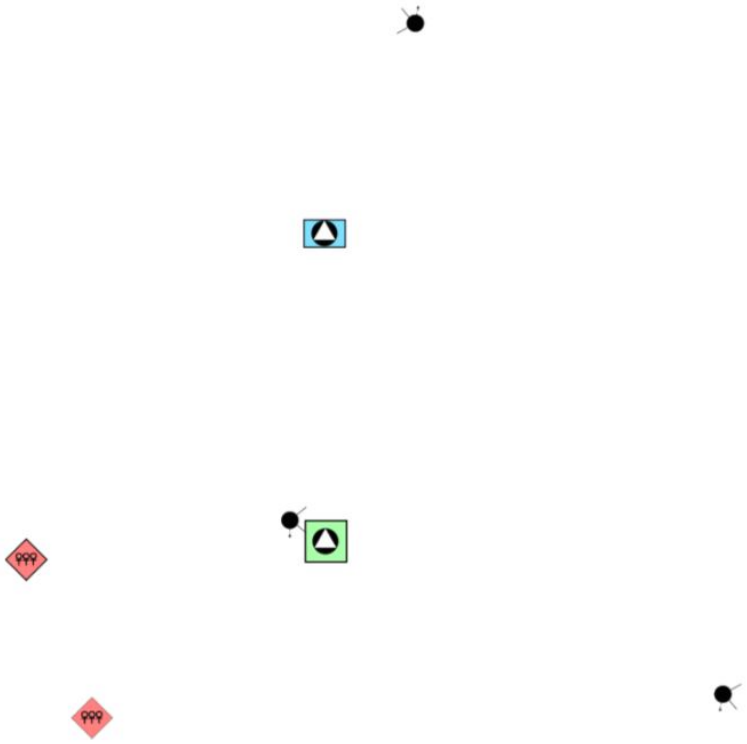


Figure 52. MIL-STD-2525 layer preview rendered by Carto using guideline style in CartoCSS

8.3. Portrayal Support using JSON Encoding and Mapbox Studio Rendering

The Mapbox solution is data-driven. The data has to be pre-processed and filtered before using them as a layer. For all attached JSON style description files, the meta-header of the symbology has been deleted. Also, the data should be pre-processed and rearranged to suit the style rendering. Our desire to use raw data to understand better the engine has not been fulfilled due to Mapbox technical limits. We used the Feature Manipulation Engine (FME) to reorder data before applying the styling rules.

Figure 53 shows a small preview of all feature classes in different zoom levels.

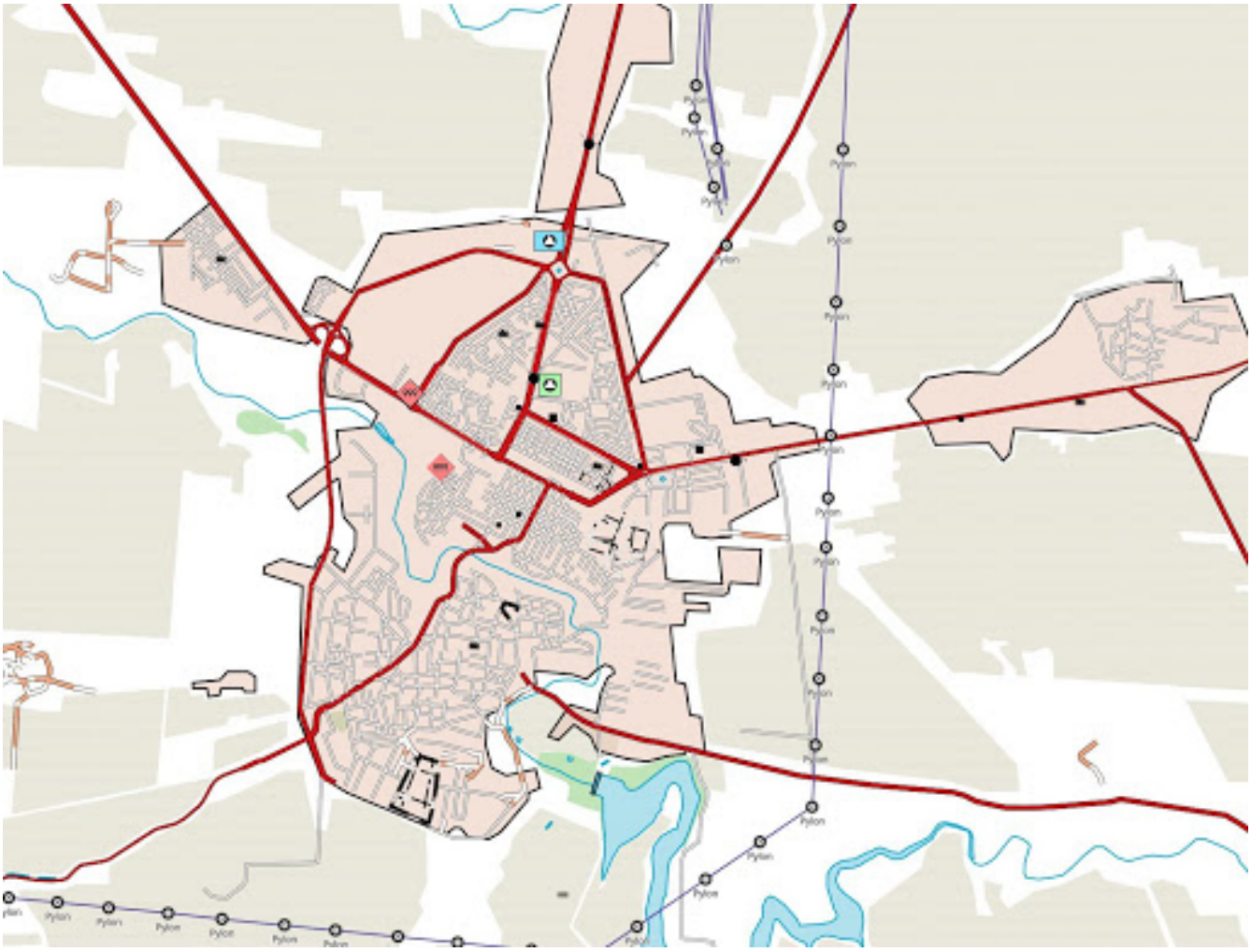


Figure 53. Preview of all the layers together rendered by Mapbox Studio using guideline style in JSON format at zoom level 12

8.3.1. *AgricultureSrf* Feature Class

Figure 54 shows a preview of the *AgricultureSrf* feature class. The JSON implementation of the Mapbox style file is available here: *AgricultureSrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/agriculturesrf/agriculturesrf.json].

Figure 54. AgricultureSrf layer preview rendered by Mapbox Studio using guideline style in JSON

- **Comparison with GeoServer:**
  - The SVG's size has been increased to get closer to GeoServer's rendering. Unfortunately, the quality of the SVG has decreased

NOTE	Two layers are needed to create this style: colour fill + pattern

8.3.2. *CulltureSrf* Feature Class

Figure 55 shows a preview of the *CulltureSrf* feature class. The JSON implementation of the Mapbox style file is available here: *CulltureSrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/culturesrf/culturesrf.json].

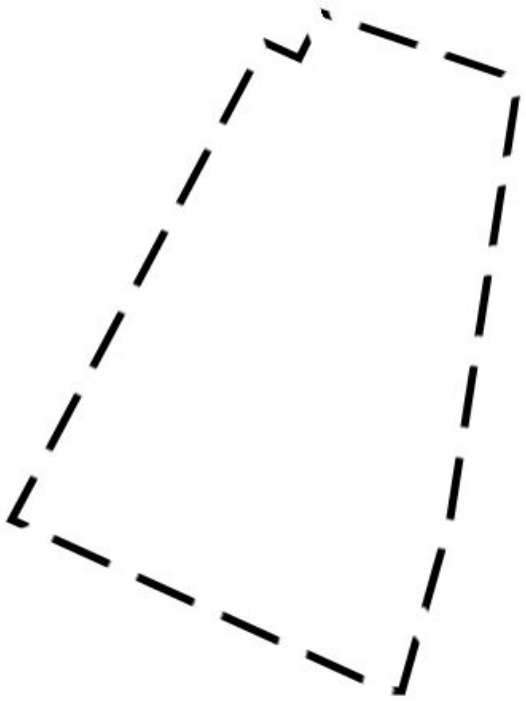


Figure 55. CulltureSrf layer preview rendered by Mapbox Studio using guideline style in JSON

- **Comparison with GeoServer:**
  - In Mapbox the Dashlines are recalculated depending the zoom level. The less the map is zoomed, the longer the dashes would be and vice versa.

8.3.3. *HydrographyCrv* Feature Class

Figure 56 shows a preview of the *HydrographyCrv* feature class. The JSON implementation of the Mapbox style file is available here: *HydrographyCrv* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/hydrographycrv/hydrographycrv.json].

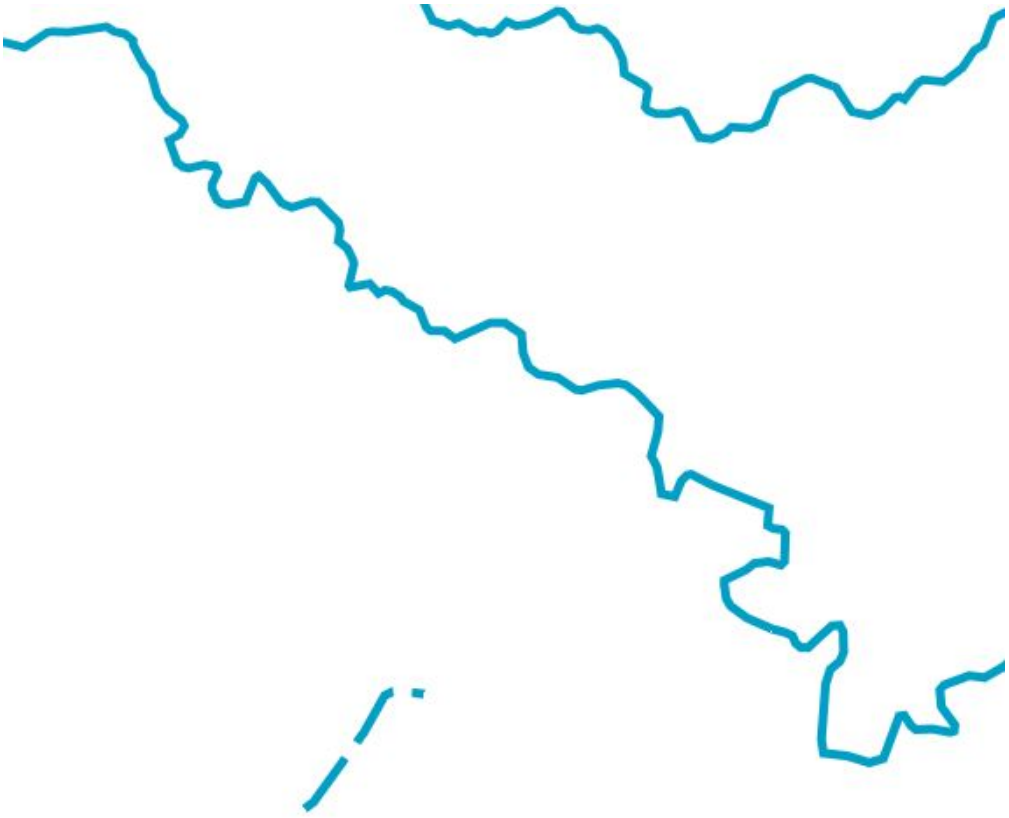


Figure 56. HydrographyCrv layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.4. *HydrographySrf* Feature Class

Figure 57 shows a preview of the *HydrographySrf* feature class. The JSON implementation of the Mapbox style file is available here: *HydrographySrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/hydrographysrf/hydrographysrf.json].

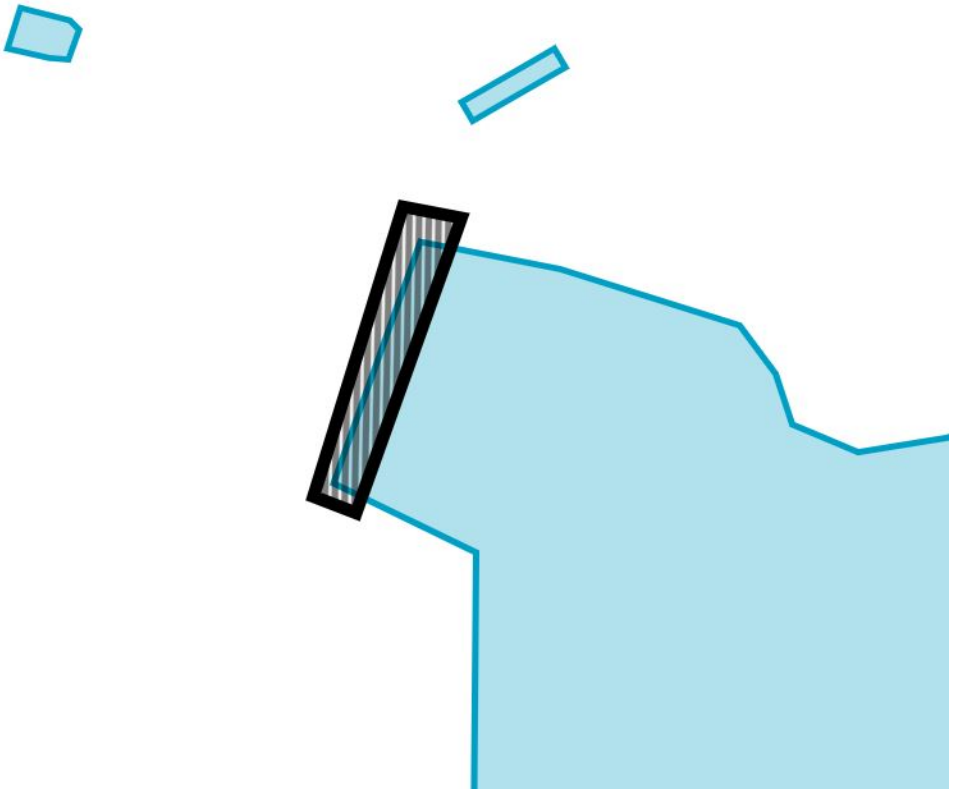


Figure 57. HydrographySrf layer preview rendered by Mapbox Studio using guideline style in JSON

NOTE	Four layers are needed to create this style: dam fill + dam line + lake line + lake fill

8.3.5. *RecreationSrf* Feature Class

Figure 58 shows a preview of the *RecreationSrf* feature class. The JSON implementation of the Mapbox style file is available here: *RecreationSrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/recreationsrf/recreationsrf.json].

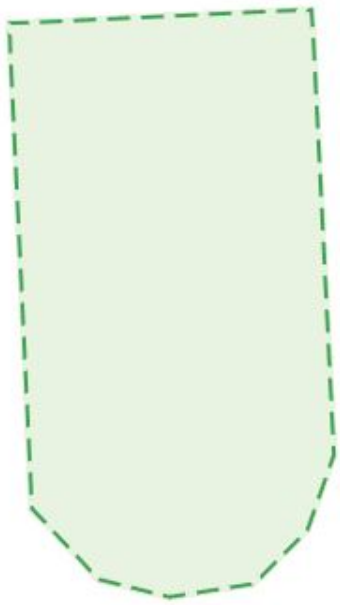


Figure 58. RecreationSrf layer preview rendered by Mapbox Studio using guideline style in JSON

NOTE	Four layers are needed to create this style: dam fill + dam line + lake line + lake fill

8.3.6. *SettlementSrf* Feature Class

Figure 59 shows a preview of the *SettlementSrf* feature class. The JSON implementation of the Mapbox style file is available here: *SettlementSrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/SettlementSrf/SettlementSrf.json].

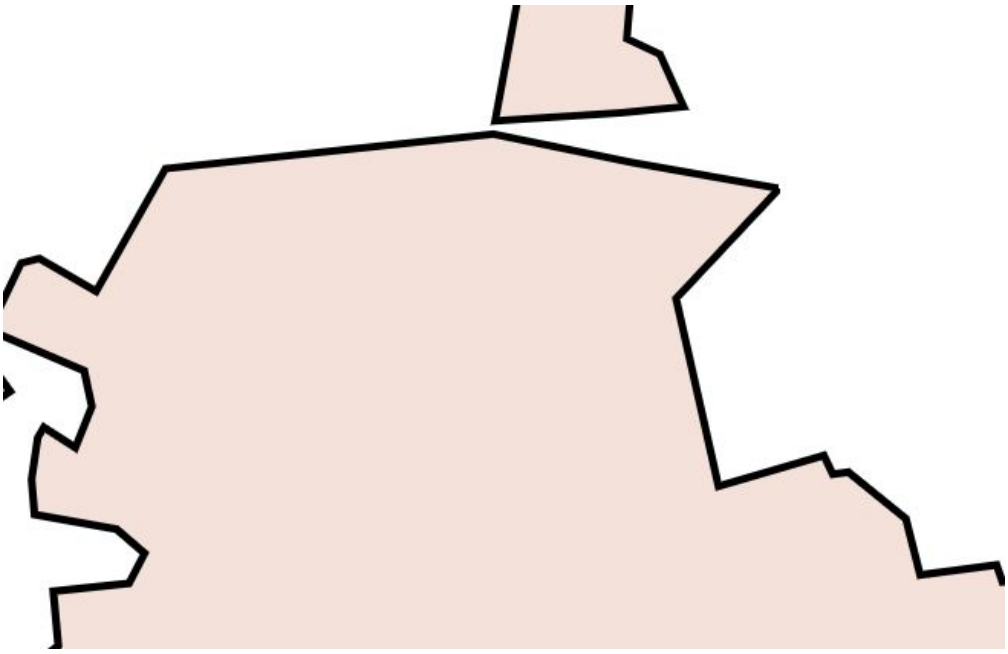


Figure 59. SettlementSrf layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.7. *StructurePnt* Feature Class

Figure 60 shows a preview of the *StructurePnt* feature class. The JSON implementation of the Mapbox style file is available here: *StructurePnt* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/structurepnt/structurepnt.json].

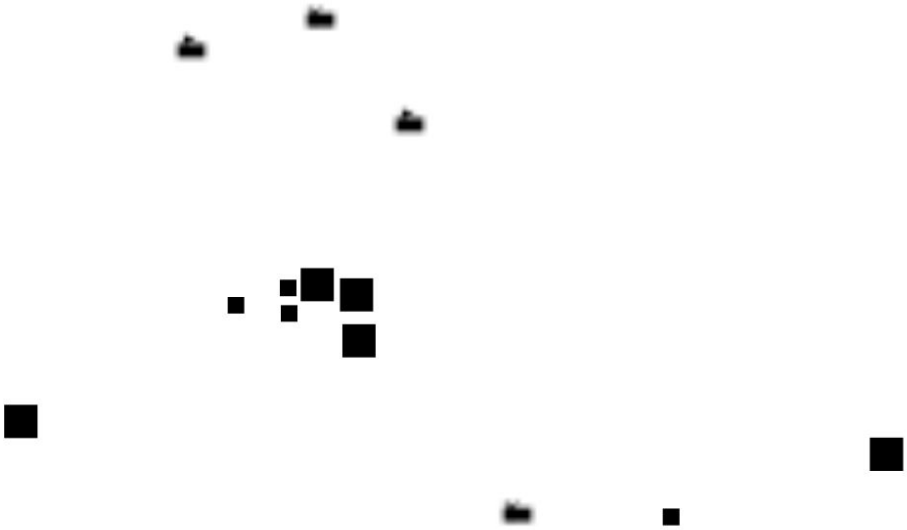


Figure 60. StructurePnt layer preview rendered by Mapbox Studio using guideline style in JSON

- **Comparison with GeoServer:**
  - The SVG's size has been increased to get closer to GeoServer's rendering. Unfortunately, the quality of the SVG has decreased

8.3.8. *TransportationGroundCrv* Feature Class

Figure 61 shows a preview of the *TransportationGroundCrv* feature class. The JSON implementation of the Mapbox style file is available here: *TransportationGroundCrv* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/TransportationGroundCrv/TransportationGroundCrv.json].

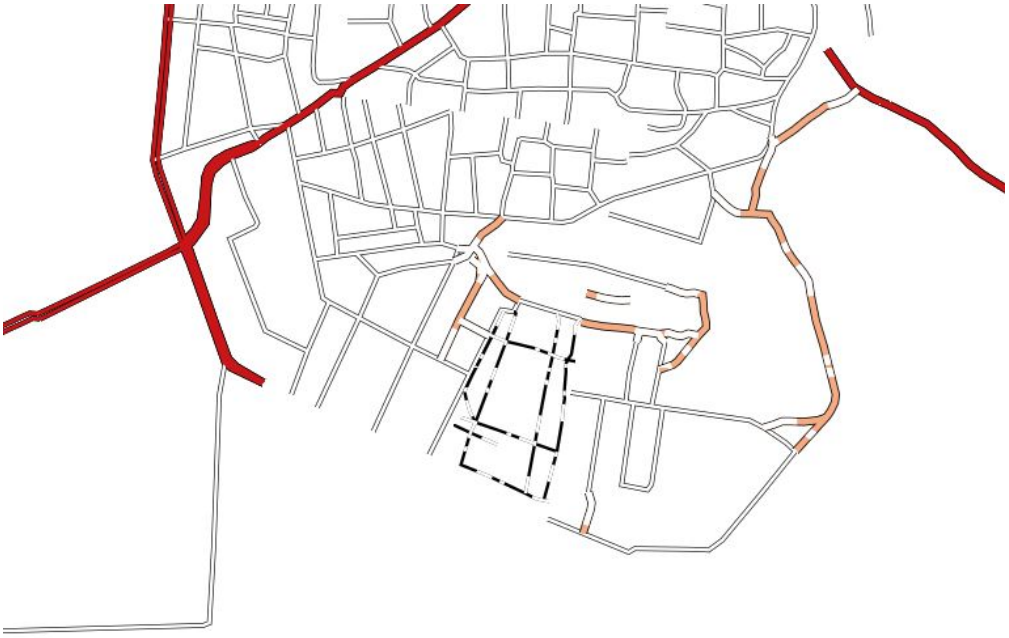


Figure 61. TransportationGroundCrv layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.9. *UtilityInfrastructureCrv* Feature Class

Figure 62 shows a preview of the *UtilityInfrastructureCrv* feature class. The JSON implementation of the Mapbox style file is available here: *UtilityInfrastructureCrv* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/utilityinfrastructurecrv/utilityinfrastructurecrv.json].

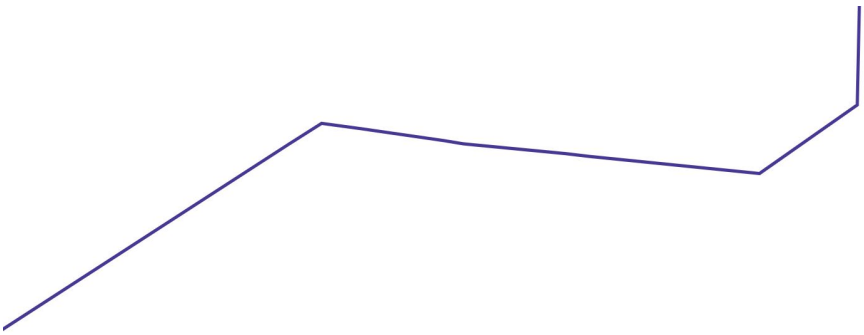


Figure 62. UtilityInfrastructureCrv layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.10. *UtilityInfrastructurePnt* Feature Class

Figure 63 shows a preview of the *UtilityInfrastructurePnt* feature class. The JSON implementation of the Mapbox style file is available here: *UtilityInfrastructurePnt* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/UtilityInfrastructurePnt/UtilityInfrastructurePnt.json].

Figure 63. UtilityInfrastructurePnt layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.11. *VegetationSrf* Feature Class

Figure 64 shows a preview of the *VegetationSrf* feature class. The JSON implementation of the Mapbox style file is available here: *VegetationSrf* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/vegetationsrf/vegetationsrf.json].

Figure 64. VegetationSrf layer preview rendered by Mapbox Studio using guideline style in JSON

8.3.12. *MIL-STD-2525* Feature Class

Figure 65 shows a preview of the *MIL-STD-2525* feature class. The JSON implementation of the Mapbox style file is available here: *MIL-STD-2525* JSON [https://github.com/MediaComem/ogc-testbed-14/blob/master/Daraa/Mapbox/MIL-STD2525D/MIL-STD2525D.json].

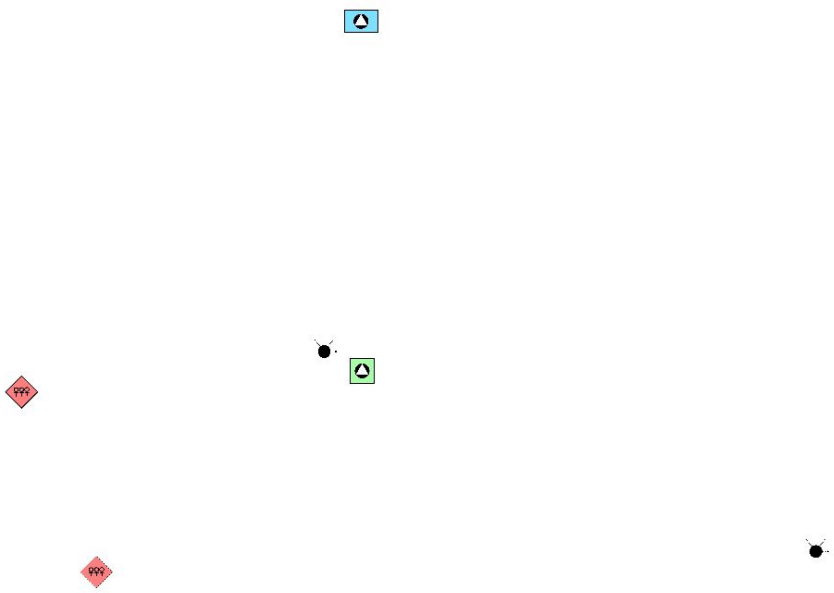


Figure 65. MIL-STD-2525 layer preview rendered by Mapbox Studio using guideline style in JSON

8.4. Insights on CartoCSS - Reverse Engineering

This reverse engineering exercise had the objective of defining a visual representation of the CartoCSS language to understand better how rendering works within the solution. We started from the documentation (available: here [https://carto.com/developers/styling/cartocss/]). In this document some technical terms are described:

- *Layer*: data set of the same type (points, lines, polygons, grids).
- *Symbolizer*: style applied to the layers (Polygon, Shield, Text, *etc.*). A symbolizer is composed of several parameters.

Polygon (polygon)	Line (lines and polygons)	Markers (points, lines, and polygons)
Shield (points and lines)	Line Pattern (lines and polygons)	Polygon Pattern (polygons)
Raster (grid data layers)	Point (points)	Text (points, lines, and polygons)
Building		

Figure 66. Symbolizers existing in CartoCSS (available: source [https://carto.com/developers/styling/cartocss/])

This figure shows the link between a layer (in parentheses) and a symbolizer (in green) with their relation. A symbolizer can be rendered on several layers (*e.g.*, Line Pattern on lines and polygons).

The next schema is the representation of the CartoCSS model which is created from the encoded name of each symbolizers. A parameter is defined as **line**-*width*. We assumed that **line** corresponds to the symbolizer and *width* is the parameter applied. From this hypothesis, the parameter **text**-**halo**-*fill* has been divided into a **text** symbolizer which can be composed of another symbolizer

halo which contains an attribute *fill*.

On this representation, +/- symbols in front of parameter names show if those attributes are shared (+) or not (-) with other symbolizers. For example, **marker** symbolizers can be composed of a **line** and use from this symbolizer *opacity* or *colour* but not *dasharray* or *offset*. In other words, **marker**-**line**-*opacity* exists, but **marker**-**line**-*dasharray* does not.

This figure demonstrates:

- In green: the symbolizers existing in the CartoCSS documentation;
- In yellow and red: extensions of those symbolizers. The colour difference describes shared and unshared parameters between symbolizers. For intense, both **polygon** and **line** can use a **pattern** extension; however, only **line** can render a **pattern**-*offset*.

This schema can be adopted as a footprint basis to be compared with other implementations or the conceptual model to highlight divergences and common points. Furthermore, the extensions of the CartoCSS are exposed in an easy to read representation.

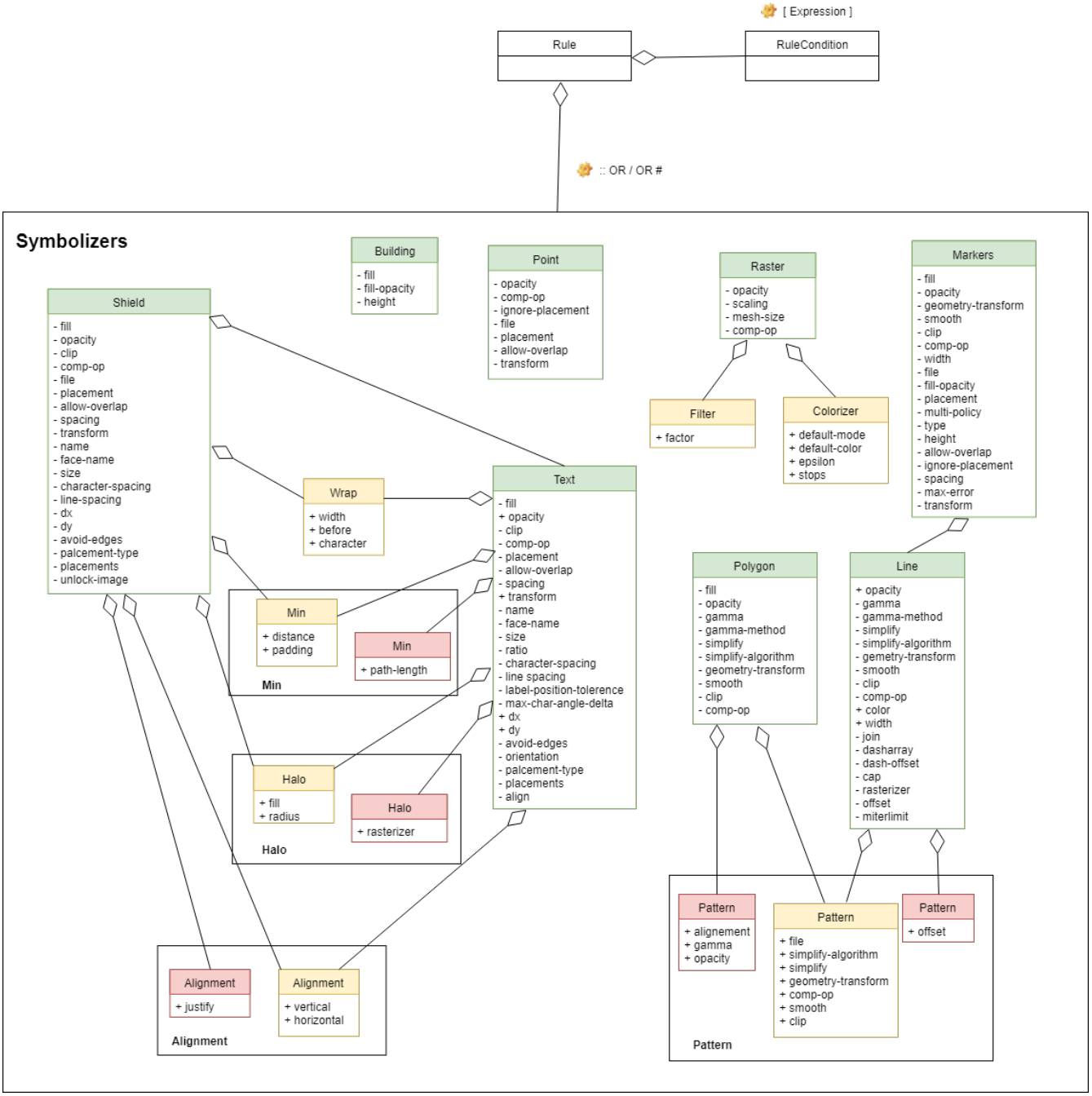


Figure 67. Reverse engineering schema of the CartoCSS engine

Chapter 9. WMS with Portrayal Support using Mapnik

A WMS that implements the conceptual model from D160. This section demonstrates that the conceptual model is portable to different mapping services and different encodings. In this case, the encoding is CartoCSS [https://github.com/mapbox/carto] that is translated into Mapnik [https://mapnik.org/] XML.

The data used in Testbed-14 is available as Shapefiles or Esri geodatabase files. As we are using Mapnik as our renderer, we chose to use Shapefiles. These data files are available in the OGC Portal.

9.1. Previewing the WMS

For testbed participants and observers, a production WMS is online at http://testbed.gswlab.ca:3000/service. This service is WMS 1.1.1 in "Node.js" library and renders map images on-the-fly using Mapnik (no caching).

Stylesheets were developed in CartoCSS and a live tile preview server (Kosmtik [https://github.com/kosmtik/kosmtik]) can be viewed at http://testbed.gswlab.ca:7000. This service is an OpenStreetMap-like tile service that is not a standard.

9.2. Styling Feature Classes

In the following subsections, we are going to explain provided styles for different feature classes using the CartoCSS encoding. Before starting, please note that GeoServer uses a continuous scale to render the Symbology. Unfortunately, both CartoCSS works on discontinuous zoom levels. To approximate the same visualization scale as GeoServer, we prepared a table to convert scales to zoom levels based on the conversion table applied by OSM [https://wiki.openstreetmap.org/wiki/Zoom_levels]. Figure 68 below illustrates our applied zoom levels instead of scales.

Scale	Zoom	Scale	Zoom
559,082,264	0	272,989	11
279,541,132	1	136,494	12
139,770,566	2	68,247	13
69,885,283	3	34,124	14
34,942,641	4	17,062	15
17,471,321	5	8,531	16
8,735,660	6	4,265	17
4,367,839	7	2,133	18
2,183,915	8	1,066	19
1,091,958	9	533	20
545,979	10	267	21
559,082,264	0	133	22

Figure 68. Conversion table between different scale and zoom levels

The stroke size used in GeoServer is in ground units. Unfortunately, Mapnik and CartoCSS use pixel units to describe their line size. To simplify the process, we manually changed the size implemented

on GeoServer to be visually as close as possible.

NOTE	All CartoCSS styles can be found in the Appendix D document

9.2.1. *AgricultureSrf* Feature Class

This feature class is Polygon geometry. The style for this layer includes applied pattern (*Green_AP77_Fill.svg*) combined with a fill solid background `#F7EDED`.

Figure 69 shows a preview of the *AgricultureSrf* feature class. The style file is available here: *AgricultureSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/agriculturesrf.mss].

Figure 69. Style preview for guideline AgricultureSrf feature class (applied the SVG pattern rendered by Mapnik)

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is different from our rendering by CartoCSS in Mapnik. It seems that this difference stems from rendering an SVG pattern. The polygon fill pattern command does not have any size attribute linked to the SVG file. We tried reaching at a closer presentation by converting original SVG to SVG Tiny 1.1 and SVG Tiny 1.2. However, the problem did not completely solve and to reach the same presentation, it seems that the original SVG must be changed.
- **Comparison with Mapbox:** Rendering this layer by Mapbox has similar problems, and it is neither similar to our presentation by Mapnik, nor to the presentation made by GeoServer.

9.2.2. *CulltureSrf* Feature Class

This feature class is Polygon geometry. The style for this layer includes stroke with dasharray as described in guidelines and line width for the borders of the filtered features.

Figure 70 shows a preview of the *CulltureSrf* feature class. The style file is available here: *CulltureSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/culturesrf.mss].

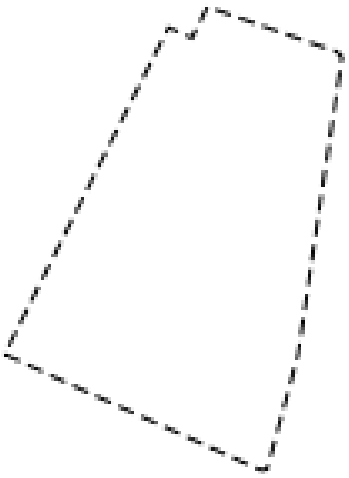


Figure 70. Style preview for guideline CultureSrf feature class (dasharray border rendered by Mapnik)

- **Comparison with GeoServer:** Renderings are the same for our CartoCSS and CSS used in GeoServer. We chose line-width: 1px instead of stroke-width: 5m in GeoServer.
- **Comparison with Mapbox:** Using the same stroke and dash size as GeoServer rendering, the dashes are closer and smaller than the GeoServer rendering.

9.2.3. *HydrographyCrv* Feature Class

This feature class is Line geometry. We used OSM tag waterway instead of *WCC* or *FCSUBTYPE* because those do not contain readable values to assign styles. All -*FCSUBTYPE* attributes are `100314`,

and *WCC* attributes values are `1`, `7` and `999`. We assume that the attribute value of `1` is corresponding to stream, `7` is corresponding to the river, and `999` is corresponding to drain. We applied a similar color of lake fill surface for lines. This change ensures visual continuity between streams lines and water surfaces at lower zoom levels.

Figure 71 shows a preview of the *HydrographyCrv* feature class. The style file is available here: *HydrographyCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/hydrographycrv.mss].

Figure 71. Style preview for guideline HydrographyCrv feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is different from our rendering. The `line width` and `line dasharray` seem to be different from our rendering.

A review has been made to the guideline styling to make it closer to OSM styling. Figure 72 shows a preview of the reviewed version of *HydrographyCrv* feature class styling. The style file is available at the following link: Reviewed *HydrographyCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/hydrographycrv_review.mss].

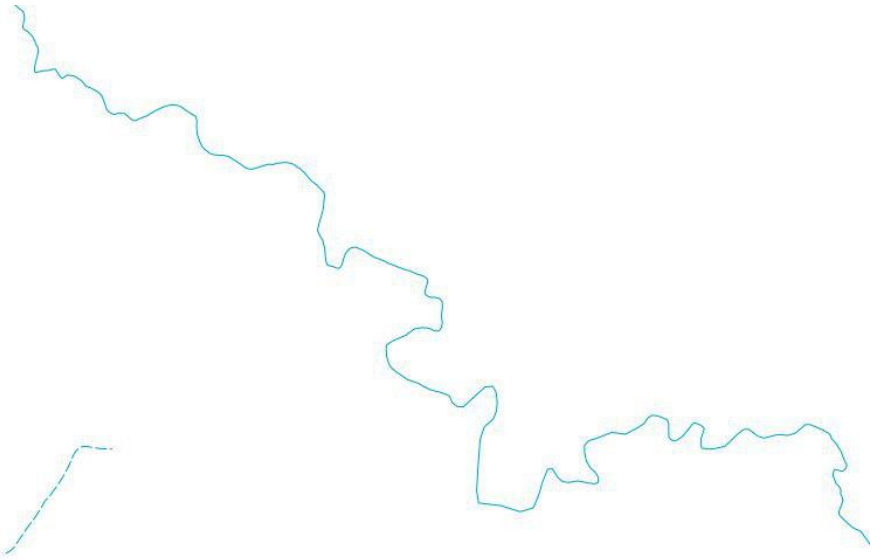


Figure 72. Style preview for reviewed HydrographyCrv feature class rendered by Mapnik

9.2.4. *HydrographySrf* Feature Class

This feature class is Polygon geometry. We applied Dam marker in the interior point of a polygon which shows dams border. We have two kinds of polygon features in this Shapefiles including dam and lake. We applied *Z*-order in CartoCSS to overlay dam area on lake polygons.

Figure 73 shows a preview of the *HydrographySrf* feature class. The style file is available here: *HydrographySrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/hydrographysrf.mss].

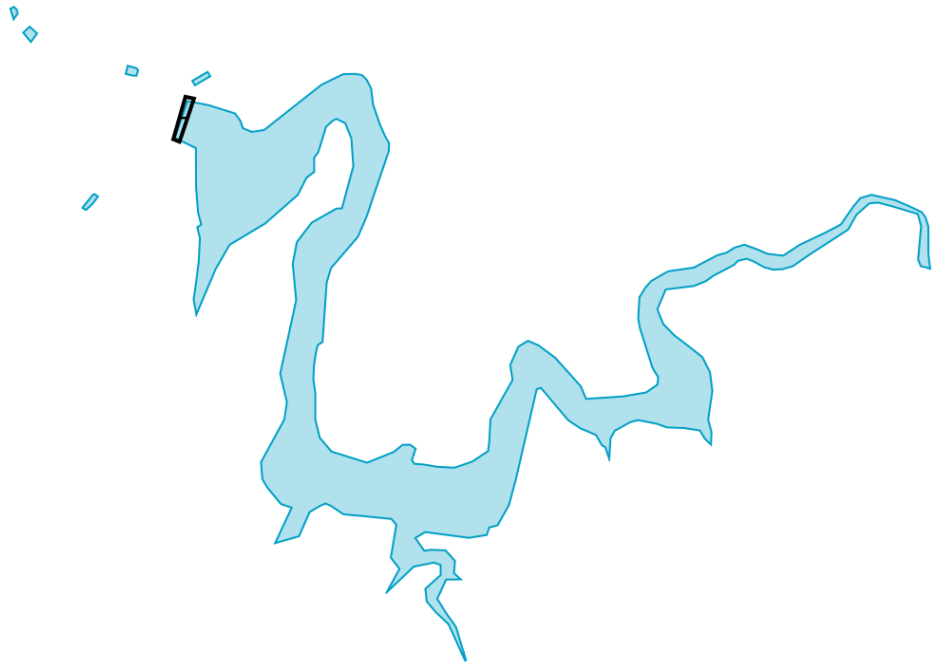


Figure 73. Style preview for guideline HydrographySrf feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is different from our rendering. Different markers are presented in the area of dam's polygon which is not desired.

A review has been made to the guideline styling to make it closer to OSM styling. Figure 74 shows a preview of the reviewed version of *HydrographySrf* feature class styling. The style file is available at the following link: Reviewed *HydrographySrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/hydrographysrf_review.mss].

Figure 74. Style preview for reviewed HydrographySrf feature class rendered by Mapnik

9.2.5. *InformationPnt* Feature Class

This feature class is Point geometry. The current style uses different possibilities of font families including *Open Sans*, *Noto*, *Hanazono*, *Unifont*, and *DejaVu*.

Figure 75 shows a preview of the *InformationPnt* feature class. The style file is available here: *InformationPnt* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/informationpnt.mss].

Figure 75. Style preview for guideline InformationPnt feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is similar to our rendering.

9.2.6. *RecreationSrf* Feature Class

This feature class is Polygon geometry. The current style simulates a polygon with line dasharray border that has an offset for zoom levels more than 13. Between zoom level 11 and 13, the polygons are filled by a pattern, so-called *sports_ground_wilson*.

Figure 76 shows a preview of the *RecreationSrf* feature class. The style file is available here: *RecreationSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/recreationsrf.mss].

Figure 76. Style preview for guideline RecreationSrf feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is almost similar to our rendering.

9.2.7. *SettlementSrf* Feature Class

This feature class is Polygon geometry. We removed stroke to blend surface with background and render this layer similar to the reviewed GeoServer rendered map.

Figure 77 shows a preview of the *SettlementSrf* feature class. The style file is available here: *SettlementSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/settlementsrf.mss].

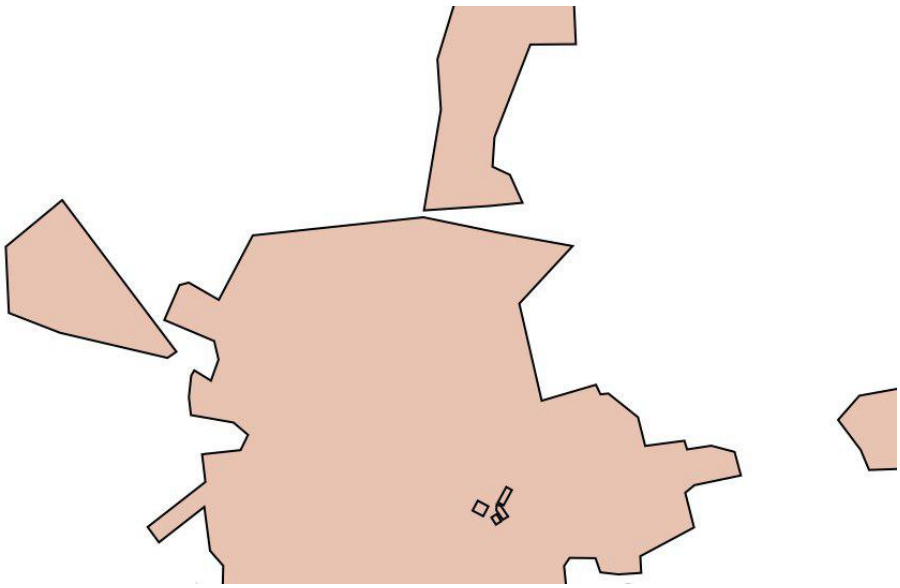


Figure 77. Style preview for guideline SettlementSrf feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is almost similar to our rendering except that they did not remove the line border in this layer (as considered in the reviewed version of the styles).

A review has been made to the guideline styling to make it closer to OSM styling. Figure 78 shows a

preview of the reviewed version of *SettlementSrf* feature class styling. The style file is available at the following link: Reviewed *SettlementSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/settlementsrf_review.mss].

Figure 78. Style preview for reviewed SettlementSrf feature class rendered by Mapnik

9.2.8. *StructurePnt* Feature Class

This feature class is Point geometry. We used amenity tag to assign desired markers for different types of buildings. Also the size of markers are changed based on different zoom levels.

Figure 79 shows a preview of the *StructurePnt* feature class. The style file is available here: *StructurePnt* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/structurepnt.mss].

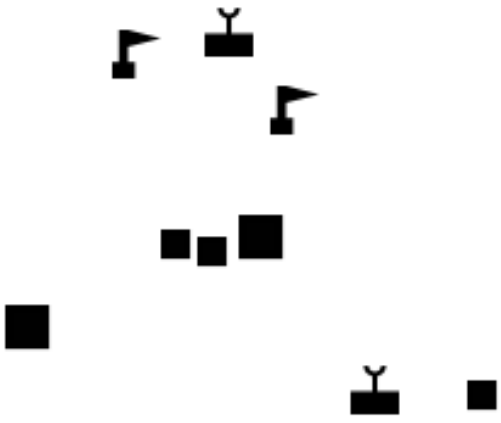


Figure 79. Style preview for guideline StructurePnt feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is also similar to our rendering.

9.2.9. *TransportationGroundCrv* Feature Class

This feature class is Line geometry. We are going to style this layer as cased lines which bare lines with borders. So, we applied line casing in implemented CartoCSS. Also, we defined different styles varying due to the different zoom levels for different types of line features. Three types of transportation lines can be recognized based on information included in `RIN_ROI` column of the attribute table. We have three attribute values for `RIN_ROI`: `3` is corresponding to "Primary Road", `4` is corresponding to "Secondary Roads", and `5` is corresponding to "Streets". We defined different styles to reach the best presentation of mentioned transportation lines based on different zoom levels. Applied a style with lighter colors to blend roads with background and have better contrast with elements on top (*e.g.*, marker). Non-primary roads now have a stroke color brighter than settlement surface but with the same gradient of color instead of white. This changes helps to highlight not primary roads in agriculture surfaces but blend them in settlement surfaces. To add their names as labels, we applied English names from` OSM_T_ENG` column and used "Sans" font family for these labels.

Figure 80 shows a preview of the *TransportationGroundCrv* feature class. The style file is available here: *TransportationGroundCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/transportationgroundcrv.mss].

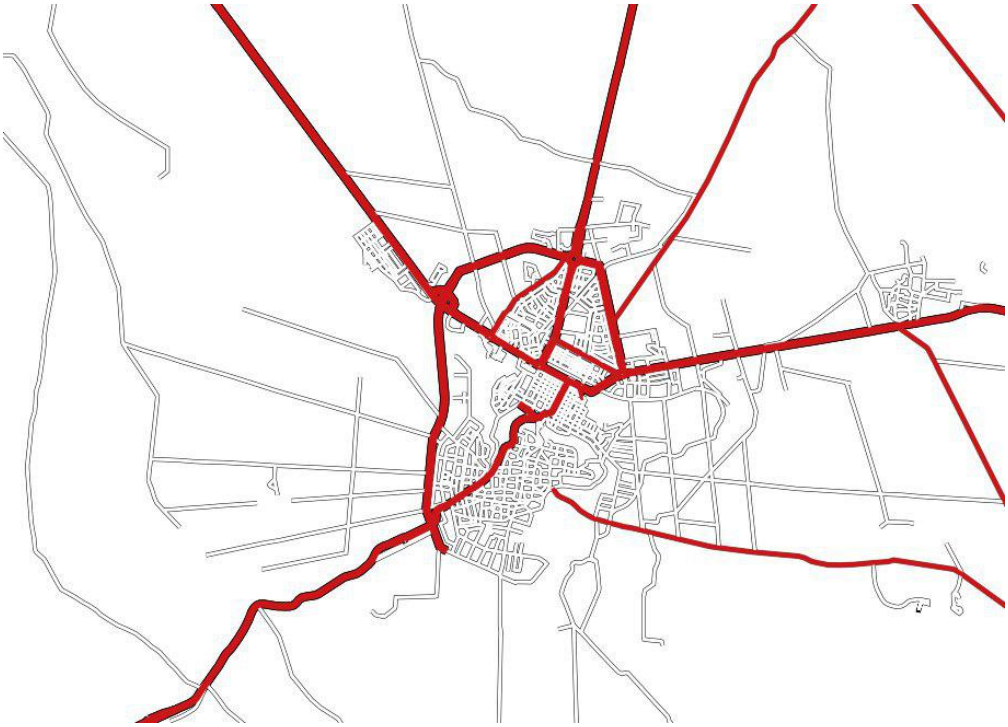


Figure 80. Preview for TransportationGroundCrv layer rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is not similar to our rendering.

A review has been made to the guideline styling to make it closer to OSM styling. Figure 81 shows a preview of the reviewed version of *TransportationGroundCrv* feature class styling. The style file is available at the following link: Reviewed *TransportationGroundCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/transportationgroundcrv_reviewed.mss].

Figure 81. Style preview for reviewed TransportationGroundCrv feature class rendered by Mapnik

9.2.10. *TransportationGroundCrv_highlight* Feature Class

This feature class is Line geometry. We are going to style this layer in such a way that show bridges as double border lines in higher zoom levels and cased lines in lower zoom levels. Due to the different line width of transportation lines, we should define the bridge's style at different zoom levels for different types of transportation layers. The details of "Bridge" styling is mentioned in a separate section.

Figure 82 shows a preview of the *TransportationGroundCrv_highlight* feature class. The style file is available here: *TransportationGroundCrv_highlight* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/transportationgroundcrv_highlight.mss].

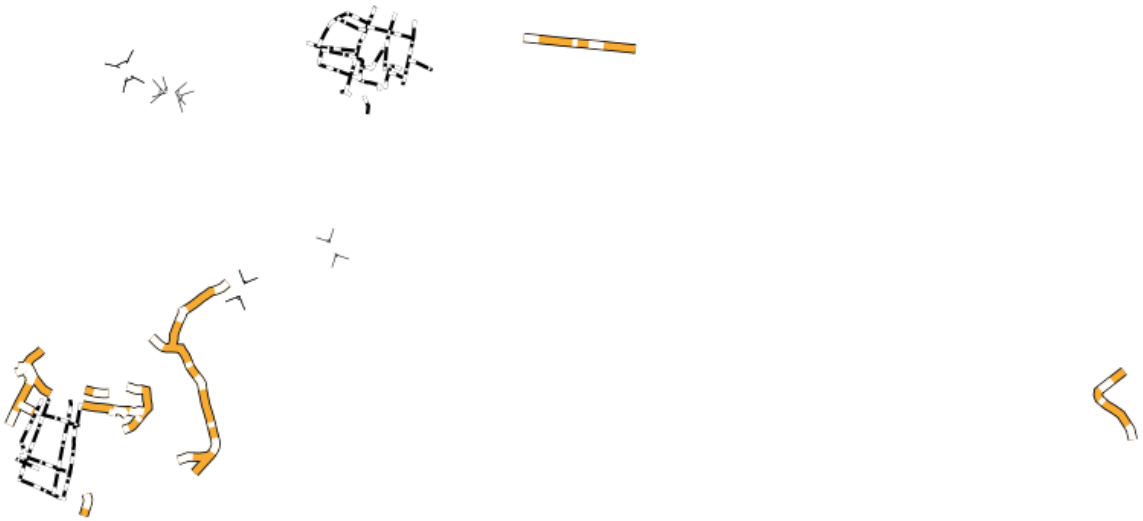


Figure 82. Style preview for guideline TransportationGroundCrv_highlight feature class

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.

- **Comparison with Mapbox:** This layer has not been rendered yet by Mapbox.

A review has been made to the guideline styling to make it closer to OSM styling. Figure 83 shows a preview of the reviewed version of *TransportationGroundCrv_highlight* feature class styling. The style file is available at the following link: Reviewed *TransportationGroundCrv_highlight* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/transportationgroundcrv_highlight_review.mss].

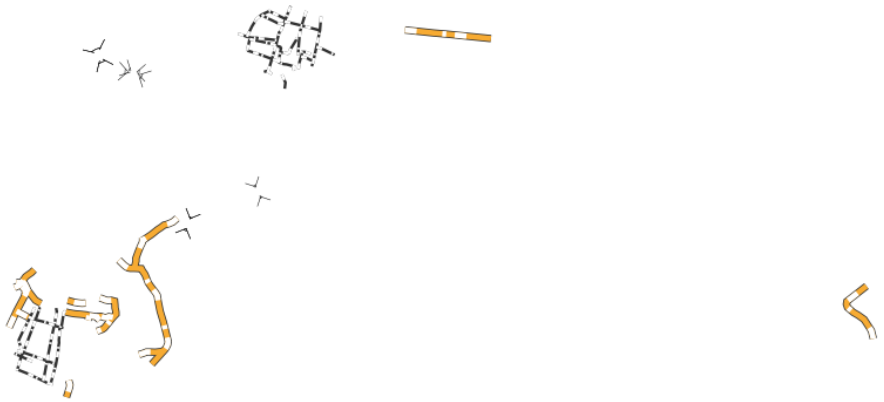


Figure 83. Style preview for reviewed TransportationGroundCrv_highlight feature class rendered by Mapnik

9.2.11. *UtilityInfrastructureCrv* Feature Class

This feature class is Line geometry. We used #333333 color for the line to obtain a high contrast.

Figure 84 shows a preview of the *UtilityInfrastructureCrv* feature class. The style file is available here: *UtilityInfrastructureCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/UtilityInfrastructureCrv.mss].

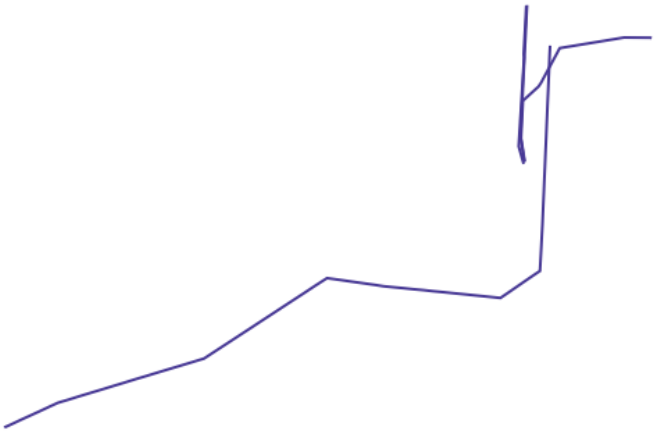


Figure 84. Style preview for guideline UtilityInfrastructureCrv feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to the

rendering by CartoCSS in Mapnik.

- **Comparison with Mapbox:** Rendering this layer by Mapbox is also similar to our rendering except for the line color which is a little bit lighter than our rendered lines.

A review has been made to the guideline styling to make it closer to OSM styling. Figure 85 shows a preview of the reviewed version of *TransportationGroundCrv_highlight* feature class styling. The style file is available at the following link: Reviewed *UtilityInfrastructureCrv* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/utilityinfrastructurecrv_reviewed.mss].

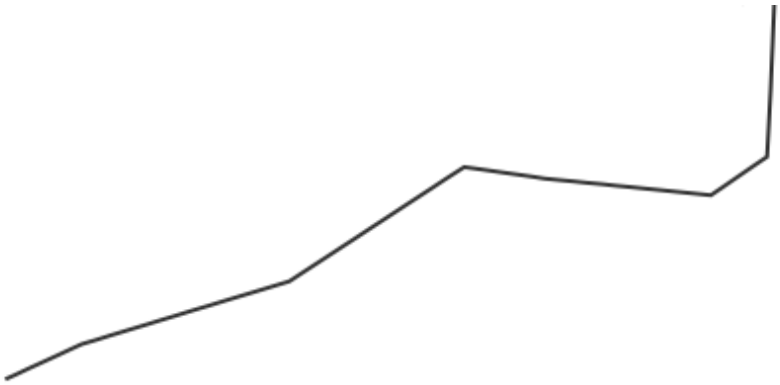


Figure 85. Style preview for guideline UtilityInfrastructureCrv feature class rendered by Mapnik

9.2.12. *UtilityInfrastructurePnt* Feature Class

This feature class is Point geometry. We applied labels different size based on zoom levels and choose Sans font family for them.

Figure 86 shows a preview of the *UtilityInfrastructurePnt* feature class. The style file is available here: UtilityInfrastructurePnt CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/utilityinfrastructurePnt.mss].

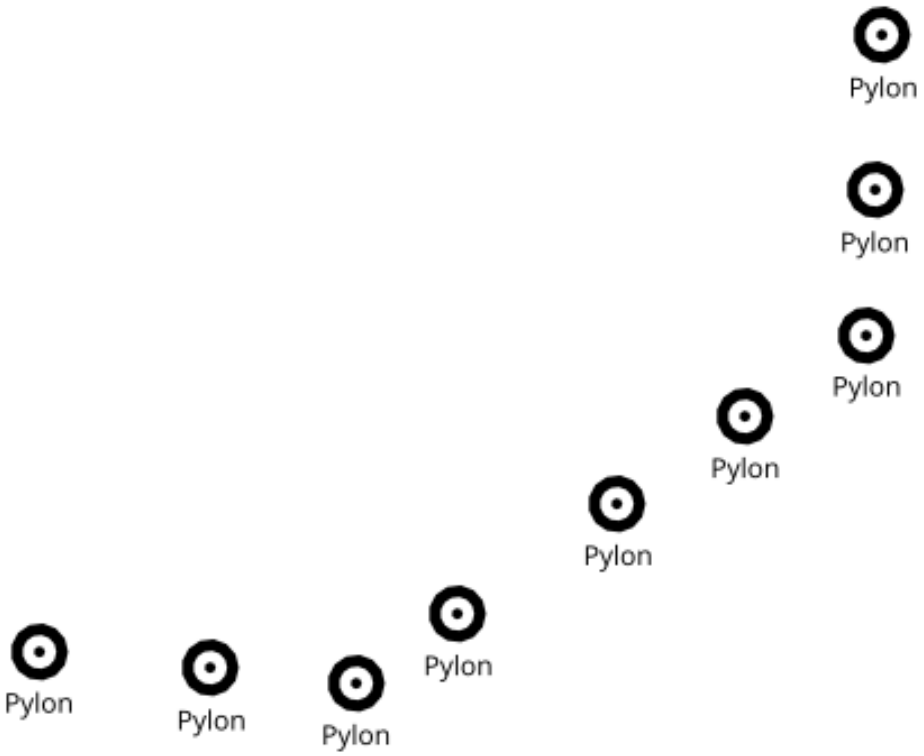


Figure 86. Style preview for guideline UtilityInfrastructurePnt feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to the rendering by CartoCSS in Mapnik. In GeoServer, we can use label-anchor to choose the part of the label which places over the point. In CartoCSS the similar task can be done by translation of label due to the point. So, we consider `dy=18` to place the middle point of label 18 pixels behind the point.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is similar to our rendering.

9.2.13. *VegetationSrf* Feature Class

This feature class is Polygon geometry. We used `#C2E4B9` color to fill the forest.

Figure 87 shows a preview of the *VegetationSrf* feature class. The style file is available here: *VegetationSrf* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/vegetationSrf.mss].

Figure 87. Style preview for guideline VegetationSrf feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is also similar to our rendering.

9.2.14. *Emergency* Feature Class

This feature class is Point geometry. We used different SVGs for different types of emergency points.

Figure 88 shows a preview of the *Emergency* feature class. The style file is available here: *Emergency* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/emergency.mss].

Figure 88. Style preview for guideline Emergency feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** This layer has not been rendered by Mapbox yet.

9.2.15. *MIL-STD-2525* Feature Class

This feature class is Point geometry. We used different SVGs which combined both symbols and frames to represent complex military icons. Also, we applied angle rotation for a particular type of military marker.

Figure 89 shows a preview of the *MIL-STD-2525* feature class. The style file is available here: *MIL-STD-2525* CartoCSS [https://github.com/GeoSensorWebLab/D162-WMS/blob/master/styles/separated_styles/milstandard2525.mss].

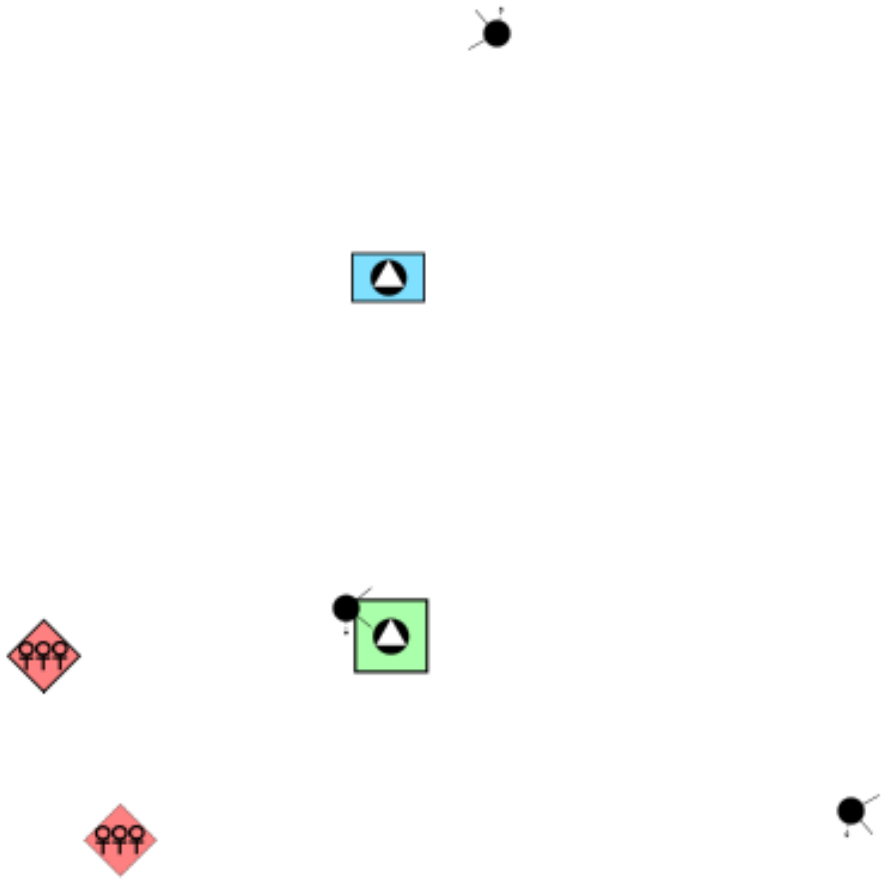


Figure 89. Style preview for guideline MIL-STD-2525 feature class rendered by Mapnik

- **Comparison with GeoServer:** Rendering of this layer by CSS used in GeoServer is similar to our rendering by CartoCSS in Mapnik.
- **Comparison with Mapbox:** Rendering this layer by Mapbox is also similar to our rendering.

9.2.16. All Feature Class Layers

Here below you can find the final results for both the styles closer to the original guidelines as well as the style we have revised a little to reuse some of the OSM stylings. The demo is available at this link [http://162.246.156.118:6789/styles/#14/32.6261/36.0873].

Figure 90 and Figure 91 show the previews of guideline styling at different zoom levels.

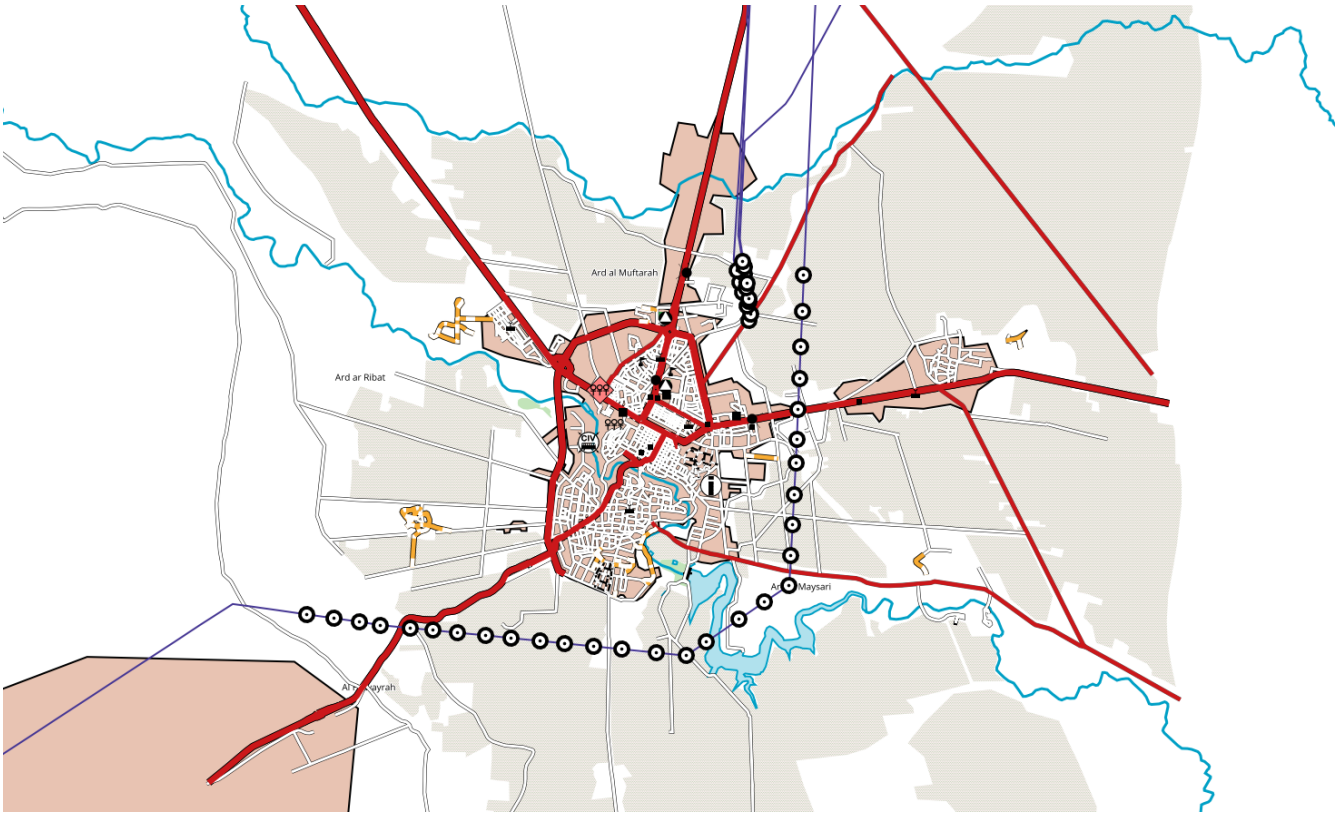


Figure 90. A preview of the guideline style for all the layers rendered by Mapnik at zoom level 13

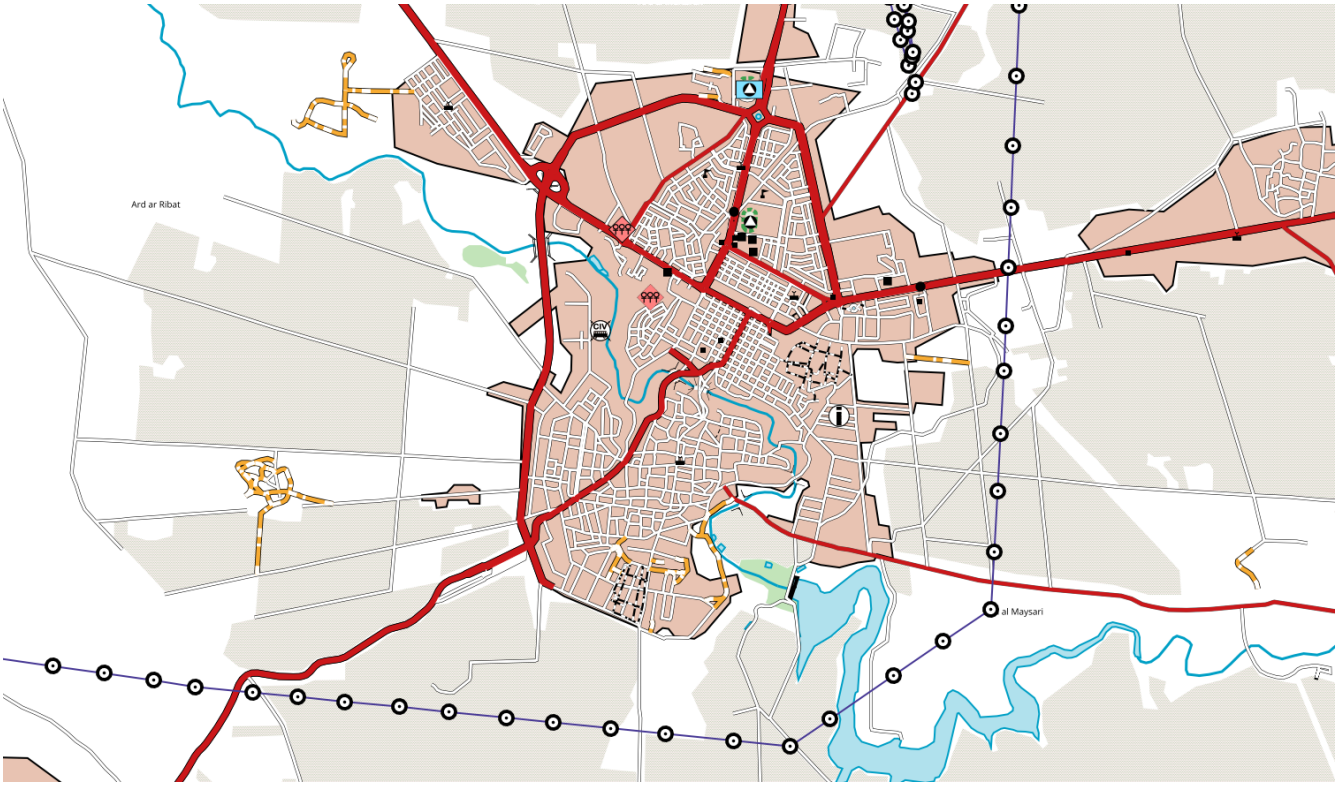


Figure 91. A preview of the guideline style for all the layers rendered by Mapnik at zoom level 14

Figure 92 and Figure 93 show the previews of guideline styling at different zoom levels.

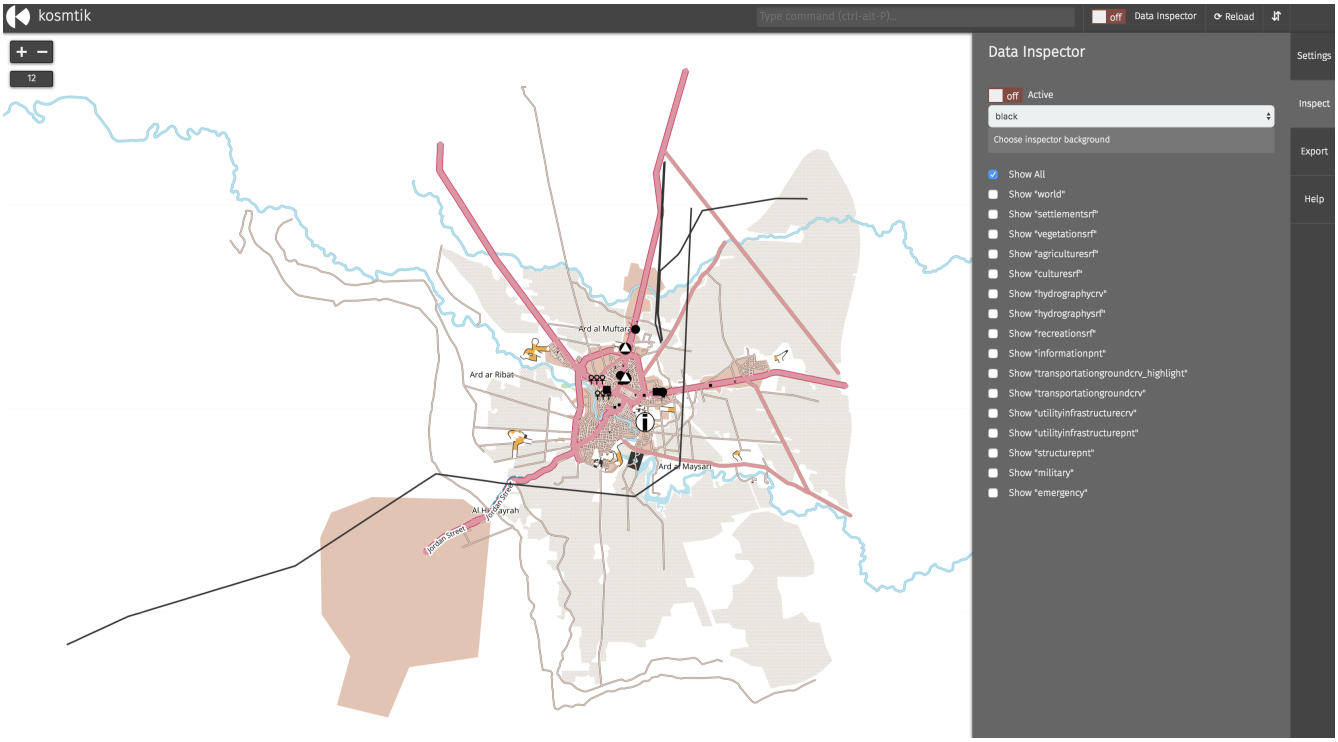


Figure 92. A preview of the reviewed style for all the layers rendered by Mapnik at zoom level 13

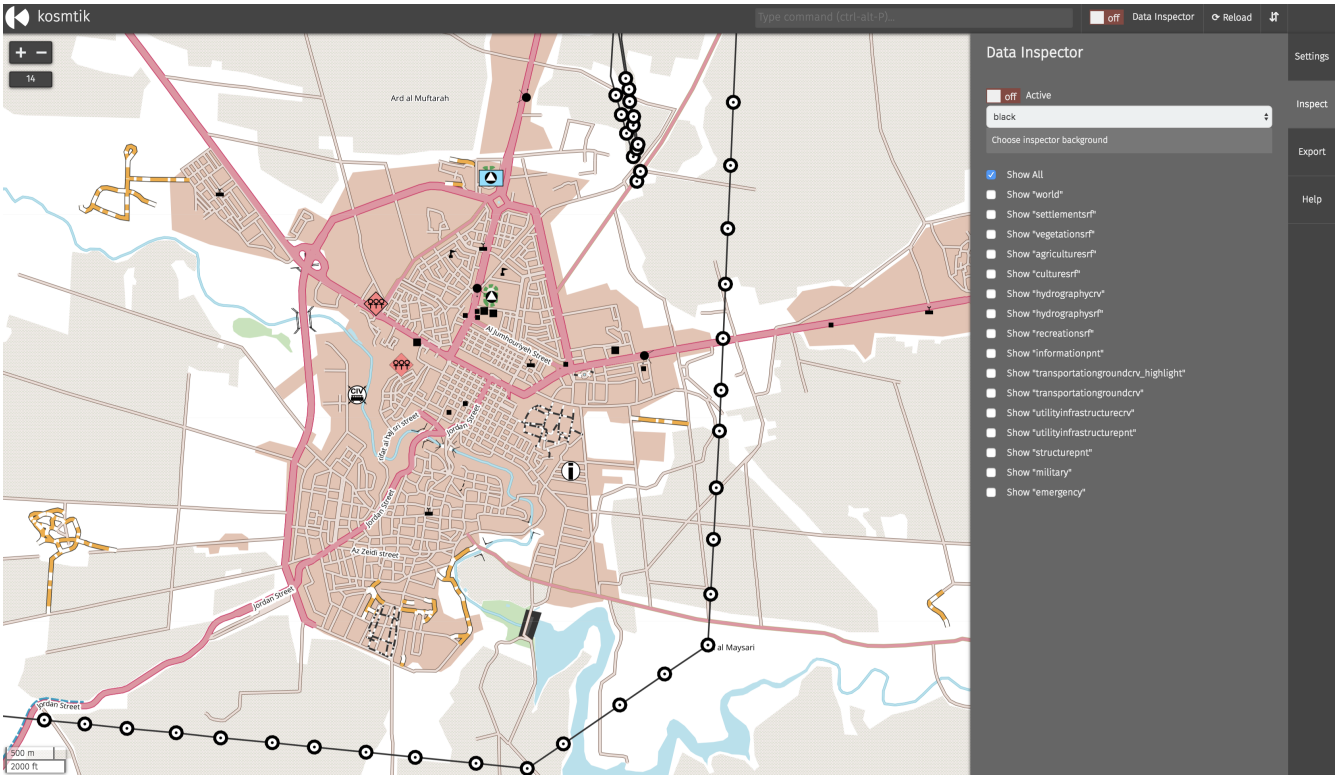
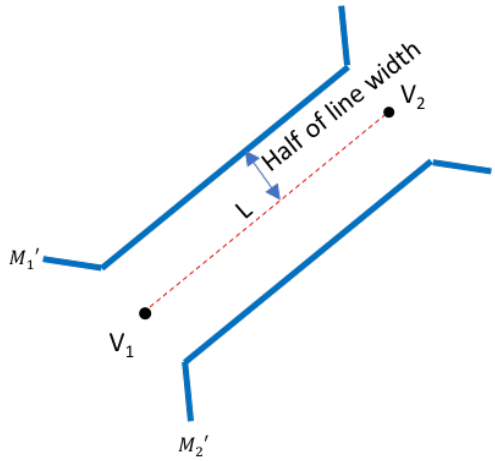


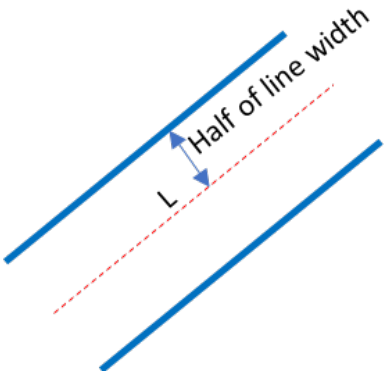
Figure 93. A preview of the reviewed style for all the layers rendered by Mapnik at zoom level 14

9.3. Bridge Style Tickmarks Challenge

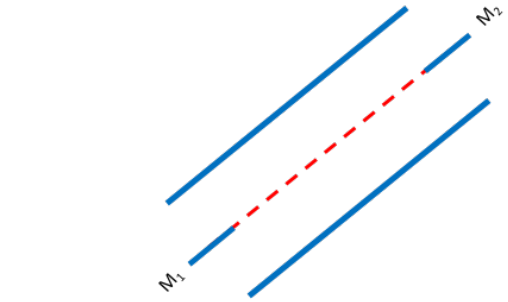
To style bridges in Daara dataset, at first, we separate their Shapefile layers from the *TransportationGroundCrv* based on those features which `F_CODE = AQ040` and `TRS = 13`. Then to style this layer, we present bridges as double border lines and hide the center lines which presented in their Shapefiles. For example, consider the figure below, the red dashed line is the bridges polyline in the Shapefile, but we have to present the bridge as the blue collection of the lines. We style bridges in six steps which is explained in Figure 94.



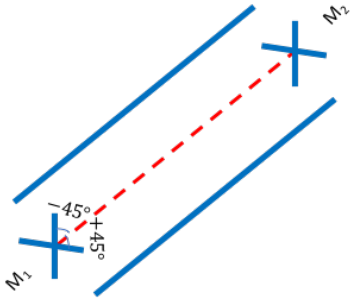
a) Bridge styling parameters description



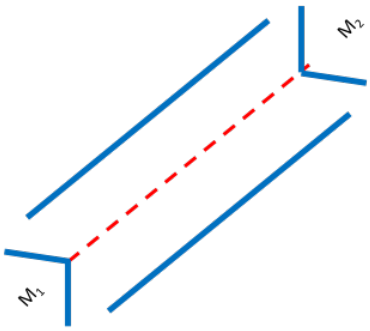
b) Step2 for bridge styling



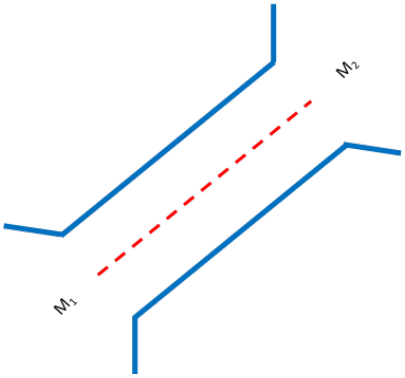
c) Step3 for bridge styling



d) Step4 for bridge styling



e) Step5 for bridge styling



f) Step6 for bridge styling

Figure 94. Preview of Bridge layer rendered by Mapnik at different zoom levels

- **Step 1:** Due to considered line-width in different levels which are half, one, and two pixels were considered. We created three different SVG files which present a horizontal line with length equal to 10 pixels and width equal to half, one, and two pixels. To be more specific, we used these SVG files as markers for inclining lines at start and end points of bridge's borders. He below, is the created SVG file for a horizontal line with `length= 10px` and `width = 1 px`.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 16.0.3, SVG Export Plug-In . SVG Version: 6.00 Build
0) -->
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1 Tiny//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
<svg version="1.1" baseProfile="tiny" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
<line x1="0" y1="0" x2="10" y2="0" style="stroke:rgb(0,0,0);stroke-width:1" />
</svg>
```

- **Step 2:** In this step, by using background and foreground features, we portray two parallel lines which define the border of our bridges. By using this solution, both parallel lines are rendered with different offsets from the main bridges' polylines which is half of the line width.
- **Step 3:** After presenting the border lines of bridges, we should place our SVG markers at the

start and end point of bridges line. We present our markers as M1 and M2. Note that the middle point of SVG marker placed at the first and end point of the central line.

- **Step 4:** At this step, we rotate each SVG markers and present two different markers in each start and end point. One of them has the angle equal to +45 degrees from the center line and the angle of other marker equals to -45 degrees.
- **Step 5:** After rotating markers, we have to transfer them to place the start point of each marker on the start point of the center line.
- **Step 6:** In this step, we need to apply an offset equal to half of the line width to place presented marker on their correct place of bridges' borders.

Here below is the CartoCSS code to style bridges in zoom level `[14 to 16]`:

```
[zoom >= 14] [zoom < 16]{
    ::background {
       line-offset: 8;
       line-color:#000000;
                          line-width: 0.5;
       marker-file: url('horline_half.svg');
      marker-placement: vertex-first;
      marker-transform: 'rotate(-45,0,0) translate(-2.5,0)';
       marker-offset: 8;
       marker-allow-overlap: true;
       ::foreground {
    line-offset: 4;
                  line-color:#ffffff;
    line-width: 4;
                  marker-file: url('horline_half.svg');
                  marker-placement: vertex-last;
                  marker-transform: 'rotate(45,0,0) translate(2.5,0)';
                  marker-offset: 8;
    marker-allow-overlap: true;
       }
    }
    ::foreground {
      line-offset: -8;
      line-color:#000000;
      line-width: 0.5;
      marker-file: url('horline_half.svg');
      marker-placement: vertex-last;
      marker-transform: 'rotate(-45,0,0) translate(2.5,0)';
      marker-offset: -8;
      marker-allow-overlap: true;
      ::background {
                  line-offset: -4;
                  line-color:#ffffff;
    line-width: 4;
                  marker-file: url('horline_half.svg');
    marker-placement: vertex-first;
                  marker-transform: 'rotate(45,0,0) translate(-2.5,0)';
                  marker-offset: -8;
    marker-allow-overlap: true;
      }
    }
  }
```

You can see the rendered bridges in these zoom levels in Figure 95.

Figure 95. Preview of Bridge layer rendered by Mapnik at different zoom levels

9.4. Z-order Challenge

Z-order is an ordering of overlapping graphics, such as polygons or lines in a vector dataset. *Z-ordering* is primarily related to sorting capabilities. For example, as mentioned in Chapter 9, Mapnik uses the painter's algorithm when drawing objects, meaning that items are drawn from bottom to top, with items defined earlier in the source layer list being drawn first. Consequently,

stylesheet developers should consider what features should appear over other features, such as roads over land uses, points of interest over buildings, and so on.

There are more complex situations, such as "spaghetti" road junctions, where multiple highways may cross multiple other highways. A map should accurately portray the vertical topology of overlapping roads. This topic needs to be further developed to address the following issues:

- The extent to which ordering considerations (and abilities) have to be included in the basic profile, given that the elaboration of an OSM (Open Street Map) style is highly dependent on such abilities but is not presumably the primary use-case;
- The ability to sort features in the same layer by their vertical topology requires the attributes that define the feature's "relative" or "absolute" vertical positions in the source data. More descriptions on the relative or absolute vertical position are mentioned in chapter 9.
    - For "relative" vertical positions, OSM uses relative position values for each feature, where "0" (zero) is the implied "layer" attribute for a ground level feature, "-1" (negative one) would be used for another feature that is underneath a ground level feature, and "1" (positive one) for a feature above a ground level feature. OSM publishes guidelines with instructions on using this attribute, and that it may range from "-5" to "5" for more complex junctions. Features are typically segmented such that only segments that overlap have a "layer" attribute defined, as a feature may be "above ground" in one location and then be "underground" farther along the feature.
    - For "absolute" position, each overlapping feature would have to define an absolute elevation relative to a datum. The features could then be sorted from lowest to the highest elevation before being read by the rendering engine for drawing on the map image. Acquiring absolute positions for overlapping features is more costly than defining relative positions using local knowledge.

When features are drawn onto the map image using the Mapnik library, there are some considerations that must be taken to properly portray the topology of the features. Mapnik uses the painter's algorithm when drawing objects, meaning items are drawn from bottom to top, with items defined earlier in the source layer list being drawn first. This means that stylesheet developers should consider what features should appear over other features, such as roads over land uses, points of interest over buildings, and so on.

Mapnik can handle more complex situations, such as "spaghetti" road junctions, where multiple highways may cross multiple other highways. A map should accurately portray the vertical topology of overlapping roads. The ordering of features is dependent on their layer order in the source layer list, and the same is true for features in the same layer. To draw overlapping road features in the correct order, they must be drawn from "bottom" to "top", and sorted such that the "lower" roads are loaded into Mapnik first.

NOTE

Once the source data has vertical topology available for its features, Mapnik can use Structured Query Language (SQL) to perform a sort clause on the layer to order the features by lowest to highest. When the source data is in a PostgreSQL database, Mapnik may use its "PostGIS" data source plugin to select and order the data directly. With Shapefile source data, the "shape" plugin for Mapnik (the plugin used by this deliverable for its map) does not support SQL nor sorting, but an experimental "ogr" plugin does support SQL and can read Shapefiles. This same "ogr" plugin may work with reading data from GeoPackage files, although it was not tested for this deliverable.

The order of how features are drawn is completely handled by Mapnik, and outside the scope of CartoCSS. This means there are no CartoCSS directives or functions that may change the order of features, aside from the ordering of source data layers in CartoCSS project files (".mml" files). This should not be confused with the ability of CartoCSS to customize the order of attached styles, such as outlines/inlines or attached symbols (such as the bridge symbol).

Chapter 10. GeoPackage with Portrayal Support

This section summarizes the findings, design approaches, and workflows used for OGC Testbed-14: D161 GeoPackage with Portrayal Support effort. This effort was conducted by Image Matters LLC to investigate how a GeoPackage could support feature portrayal on mobile devices.

10.1. Approach

An evaluation was conducted to determine the benefit(s) and timeline for the following approaches.

1) Proposing a new iteration of the GeoPackage Standard to include Portrayal support.

2) Proposing a new extension for the current GeoPackage Standard to add Portrayal support.

Ultimately, it was decided a "community" extension would provide the broadest benefit while maintaining the highest level of feasibility, and legacy compatibility.

10.2. Extension Requirements

The GeoPackage Portrayal team determined the following requirements were key.

- The proposed extension must provide the capability to specify common drawing parameters for simple features, such as: color, fill, fill style, line thickness, casing, *etc.*
- The proposed extension must provide the capability to transport and use pre-rendered symbols (PNG, GIF, SVG, *etc.*).
- Provide Portrayal rules to determine styling and visibility based on feature attribution.
- Provide Portrayal rules to determine styling and visibility based on map scale / zoom.
- Provide Portrayal support to GeoPackage **without** adding style information to each-and-every feature.
- Provide Portrayal support for feature classes contained within the same GeoPackage file as the Portrayal information.
- **Stretch Goal** Provide Portrayal support that can be assessed and applied to pre-existing feature classes within a GeoPackage (*e.g.*, A GeoPackage with portrayal information can be loaded with a separate GeoPackage containing feature data and still provide symbolization based on the feature class and feature metadata)

10.2.1. GeoPackage Portrayal Encoding

The GeoPackage Portrayal Encoding table structure follows the pattern of current GeoPackage extensions, such as Related-Tables Extension.

The proposed Portrayal Extension provides two additional tables and two rows in the "gpkg_extensions" table:

- "gpkgext_portrayal_symbols"
- "gpkgext_portrayal_rules"

The resulting entries in the "gpkg_extensions" table are presented below.

Table 22. GeoPackage "gpkg_extensions" table entries

id	description	table_name	extension_name
*TBD**	Portrayal Symbol Styles	gpkgext_portrayal_symbols	portrayal
TBD	Portrayal Style Rules	gpkgext_portrayal_rules	portrayal

*TBD: an arbitrary ID (starting at 1 and increasing)

GeoPackage Portrayal Tables

gpkgext_portrayal_symbols

The Portrayal table responsible for symbol styles is the "gpkgext_portrayal_symbols" table. The extension supports raster symbols such as .png and .gif file formats as a Binary Large OBject (BLOB) entry. Additionally, the portrayal extension also allows vector symbols such as .svg files. The GeoPackage Portrayal team has successfully rendered both types of symbols on a mobile client.

The *gpkgext_portrayal_symbols* table includes the *name* (Text) of the symbol. The name must be unique, and is the reference property in the portrayal rules. Additionally, the *symbol_data* (BLOB or Text) and *type* (Text) corresponding to the name of the symbol make up an entry in the table.

Table 23. GeoPackage “gpkgext_portrayal_symbols” table entries

name	symbol_data	type
sample_symbol	*BINARY DATA*	PNG
sample_symbol2	*"<svg>.....</svg>"*	SVG

gpkgext_portrayal_rules

The “gpkgext_portrayal_rules” table is responsible for feature and feature class specific symbolization rules. The approach taken for a style rules encoding was to use a standard that was human and machine readable, while additionally not requiring source code compilation. The participants ultimately chose a JavaScript Object Notation (JSON) format for the style rules encoding. The set of requirements to determine the style encoding were as follows:

- Each rule shall describe the feature class and geometry type for which the rule applies
- Each portrayal rules table entry is a one-to-one mapping to a feature class table

- Portrayal rules shall provide feature basic styling information (color, transparency, width, casing, fill-style, *etc.*)
- Portrayal rules shall provide the ability to adjust styling and visibility based on map scale
- Portrayal rules shall provide the ability to adjust styling and visibility based on feature attribution

Portrayal Rules Encoding

An elaboration of the Portrayal Rules encoding is presented by an example in JSON. The entire rule is encapsulated in a JSON object. Each object is being displayed and explained part-by-part for clarity purpose. A complete example follows the breakdown.

Example Mapping Rule declaration for "StructurePnt" feature class

```
{
    "symbol-mapping": {
      "table": "StructurePnt",
      "geometry-type": "POINT"
  },
  {...portrayal rules...}
}
```

This snippet encapsulates a rules object for all point features in the "StructurePnt" feature class. If the feature class were to contain more than a single geometry type, then the rules mapping could / would provide a mapping for each geometry type:

Example Mapping Rule declaration for “MixedFeatureClass”, a feature class with multiple geometries

```
[
  {"symbol-mapping": {
      "table": "MixedFeatureClass",
      "geometry-type": "POINT"
  }, "style-rule": ...point rules...},
  {"symbol-mapping": {
      "table": "MixedFeatureClass",
      "geometry-type": "POLYGON"
  }, "style-rule": ...polygon rules...}
]
```

Within the "style-rule" property, the participants found the selection criteria and applicable symbol styling declarations.

Example Style Rule for a point feature with specific attribution values

```
"style-rule": {
  "conditions": [{
      "criteria": [{
          "operand": "StndID_1",  */ Feature Attribute Name /*
          "operator": "=",        */ Operator /*
          "value": 0              */ Value /*
        },
        {                         */ All array entries are a stacked criteria e.g., f
eatures must meet all conditions /*
          "operand": "StndID_2",
          "operator": "=",
          "value": 2
        }
      ],
      "style": {
        "map-scales": [{  */ Map scale w/out specified min and max defaults to all map
scales /*
          "symbols": [{
            "image": {
              "name": "0310_Friend.svg",  */ Portrayal Symbol name to display /*
              "scale": 0.15,              */ Symbol scale /*
              "apply-gpkg-angle": false,  */ Whether or not to apply an offset angle
/*
              "angle-offset": 0,          */ Offset angle to apply /*
              "format": "image/svg+xml"   */ Symbol format /*
            }
          }]
        }]
      }
    }]
  }
```

Thee multiple conditional sections for specific combinations of feature attribution may be applied for styling purposes.

Example of multiple conditional statements

```
"style-rule": {
  "conditions": [{
        "criteria": [{
            "operand": "StndID_1",
            "operator": "=",
            "value": 0
          },
          {
            "operand": "StndID_2",
            "operator": "=",
            "value": 2
          }
        ],
        "style": {
          ...
        },
        {
          "criteria": [{
              "operand": "StndID_1",
              "operator": "=",
              "value": 0
            },
            {
              "operand": "StndID_4",
              "operator": ">=",
              "value": 3
            }
          ],
          "style": {
            ...
          }
        ]
      }
```

Example of style changes at various map scales

```
"map-scales": [{  */ Map scale with unit of measure being the scale denominator /*
  "min": 150000,  */ lowest map scale denominator for which to apply the style rule /*
  "max": 250000,  */ highest map scale denominator for which to apply the style rule
/*
  "symbols": [{
    "image": {
      "name": "building.svg",
      "scale": 0.07,
      "apply-gpkg-angle": false,
      "angle-offset": 0,
      "format": "image/svg+xml"
    }
  }]
}, {
  "min": 0,
  "max": 150000,
  "symbols": [{
    "image": {
      "name": "building.svg",
      "scale": 0.14,
      "apply-gpkg-angle": false,
      "angle-offset": 0,
      "format": "image/svg+xml"
    }
  }]
}]
```

Cursory note: The participants specifically chose map scale denominator here in stead of map zoom level. This was performed deliberately as various vendors and map source data providers use varying zoom level scales (0 - 19 vs. 0 -23, *etc.*). By specifying the map denominator, the participants removed any ambiguity on author visualization preference. Further discussion points are added in the Findings section of this chapter.

An example of another simple Geometry, Line is presented below:

Example of style rules for a feature class with a Line geometry type

```
"style": {
  "map-scales": [{
    "min": 0,
    "max": 200000,
    "symbols": [{
      "stroke": {
        "color": "0x00a0c6",
        "opacity": 1.0,
        "background-color": "0xFFFFFF",
        "background-opacity": 1.0,
        "width": 10,
        "vertical-offset": 0,
        "horizontal-offset": 0,
        "dashes": [],
        "cap": "",
        "casing": {
          "color": "0x0",
          "opacity": 1.0,
          "width": 0
        }
      }
    }]
  }]
}
```

Example of style rules for a feature class with Polygon geometry type

```
"style": {
  "map-scales": [{
    "min": 0,
    "max": 200000,
    "symbols": [{
        "fill": {
          "color": "0xe8c3b2",
          "opacity": 1.0,
          "image": {
            "name": "",
            "scale": 0,
            "apply-gpkg-angle": false,
            "angle-offset": 0,
            "format": ""
          }
        }
      },
      {
        "stroke": {
          "color": "0x000",
          "opacity": 1.0,
          "background-color": "0xFFFFFF",
          "background-opacity": 1.0,
          "width": 5,
          "vertical-offset": 0,
          "horizontal-offset": 0,
          "dashes": [],
          "cap": "",
          "casing": {
            "color": "0x000",
            "opacity": 1.0,
            "width": 0
          }
        }
      }
    ]
  }]
}
```

10.3. Interoperability Experiment

10.3.1. Dataset

A Daraa, Syria Shapefile was provided by Testbed sponsors for use by all Portrayal participants. The approach taken was to parse the styling rules provided by the sponsor and produce a reference encoding. The GeoPackage Portrayal team utilized the reference encoding and also an analysis of the original data provided where the source encoding was ambiguous or lacked clarity (such as unit

of measure).

10.3.2. Interoperability Among Participants

The participants' goal, as stated, was to provide a GeoPackage extension with Portrayal capability. However, the participants also took part in a few interoperability "stretch" goals. Namely testing WMS and WMTS with Portrayal support. The approach taken to support the interoperability test was to simply register the WMS and WMTS servers from each participant and provision a GeoPackage with a raster tile pyramid from each. Then to visually validate the contents and symbolization.

The participants were successful in registering and provisioning raster tiles from the GeoSolutions WMS and WMTS.

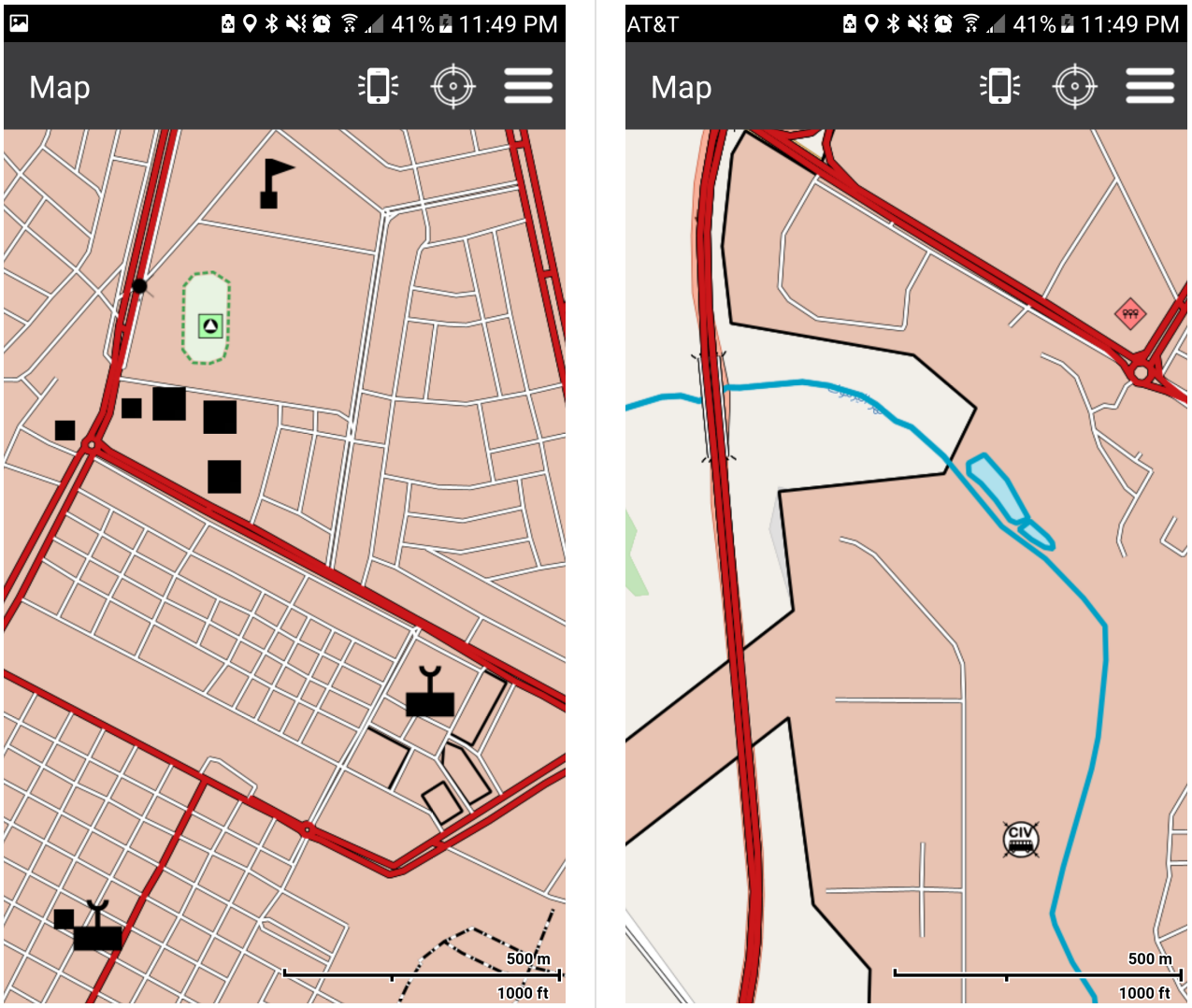
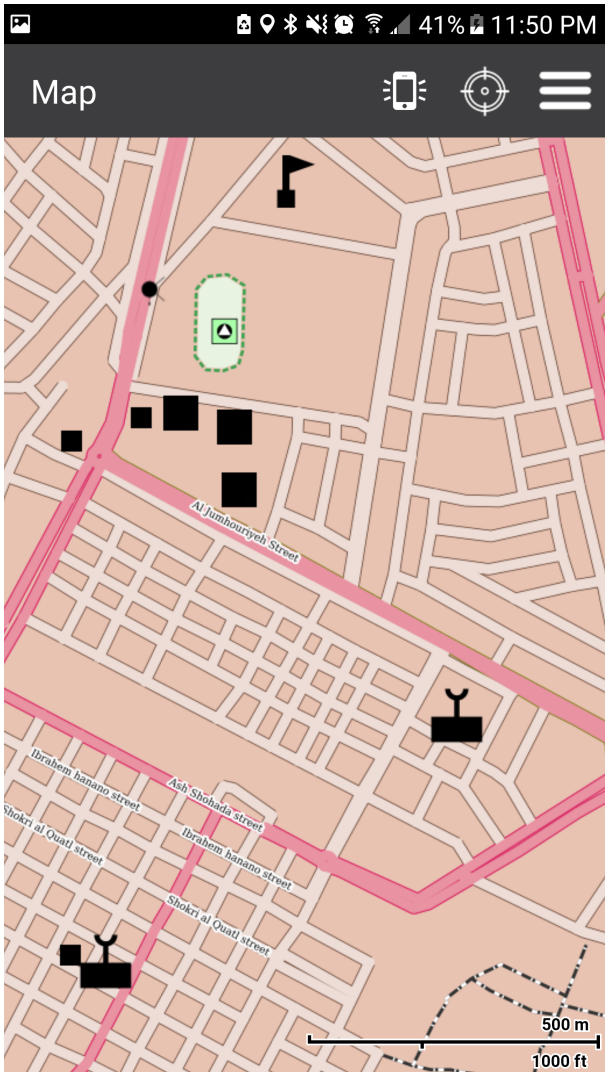


Figure 96. GeoSolutions WMS provisioned into a GeoPackage (rendered via GeoServer)



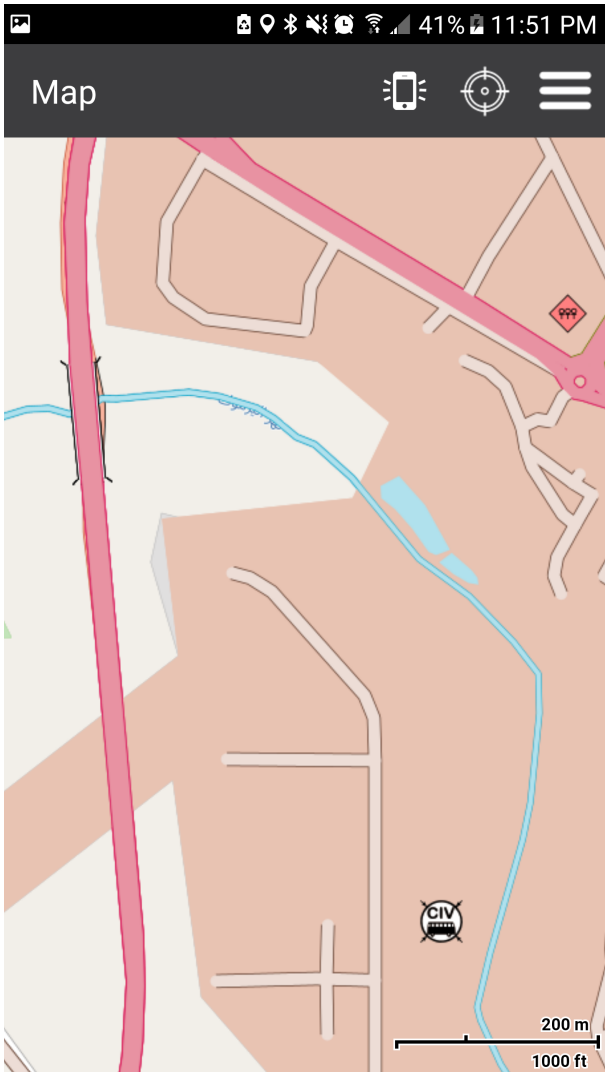


Figure 97. GeoSolutions WMS provisioned into a GeoPackage (rendered via Mapbox)

Image Matters were unable to provision a GeoPackage from the University of Calgary Server.

10.3.3. Computing Environments

The Portrayal processing workflow crossed a number of computing environments. Initial validation of the Shapefile was performed on a desktop GIS application.

The validated dataset was hosted as a WFS using a cloud-based GeoServer instance.

The WFS was harvested and provisioned into a GeoPackage with Portrayal extension, via a cloud-based GeoPackage provisioning tool.

Resulting GeoPackages were delivered to mobile handsets for rendering.

- Cloud-based GeoPackage Provisioning
- Connected/disconnected/intermittent/limited(communications) Mobile Client (Android / IOS)

10.3.4. Attribute Parsing

Feature attributes were parsed into *KeyValuePairs* where:

column_name → Key

column_value → Value

OSM feature attribution was encapsulated as a JSON value within an attribute *ZI0006* in many of the feature classes contained in the Shapefile. The participants parsed the internal JSON and added the results to the *KeyValuePairs* that were generated from the feature class columns.

The resulting feature object carried enough metadata to apply the Portrayal rules for rendering.

10.3.5. Simple Feature Rules Application

Point

The un-styled "StructurePnt" feature class appears as light green point symbols. While the styled symbols apply one of three .svg symbols based on feature attributes. Additionally, the size of the .svg file is scaled based on the feature attributes and zoom level. The "StructurePnt" feature class also included a composite symbol for the attribute `osm_religion = Mosque`, as shown by the rectangular symbol with a vertical line connecting a half circle.

Figure 98. StructurePnt multipoint features **without** *Portrayal rules*

Figure 99. StructurePnt multipoint features with Portrayal rules applied

Line

We can observe the line thickness, color, and dashes were applied to appropriate feature classes. The dashed lines were draw to the color and dash frequency as specified by the source encoding, however due to the lack of specificity in the unit of measure these lines do not appear as intended. Additionally, the participants found issues during zoom operations as the dashed symbols would constantly recalculate and redraw between map scales.

Figure 100. TransportationGroundCrv features **without** *Portrayal rules*

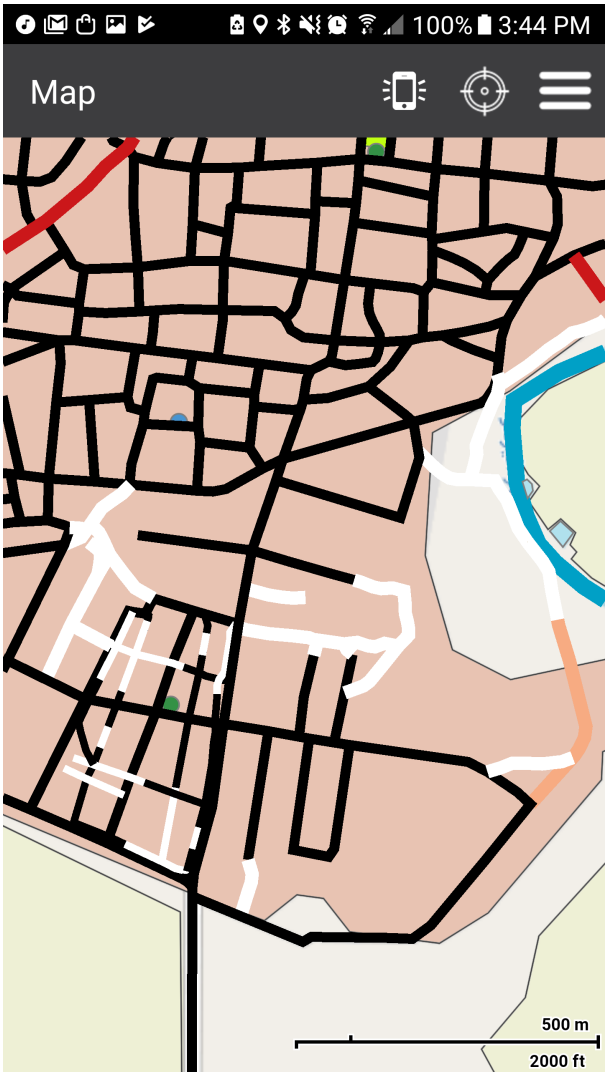


Figure 101. TransportationGroundCrv multilinestring features with Portrayal rules applied (Styled)

Polygon

Polygon features were capable of having color and transparency set and rendered by the mobile client. However, the .svg fill style is not currently supported by our rendering engine. The encoding however, does fully support the symbol ruleset required to specify common fill types (*e.g.*, solid, hatched, *etc.*) and symbol fill (*e.g.*, .svg file).

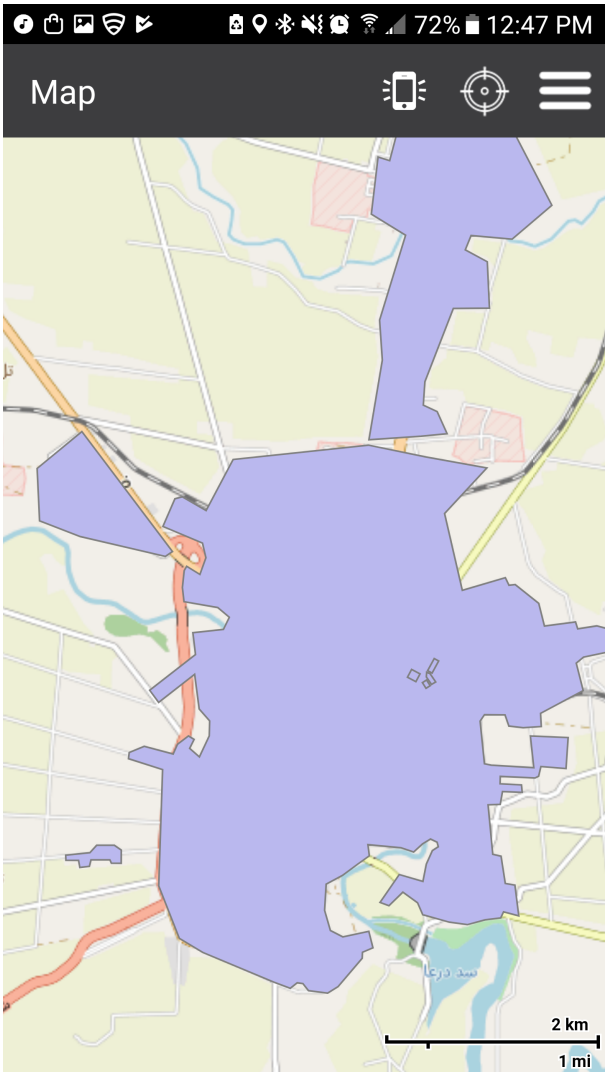


Figure 102. SettlementSrf multipolygon features with Portrayal rules applied (Unstyled)

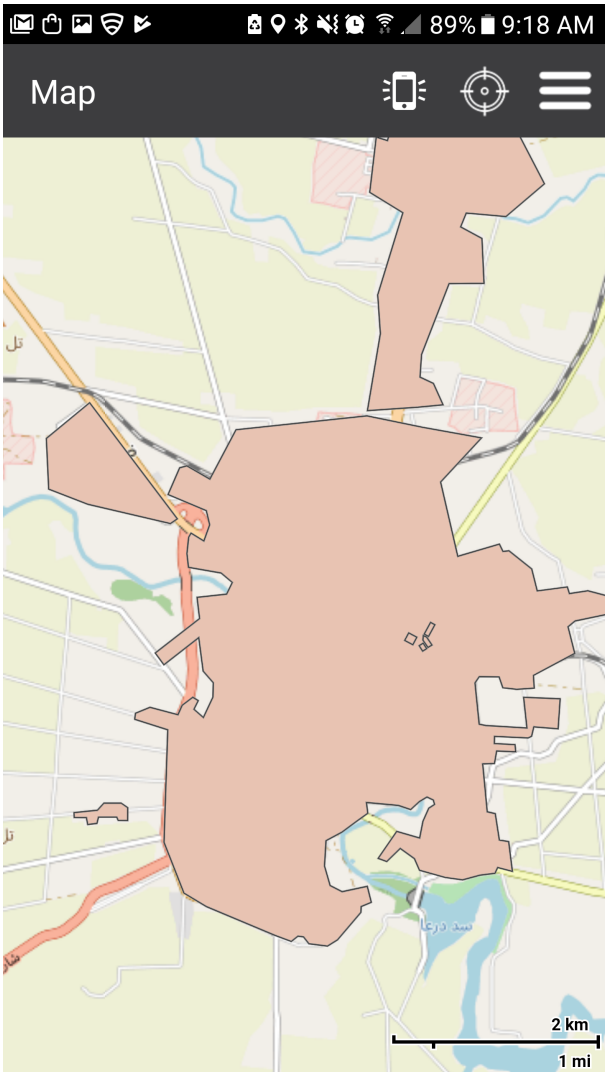


Figure 103. SettlementSrf multipolygon features with Portrayal rules applied (Styled)

10.3.6. Feature Symbols

Now, we further breakdown the symbolization rules.

Styling and Sizing

As mentioned above, we saw an example of features being styled according to attributes with varying symbolizations within the same feature class.

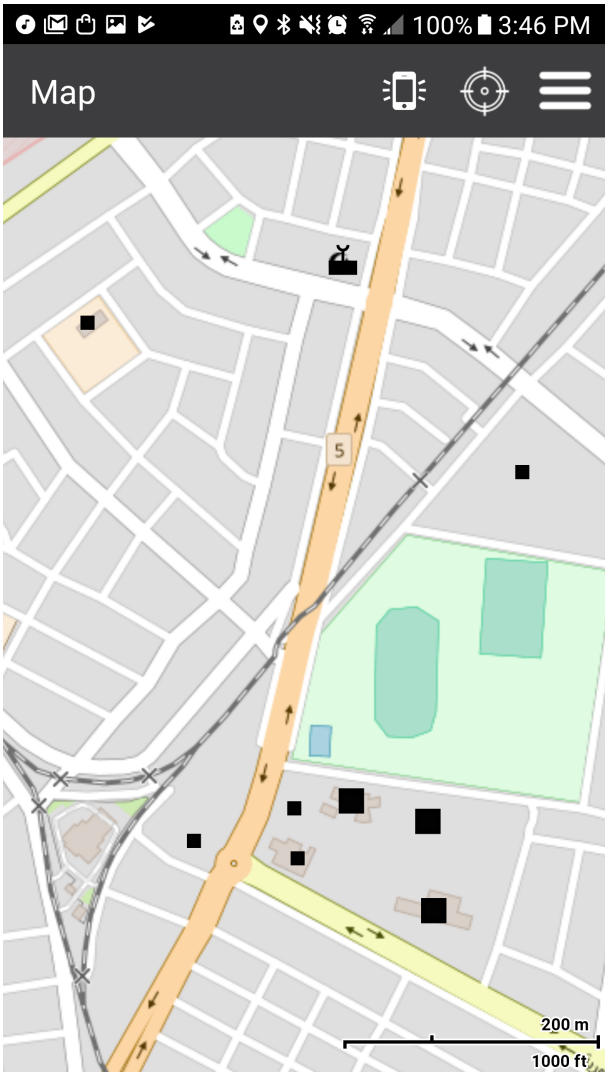


Figure 104. StructurePnt multipoint features with SVG style and size

Zoom Levels

Alternate symbolization, scale, drawing properties, *etc.* based on map zoom level.

TIP `Map Scale greater than or equal to 1:250000, Zoom Level >= 11 ⇒ Rule: No Style`

Figure 105. StructurePnt Map Scale 1:250000+

TIP Map Scale less than 1:250000, Zoom Level < 11 ⇒ Rule: Display w/ appropriate symbol scale

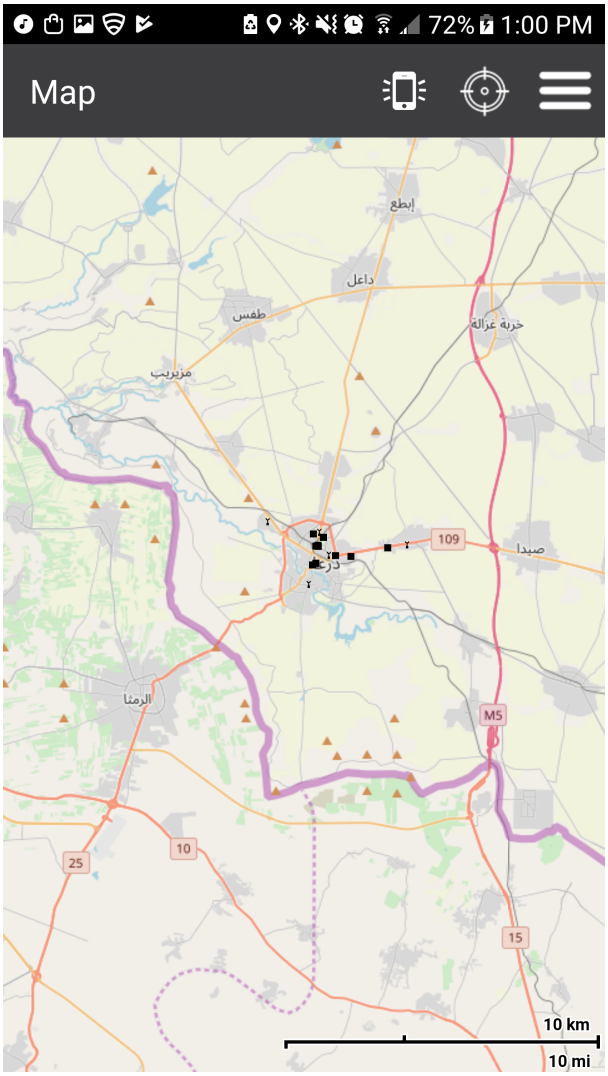


Figure 106. StructurePnt Map Scale ~1:170000

TIP `Map Scale less than or equal to 1:70000, Zoom Level ⇐ 13 ⇒ Rule: Resize symbol scale`

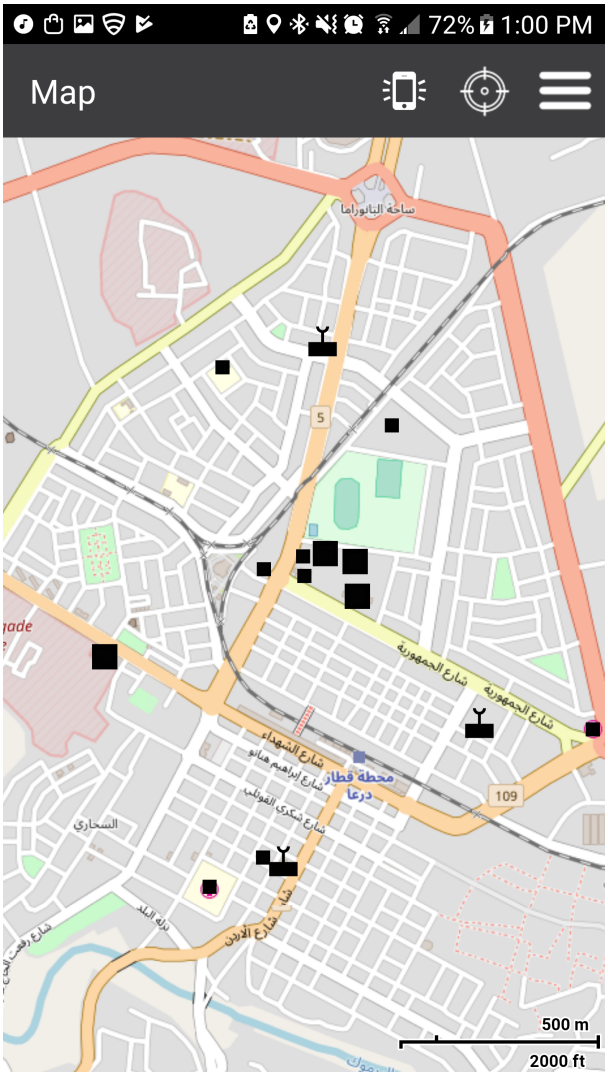


Figure 107. StructurePnt Map Scale ~1:7000

Composite Symbols

We see a composite symbol "pylon_wilson.png" with an applied label (also provided by the portrayal rules). Additionally, we see the *Z-order* of the line feature and the placement of the pylons above.

Figure 108. UtilityInfrastructurePnt and Curve features

MIL-STD-2525

MILSPEC 2525 symbols are shown with composite symbols. One of which is also provided a rotation.

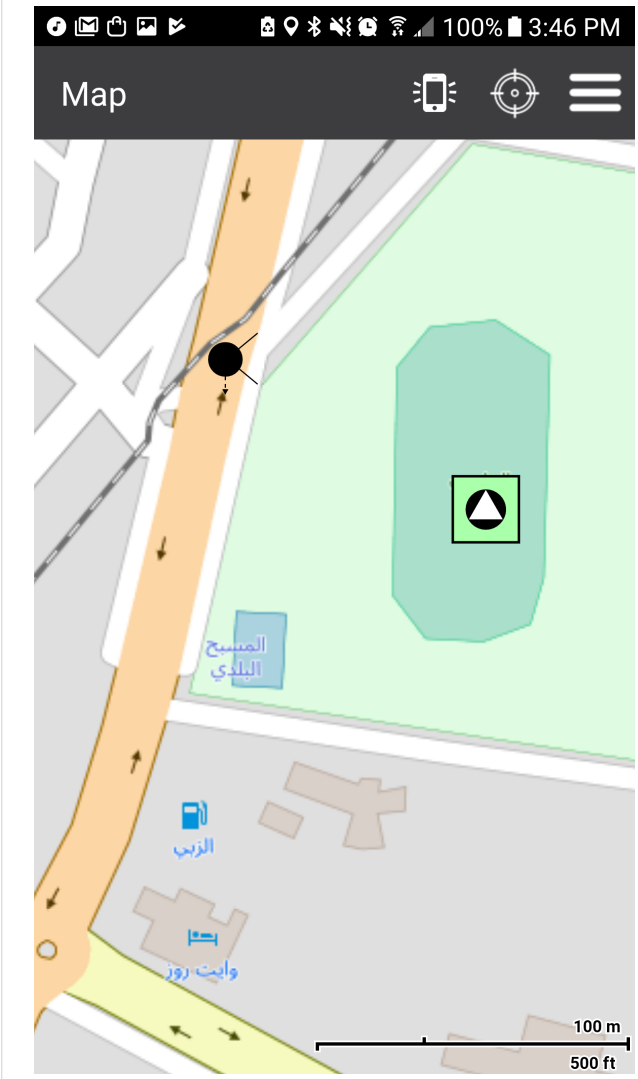


Figure 109. MIL-STD-2525 features

10.4. Findings

Implementation of the proposed GeoPackage Portrayal extension had mixed results in the mobile rendering engine. However, the Portrayal rules were concise, interpretable, and simple to apply. The participants were able to produce a set of rules for feature classes and feature attributes from a single GeoPackage while also loading alternate styling by utilizing two GeoPackage files (one with feature data, and one with portrayal rules).

10.4.1. Improvements

Though the participants were successful in providing an extension to satisfy the testbed TIE demo, there are additional improvements the GeoPackage Portrayal team feels must be made prior to proposing a stable version of the GeoPackage Portrayal Extension to the OGC.

- Allow for map scale OR zoom level
- Reduce redundancy in the extension

```
For example, the symbol type is repeated in the rules and the
gpkgext_portrayal_symbols table.
```

- Conditional criteria is an "AND" operator implicitly.

```
Adjustments should be made to include other logical operators (_e.g._, OR, NOT).
```

- Specification of colors should follow the W3C standard for CSS specified color codes.
- Line dashes need a unit of measure to commiserate with the viewport and screen pixel density similar to that of `em` and `rem` in CSS which are are relative units.

10.4.2. Future Considerations

A relatively small amount of effort would be required to 1) adjust the shortcomings of the current extension and 2) allow the Portrayal encoding to support Vector Tiles features. The participants believe the benefit of Vector Tiles along with portable GeoPackage Portrayal rules would allow for great flexibility and reduced data transfer for a not long connected client, but in particular DIL clients.

Chapter 11. Findings

Section 5 Portrayal conceptual model

- Testbed-14 extended the portrayal ontology developed in previous Testbeds, and described the Portrayal Conceptual Model as a common reference to drive implementations through encoding formats.
- Currently, the SLD/SE Standards Working Group (SWG) supports the following principle that can also be considered as the primary hypothesis driving this testbed
  - given a styling document `X` encoded in conformance with the conceptual model, and the related rendering engine producing map `X`;
  - given a styling document `Y` encoded in conformance with the conceptual model, and the related rendering engine producing map `Y`;
  - then, map `X` is “very similar” to map `Y` with narrow preservation of the cartographic message.
- The development of the portrayal conceptual model includes analysis, thoughts and design recommendations related to:
  - the concept of the Symbolizer, the related extension points and the relation with the Symbol concept introduced by previous Testbeds;
  - the rule interpretation with cascading and the importance to frame the rendering ordering;
  - the ability to control multiple and straightforward pass composition;
  - the attribute-dependent graphical parametrization (in other words "expressions everywhere") concerning the binding mechanism proposed by previous Testbeds;
  - the symbolization capabilities of a basic/simple profile useful for 2D vectors rendering;
  - the default values and the definition of a concept of color.

Section 7 WMTS With Portrayal Support

- Implement WMTS instance based on the Portrayal Conceptual Model
- In this section, OGC **SLD** and **CSS** encodings were used for styling vector data and **GeoServer** used for rendering the data.

Section 8 Lessons from CARTO and Mapbox underlying encoding formats

- Analyzes two other encoding formats for styling vector data and rendering them (esp. with the Daraa's use case) with two platforms based on free and open source solutions: **CARTO online builder** (CartoCSS code generator) and **Mapbox studio** (data oriented with JSON filtering).
- Especially **CartoCSS** is analyzed (somehow reverse-engineered) to extract relevant cartographic abilities to naturally put together in extensions of the conceptual model based on the extension points resulting from Testbed-14.
- Also, some comparisons are made **GeoCSS** used by GeoServer.

Section 9 WMS With Portrayal Support

- implement WMS instance based on the Portrayal Conceptual Model.
- In this section, **CartoCSS** encoding has been used for styling vector data and **Mapnik** used for rendering.

Section 10 GeoPackage With Portrayal Support

- The Portrayal rules were concise, interpretable, and simple to apply.
  - a set of rules was produced for feature classes and feature attributes from a single GeoPackage.
  - also, alternate styling was loaded by utilizing two GeoPackage files (one with feature data, and one with portrayal rules).
- A relatively small amount of effort would be required to 1) adjust the shortcomings of the current extension and 2) allow the Portrayal encoding to support Vector Tiles features. We believe the benefit of Vector Tiles along with portable GeoPackage Portrayal rules would allow for great flexibility and reduced data transfer for a not long connected client.
- Though we were successful in providing an extension to satisfy the testbed TIE demo, there are additional improvements the GeoPackage Portrayal team feels must be made before proposing a stable version of the GeoPackage Portrayal Extension.
  - Allow for map scale OR zoom level.
  - Reduce redundancy in the extension.

    ```
    For example, the symbol type is repeated in the rules and the
    _gpkgext_portrayal_symbols_ table.
    ```

  - Conditional criteria is an "AND" operator implicitly.

    ```
    Adjustments should be made to include other logical operators (_e.g._, OR,
    NOT).
    ```

  - Specification of colors should follow the W3C standard for CSS specified color codes.
  - Line dashes need a unit of measure to commiserate with the viewport and screen pixel density similar to that of `em` and `rem` in CSS which are relative units.

Appendix A: Conceptual Model Expressions

A.1. Expressions

A.1.1. Overview

Expressions are used both for selectors as well as for styles values.

A **selector** is an expression which shall resolve to either TRUE or FALSE, and determines whether a rule is to be applied or not

CMSS Encoding examples

```
#Roads // Equivalent to [lyr.id='Roads']

[scale>10M][FEATCODE=123]
```

When resolving to a Boolean value an expression which is either:

- null (unset)
- a numeric 0 (whether integer or real)
- or an enumeration value assigned the integer value 0 (e.g. false)

  evaluates to false;

all other expressions evaluate to true.

A.1.2. Primary Expressions

Kind of expression	Description	CMSS Encoding examples
identifier	An identifier of a certain kind resolving to a value ([l|L][l|L|d]*) (*see Identifiers*)	FEATCODE
text	A fixed UTF-8 character string	'Parking'
integer	An integral number (up to 64 bit)	10
real	A real number (IEEE floating-point, up to double precision)	3.14159

Kind of expression	Description	CMSS Encoding examples
object	If some elements are omitted, they inherit default values (0 or null, but which can be interpreted in a special way based on the object class) Objects classes can support inheritance and the definition of an object can include the specification of a derived class for this object	{ hour = 16, minutes = 30 } Text { Name } Circle { radius = 5, color = red }
list	A list of expressions	[1, 2, 3]
operation	A number of operands combined by an operator (*see Operators*)	viz.sd > 10M
variable	If variables are supported, the encoding should define default value for them which should produce reasonable styling.	@colorScheme

A.1.3. Identifiers

Kind of identifier	Description	CMSS Encoding examples
null	An identifier representing an unset value	[someValue!=null]
enumValue	A valid value for expected for an enumeration type	[true]
layer	An object describing the layer being styled	lyr
identifier	A unique identifier to reference the layer	[lyr.id = Roads] (equivalent: #Roads)
featureClass	The type of geospatial data	[lyr.fc = vector]
vectorType	The type of vector data	[lyr.vt = points]
geometry	The geometry of the entire layer (a VectorFeatureCollection)	lyr.geom
visualization	An object describing the current state of the visualization	viz
scaleDenominator	The denominator for a fraction representing the scale / resolution at which the data is being looked at	[viz.sd > 10M]

Kind of identifier	Description	CMSS Encoding examples
time	The current time of visualization (This contains the time elements of both date as well as timeOfDay)	[viz.time > { 2018, june, 30, 16, 30 }]
date	The current date of visualization (year, month, day)	[viz.date > { 2018, june, 30 }]
timeOfDay	The current time of visualization (hours, minutes, seconds)	[viz.timeOfDay > { 16, 29, 59 }]
record	An object describing the current record (vector data)	rec
id	The unique ID identifying the record	[rec.id = 1234]
geometry	The actual geometry of the record (a single VectorFeature)	centroid(rec.geom)
(data attributes)	Data attributes associated with the record	FEATCODE
records	A list of all records in the layer	[featcode = 'bay' & iterate(or, records, it.featcode = 'ocean' & intersect(it, rec.geom))] True if this is a bay intersecting with an ocean in another record

A.1.4. Operators

Operator	Operands Count	Description	CMSS Encoding	SLD/SE
Logical				
and	2		&	And
or	2		|	Or
not	1		!	Not
Comparison				
equal	2		=	PropertyIsEqualTo

Operator	Operands Count	Description	CMSS Encoding	SLD/SE
notEqual	2		!=	PropertyIsNotEqualTo
greater	2		>	PropertyIsGreaterThan
lesser	2		<	PropertyIsSmallerThan
greaterEqual	2		>=	PropertyIsGreaterThanOrEqualTo
lesserEqual	2		<=	PropertyIsLessThanOrEqualTo
Text-specific				
stringContains	2		~	
stringStartsWith	2		^	
stringEndsWith	2		$	
stringNotContains	2		!~	
stringNotStartsWith	2		!^	
stringNotEndsWith	2		!$	
Arithmetic				
add	2	For numeric values: addition For text values: concatenation	+	Add
sub	2		-	Sub
mul	2		*	Mul
div	2		/	Div
intDiv	2		idiv	
mod	2		%	Mod
Others				
parentheses	1		( )	
conditional	3	condition-then-else expression	viz.sd>10M?1:3	
in	2	True if the left operand is within the right operand list	TYPE in [1,2,3]	
functionCall	fn,param*	(*See Functions*)	length(r.geom)	

A.1.5. Functions

Fuction	Description	CMSS Encoding	SLD/SE
Text Manipulation			
format			
replace			
Geometry Operation			
area			
length			
centroid			
Spatial Operations			
intersects			
contains			
Iteration			
iterate	Iterate over a list to evaluate an expression *it* represents the current iterator		

Appendix B: GNOSIS CMSS Encoding of the Conceptual Styling Model

This appendix presents an example of the conceptual styling model, with portrayal rules for the data set used during the Testbed and Vector Tiles Pilot, encoded as *GNOSIS Cartographic Map Style Sheets (CMSS)*. Objectives of the CMSS encoding (which was improved upon during Testbed-13 and Testbed-14) include maximizing expressiveness, while attempting to correspond as closely as possible to the conceptual model developed in this testbed (although it currently reflects a version of the conceptual model before some modifications were introduced).

Note that the proper styling of bridges is currently not reflected in the portrayal rules encoded here.

An example of cascading portrayal rules from the conceptual styling model encoded as GNOSIS CMSS

```
// Everything is off by default
{ visibility = false; }

#agriculturesrf
{
   zOrder = 0;
   [F_CODE = 'EA010'][OSM_T_LAN = 'farmland']
   {
      fill =
      {
         backgroundColor = 0xf7eded,
         elements = [ Image { 'symbols/Green_AP77_Fill.svg', size = Meters { 100 } } ]
      };
      stroke = { black, width = Meters { 5 } };

      [viz.sd < 200k] { visibility = true; }
   }
}

// Woods
#vegetationsrf
{
   zOrder = 1;
   [F_CODE = 'EC015'][OSM_T_LAN = 'forest']
   {
      fill = { 0xc2e4b9 };
      [viz.sd < 1M] { visibility = true; }
   }
}

#hydrographysrf
{
   // Lake
   [F_CODE = 'BH082']
```

```
   {
      fill = { 0xb0e1ed };
      zOrder = 2;

      [viz.sd < 200k] { visibility = true; }
   }
   // Dam (No Transportation)
   [F_CODE = 'BI020']
   {
      stroke = { 0x333333, Meters { 20 } };
      zOrder = 3;

      [viz.sd < 500k] { visibility = true; }
   }
}

#culturesrf
{
   zOrder = 4;
   [F_CODE = 'AL030'][OSM_T_LAN = 'cemetery']
   {
      stroke = { black, width = Meters { 5 }, dashes = [ 5, 2.5 ] };
      [viz.sd < 200k] { visibility = true; }
   }
}

#emergency
{
   zOrder = 12;
   [viz.sd < 35k]
   {
      visibility = true;
      [type = 'reunification']
         { marker = { [ Image { "symbols/emergency/civilian-staging-transit.svg" },
size = 32 ] } }
      [type = 'info_point']
         { marker = { [ Image { "symbols/emergency/public-information.svg"       },
size = 32 ] } }
   }
}

#hydrographycrv
{
   zOrder = 10;
   [F_CODE = 'BH140']
   {
      stroke = { 0x00a0c6, width = Meters { 10 } };

      [OSM_T_LAN = 'drain'] { stroke.dashes = [ 10, 2.5 ]; }
      [viz.sd < 200k] { visibility = true; }
   }
```

```
}
#informationpnt
{
   zOrder = 11;
   [OSM_T_LAN ~ 'Syria' or OSM_T_PLA = 'locality'][viz.sd < 500k]
   {
      visibility = true;
      label = { [
         Text {
            OSM_T_ENG, color = black, font = { "OpenSans", size = 10 },
            alignment = middleCenter, outline = { white, size = 1 }
         }
      ] }
   }
}

#SettlementPnt
{
   [viz.sd < 20k]
   {
      visibility = true;
      label = { [
         Text
         {
            OSM_T_ENG, color = black, font = { "Open Sans", size = 12, bold = true },
            alignment = middleCenter, outline = { white, size = 1.5, fade = 0.25 }
         }
      ] }
   }
}

// Sports Ground
#recreationsrf
{
   zOrder = 6;
   [F_CODE = 'AK040']
   {
      stroke = { 0x3ea250, width = Meters { 5 }, dashes = [ 5, 2.5 ], offset = 2 };
      fill = { 0xe8f3e2, elements = [ Image { 'symbols/sports_ground_wilson.svg', size
= Meters {6} } ] };
      [viz.sd < 200k] { visibility = true; }
   }
}

#settlementsrf
{
   zOrder = 4;
   [F_CODE = 'AL020']
   {
      fill = { 0xe8c3b2 };
      stroke = { 0xe8c3b2, width = Meters { 5 } };
```

```
      [viz.sd < 200k] { visibility = true; }
   }
}

#transportationgroundcrv
{
   label = { [
      Text {
            OSM_T_ENG, color = black, font = { "Open Sans", size = 12, bold = true },
            alignment = middleCenter, outline = { white, size = 2.5, fade = 0.25 } }
   ] };

   [F_CODE = 'AP030']
   {
      // Primary & Secondary roads
      [RIN_ROI in (3,4)]
      {
         stroke = { 0xcb171a, width = 0.9, casing = { black, width = 0.12 } };

         [viz.sd < 25M] { visibility = true; }
         [viz.sd < 2.5M] { stroke.width = 1.14 , stroke.casing.width = 0.18; }
         [viz.sd < 500k] { stroke.width = 2.25 , stroke.casing.width = 0.375; }
         [viz.sd < 100k] { stroke.width = Meters { 8 }, stroke.casing.width = 0.8225;
}
         [viz.sd <   5k] { stroke.width = Meters { 8 }, stroke.casing.width = 2; }
      }
      // Streets
      [RIN_ROI = 5]
      {
         stroke = { black, width = 0.65, casing = { white, width = 0.09 } };

         [viz.sd < 25M] { visibility = true; }
         [viz.sd < 2.5M] { stroke.width = 0.885, stroke.casing.width = 0.135; }
         [viz.sd < 500k] { stroke.width = 1.688, stroke.casing.width = 0.281; }
         [viz.sd < 100k] { stroke.width = Meters { 6 }, stroke.casing.width = 0.617; }
         [viz.sd <   5k] { stroke.width = Meters { 6 }, stroke.casing.width = 1.5  ; }
      }
   }

   // Bridge (old styles, to be improved with geometry modification or alternate
approach)
   [F_CODE = 'AQ040'][TRS = '13']
   {
      stroke =                { black, width = 12, cap = square };

      [viz.sd < 100k] visibility = true;
      [viz.pass=1] { stroke = { white, width = 10, cap = round }; }
      [viz.sd < 5k]
      {
         stroke =                { black, width = 20, cap = square };
         [viz.pass=1] { stroke = { white, width = 14, cap = round  }; }
```

```
      }
   }

   // Cart Track
   [F_CODE = 'AP010']
   {
      stroke =
      {
         color = white, backgroundColor = 0xf7ab82, width = 0.9, casing = { black,
width = 0.12 },
         dashes = [ 1.870, 4.124, 0.625, 1.375 ]
      };

      [viz.sd < 25M] { visibility = true; }
      [viz.sd < 2.5M] {
         stroke.width = 1.14,         stroke.casing.width = 0.18,
         stroke.dashes = [ 2.415, 5.323, 0.807, 1.774 ]; }
      [viz.sd < 500k] {
         stroke.width = 2.25,         stroke.casing.width = 0.375,
         stroke.dashes = [ 4.797, 10.577, 1.603, 3.526 ]; }
      [viz.sd < 100k] {
         stroke.width = Meters { 8 }, stroke.casing.width = 0.8225,
         stroke.dashes = [ 10, 22, 3.333, 7.333 ]; }
      [viz.sd <   5k] {
         stroke.width = Meters { 8 }, stroke.casing.width = 2,
         stroke.dashes = [ 18.100, 39.820, 6.033, 13.273 ]; }
   }

   // Trail
   [F_CODE = 'AP050' ]
   {
      stroke = { color = white, backgroundColor = black, width = 2.363, dashes = [ 5,
9, 1.5, 2.5 ] };

      [viz.sd < 100k] { visibility = true; }
      [viz.sd < 5k] { stroke.width = Meters { 3 }, stroke.dashes = { 10, 18, 3, 5 }; }
   }
}

#utilityinfrastructurepnt
{
   [F_CODE = 'AT042'][OSM_T_POW = 'tower']
   {
      marker = { [ Image { 'symbols/modified/pylon_wilson.svg', size = Meters { 250 }
} ] };

      [viz.sd < 200k] { visibility = true; }
      [HGT >= 46] { marker = { [ Image { 'symbols/PT_Blue072_VO.svg', size = Meters {
600 } } ] }; }
      [viz.sd < 35k]
      {
```

```
         label = { [
            Text
            {
               'Pylon', color = black, alignment = upperCenter, offset = { y = 3 },
               font = { 'Open Sans', size = 12 }, outline = { white, size = 1 }
            }
         ] };
      }
   }
}

#utilityinfrastructurecrv
{
   [F_CODE = 'AT005'][OSM_T_POW = 'line']
   {
      stroke = { 0x473895, width = Meters { 20 } };

      [viz.sd < 200k] visibility = true;
   }
}

#structurepnt
{
   [F_CODE = 'AL013']
   {
      marker = { [ Image { 'symbols/building.svg', size = Meters { viz.sd < 100k ? 40
: 3 } } ] };

      [viz.sd < 200k] { visibility = true; }

      [OSM_T_AME = 'place_of_worship'][OSM_T_REL = 'muslim']
      {
         marker =                  { [ Image { 'symbols/PT_Black_Mosque.svg', size =
Meters { 15} } ] };
         [viz.sd < 100k] { marker = { [ Image { 'symbols/AR_Black_Mosque.svg', size =
Meters {130} } ] }; }
      }
      [OSM_T_AME = 'college']
      {
         marker =                  { [ Image { 'symbols/PT_Black_School.svg', size =
Meters { 15} } ] };
         [viz.sd < 100k] { marker.elements[0].size = Meters { 130 }; }
      }
      [OSM_T_AME = 'public_building']
      {
         marker.elements[0].size = Meters { 70 };
      }
   }
}

#Mil2525
```

```
{
   marker = { };

   [Stnd_1=0][StndID_2 in (2,3,4,5)]
   {
      visibility = true;
      marker.elements += [ Image {
        (StndID_2 == 2 ? "symbols/0310_Friend.svg" :
         StndID_2 == 3 ? "symbols/0410_Neutral.svg" :
         StndID_2 == 4 ? "symbols/0510_Suspect_Joker.svg" :
                         "symbols/0610_Hostile_Faker.svg")
      } ];
   }
   [Entity=11][EntityT=4][EntityST=0]
   {
      visibility = true;
      marker.elements += [ Image { "symbols/11110400_Organization or Group.svg" } ];
   }

   [Entity=19][EntityT=0][EntityST=0]
   {
      visibility = true;
      marker.elements += [ Image { "symbols/10190000_Emergency_Operation.svg" } ];
   }

   [Entity=28][EntityT=2][EntityST=1]
   {
      visibility = true;
      marker.elements += [ Image { "25280201_Antipersonnel Mine with Directional
Effects.svg",
         orientation = viz.orientation + Angle - Degrees { 90 }
      } ];
   }
}
```

Appendix C: GeoCSS Style Encoding

This appendix shows the GeoCSS Style Encoding for the styles described in the chapter 7.

C.1. Description of the GeoCSS Style Encoding

C.1.1. agriculturesrf.css

```
[F_CODE = 'EA010' and OSM_T_LAN = 'farmland'] [@sd < 200k] {
  fill: #F7EDED, url('file://symbols/Green_AP77_Fill.svg');
  fill-mime: 'image/svg';
  fill-size: 100m;
}
```

C.1.2. vegetationsrfv.css

```
/* @title Woods. */
[F_CODE = 'EC015' and OSM_T_LAN = 'forest'] [@sd < 1M] {
  fill: #C2E4B9;
}
```

C.1.3. hydrographysrf.css

```
/* @title Lake */
[F_CODE = 'BH082'] [@sd < 200k] {
  fill: #B0E1ED;
  stroke: #00A0C6;
  stroke-width: 10m;
  z-index: 0;
}

/* @title Dam (No Transportation)*/
[F_CODE = 'BI020'] [@sd < 500k] {
  stroke: #000000;
  stroke-width: 20m;
  z-index: 1;
}

/* @title Dam*/
[F_CODE = 'BI020'] [@sd < 500k] {
  fill-geometry: [centroid(the_geom)];
  mark: url('file://symbols/PT_Black_Dam.svg');
  mark-mime: 'image/svg';
  mark-size: 10m;
}
```

C.1.4. culturesrf.css

```
/* @title Cemetery, Unknown */
[F_CODE = 'AL030' and OSM_T_LAN = 'cemetery'] [@sd < 200k] {
  stroke: #000000;
  stroke-width: 5m;
  stroke-dasharray: 5 2.5;
}
```

C.1.5. emergency.css

```
[type = 'reunification'] [@sd < 35000] {
  mark: url('file://symbols/emergency/civilian-staging-transit.svg');
  mark-mime: 'image/svg';
  mark-size: 32;
}

[type = 'info_point'] [@sd < 35000] {
  mark: url('file://symbols/emergency/public-information.svg');
  mark-mime: 'image/svg';
  mark-size: 32;
}
```

C.1.6. hydrographycrv.css

```
/* @title River, Permanent */
[F_CODE = 'BH140' and OSM_T_WAT <> 'drain'] [@sd < 200k] {
  stroke: #00A0C6;
  stroke-width: 10m;
}

/* @title River, Intermittent */
[F_CODE = 'BH140' and OSM_T_WAT = 'drain'] [@sd < 200k] {
  stroke: #00A0C6;
  stroke-width: 10m;
  stroke-dasharray: 10 2.5;
}
```

C.1.7. informationpnt.css

```
/* Geographic Name */
[OSM_T_IIN LIKE '%Syria%' or OSM_T_PLA = 'locality'] [@sd < 500k] {
  label: [OSM_T_ENG];
  font-family: "Open Sans";
  font-size: 10;
  font-fill: #000000;
  halo-color: #ffffff;
  halo-radius: 1;
  label-anchor: 0.5 0.5;
}
```

C.1.8. recreationsrf.css

```
/* @title Sports Ground */
[F_CODE = 'AK040'] [@sd < 80k] {
  fill-geometry: [the_geom], [boundary(the_geom)];
  fill: #E8F3E2, none;
  stroke: none, #3EA250;
  stroke-opacity: 0, 1;
  stroke-width: 5m;
  stroke-dasharray: 5 2.5;
  stroke-offset: 2;
}

[F_CODE = 'AK040'] [@sd > 80k] [@sd < 200k] {
  fill: url('file://symbols/sports_ground_wilson.svg');
  fill-mime: 'image/svg';
  fill-size: 6m;
}
```

C.1.9. settlementsrf.css

```
/* @title: Built Up Area */
[F_CODE = 'AL020'] [@sd < 200k] {
  fill: #E8C3B2;
  stroke: #000000;
  stroke-width: 5m;
}
```

C.1.10. transportationgroundcrv.css

```
/* @title: Road Primary */
[F_CODE = 'AP030' and RIN_ROI = '3'] {
  z-index: 0, 1;
  [@sd < 25M][@sd > 2.5M] {
    stroke-width: 1.14, 0.9;
```

```
    stroke: #000000, #cb171a;
  };
  [@sd < 2.5M][@sd > 500k] {
    stroke-width: 1.5, 1.14;
    stroke: #000000, #cb171a;
  };
  [@sd < 500k][@sd > 100k] {
    stroke-width: 3, 2.25;
    stroke: #000000, #cb171a;
  };
  [@sd < 100k][@sd > 5000] {
    stroke-width: 6.3, 4.655;
    stroke: #000000, #cb171a;
  };
  [@sd < 5000][@sd > 100] {
    stroke-width: 12, 8;
    stroke: #000000, #cb171a;
  };
}

/* @title: Road Secondary */
[F_CODE = 'AP030' and RIN_ROI = '4'] {
  z-index: 0, 1;
  [@sd < 25M][@sd > 2.5M] {
    stroke-width: 1.14, 0.9;
    stroke: #000000, #cb171a;
  };
  [@sd < 2.5M][@sd > 500k] {
    stroke-width: 1.5, 1.14;
    stroke: #000000, #cb171a;
  };
  [@sd < 500k][@sd > 100k] {
    stroke-width: 3, 2.25;
    stroke: #000000, #cb171a;
  };
  [@sd < 100k][@sd > 5000] {
    stroke-width: 6.3, 4.655;
    stroke: #000000, #cb171a;
  };
  [@sd < 5000][@sd > 100] {
    stroke-width: 12, 8;
    stroke: #000000, #cb171a;
  };
}

/* @title: Street */
[F_CODE = 'AP030' and RIN_ROI = '5'] {
  z-index: 0, 1;
  [@sd < 25M][@sd > 2.5M] {
    stroke-width: 0.855, 0.675;
    stroke: #000000, #ffffff;
```

```
  };
  [@sd < 2.5M][@sd > 500k] {
    stroke-width: 1.125, 0.855;
    stroke: #000000, #ffffff;
  };
  [@sd < 500k][@sd > 100k] {
    /* modified width */
    /*stroke-width: 2.250, 1.688;*/
    stroke-width: 1.9, 1.2;
    stroke: #000000, #ffffff;
  };
  [@sd < 100k][@sd > 5000] {
    /* modified width */
    /*stroke-width: 4.725, 3.491;*/
    stroke-width: 3.725, 2.491;
    stroke: #000000, #ffffff;
  };
  [@sd < 5000][@sd > 100] {
    stroke-width: 9, 6;
    stroke: #000000, #ffffff;
  };
}
```

C.1.11. transportationgroundcrv_highlight.css

```
/* @title: Bridge */
[F_CODE = 'AQ040' and TRS = '13'] {

  [@sd < 50k] {

    mark-geometry: [startPoint(the_geom)], [endPoint(the_geom)], [startPoint(the_geom
)], [endPoint(the_geom)];
    mark: symbol('shape://vertline');
    mark-rotation: [startAngle(the_geom) - 45], [endAngle(the_geom) + 45], [startAngl
e(the_geom) + 45], [endAngle(the_geom) - 45];
    mark-anchor: 0 0, 0 0, 0 1, 0 1;
    mark-offset: [10 * sin(toRadians(startAngle(the_geom)))] [-10 * cos(toRadians(sta
rtAngle(the_geom)))],
    [10 * sin(toRadians(endAngle(the_geom)))] [-10 * cos(toRadians(endAngle(the_geom)
))],
    [-10 * sin(toRadians(startAngle(the_geom)))] [10 * cos(toRadians(startAngle
(the_geom)))],
    [-10 * sin(toRadians(endAngle(the_geom)))] [10 * cos(toRadians(endAngle(the_geom)
))];

    :mark {
      stroke: #000000;
      stroke-width: 2;
      size: 15;
    };
```

```
    [@sd >= 5000] {
      :mark {
        stroke-width: 0.5;
        size: 5;
      };
      mark-offset: [6 * sin(toRadians(startAngle(the_geom)))] [-6 * cos(toRadians(sta
rtAngle(the_geom)))],
      [6 * sin(toRadians(endAngle(the_geom)))] [-6 * cos(toRadians(endAngle(the_geom)
))],
      [-6 * sin(toRadians(startAngle(the_geom)))] [6 * cos(toRadians(startAngle
(the_geom)))],
      [-6 * sin(toRadians(endAngle(the_geom)))] [6 * cos(toRadians(endAngle(the_geom)
))];
    };
  };
  [@sd < 100k][@sd >= 5000] {
    z-index: 1, 2;
    stroke-width: 12, 10;
    stroke: #000000, #ffffff;

  };
  [@sd < 5000][@sd > 100] {
    z-index: 1, 2;
    stroke-width: 20, 14;
    stroke: #000000, #ffffff;
  };
}

/*@title: Cart Track*/
[F_CODE = 'AP010'] {
  [@sd < 25M][@sd > 2.5M] {
    stroke-width: 1.14, 0.9, 0.9;
    stroke: #000000, #f7ab82, #ffffff;
    stroke-dasharray: 1 0, 1 0, 1.870 4.124 0.625 1.375;
  };
  [@sd < 2.5M][@sd > 500k] {
    stroke-width: 1.5, 1.14, 1.14;
    stroke: #000000, #f7ab82, #ffffff;
    stroke-dasharray: 1 0, 1 0, 2.415 5.323 0.807 1.774;
  };
  [@sd < 500k][@sd > 100k] {
    stroke-width: 3, 2.25, 2.25;
    stroke: #000000, #f7ab82, #ffffff;
    stroke-dasharray: 1 0, 1 0, 4.797 10.577 1.603 3.526;
  };
  [@sd < 100k][@sd > 5000] {
    stroke-width: 6.3, 4.655, 4.655;
    stroke: #000000, #f7ab82, #ffffff;
    stroke-dasharray: 1 0, 1 0, 10 22 3.333 7.333;
  };
  [@sd < 5000][@sd > 100] {
```

```
    stroke-width: 12, 8, 8;
    stroke: #000000, #f7ab82, #ffffff;
    stroke-dasharray: 1 0, 1 0, 18.100 39.820 6.033 13.273;
  };
}

/* @title: Trail */
[F_CODE = 'AP050' ] {
  [@sd < 100k][@sd > 5000] {
    stroke-width: 2.363, 2.363;
    stroke: #000000, #ffffff;
    stroke-dasharray: 1 0, 5 9 1.5 2.5;
  };
  [@sd < 5000][@sd > 100] {
    stroke-width: 4.5, 4.5;
    stroke: #000000, #ffffff;
    stroke-dasharray: 1 0, 10 18 3 5;
  };
}
```

C.1.12. utilityinfrastructurepnt.css

```
/* @title Pylon */
[F_CODE = 'AT042' and OSM_T_POW = 'tower'] [@sd < 200k] {
  mark: url('file://symbols/modified/pylon_wilson.svg');
  mark-mime: 'image/svg';
  mark-size: 250m;
  [@sd < 35k] {
    label: 'Pylon';
    font-family: "Open Sans";
    font-size: 12;
    font-fill: #000000;
    halo-color: #ffffff;
    halo-radius: 1;
    label-anchor: 0.5 2.5;
  }

}

/* @title Vertical Obstruction */
[F_CODE = 'AT042' and HGT >= 46 and OSM_T_POW = 'tower'] [@sd < 200k] {
  mark: url('file://symbols/PT_Blue072_VO.svg');
  mark-mime: 'image/svg';
  mark-size: 600m;
  [@sd < 35k] {
    label: 'Pylon';
    font-family: "Open Sans";
    font-size: 12;
    font-fill: #000000;
    halo-color: #ffffff;
    halo-radius: 1;
    label-anchor: 0.5 2.5;
  }
}
```

C.1.13. utilityinfrastructurecrv.css

```
/*@title: Powerline, Non Obstruction */
[F_CODE = 'AT005' and OSM_T_POW = 'line'] [@sd < 200k] {
  stroke: #473895;
  stroke-width: 20m;
}
```

C.1.14. structurepnt.css

```
/* @title: Building, General */
[F_CODE = 'AL013'] {
  [@sd < 100k] {
    mark: url('file://symbols/building.svg');
    mark-mime: 'image/svg';
    mark-size: 40m;
  };
  [@sd > 100k] [@sd < 200k]  {
    mark: url('file://symbols/building.svg');
    mark-mime: 'image/svg';
    mark-size: 3m;
  };
}

/* @title: Building, Mosque */
[F_CODE = 'AL013' and OSM_T_REL = 'muslim' and OSM_T_AME = 'place_of_worship'] {
  [@sd < 100k] {
    mark: url('file://symbols/AR_Black_Mosque.svg');
    mark-mime: 'image/svg';
    mark-size: 130m;
  };
  [@sd > 100k] [@sd < 200k]  {
    mark: url('file://symbols/PT_Black_Mosque.svg');
    mark-mime: 'image/svg';
    mark-size: 15m;
  };
}

/* @title: Building, School */
[F_CODE = 'AL013' and OSM_T_AME = 'college'] {
  [@sd < 100k] {
    mark: url('file://symbols/PT_Black_School.svg');
    mark-mime: 'image/svg';
    mark-size: 130m;
  };
  [@sd > 100k] [@sd < 200k]  {
    mark: url('file://symbols/PT_Black_School.svg');
    mark-mime: 'image/svg';
    mark-size: 15m;
  };
}

/* @title: Building, Public */
[F_CODE = 'AL013' and OSM_T_AME = 'public_building'] [@sd < 200k] {
  mark: url('file://symbols/building.svg');
  mark-mime: 'image/svg';
  mark-size: 70m;
}
```

C.1.15. MIL-STD2525D.css

```
[@sd < 250k] {
  [StndID_1 = 0] {
    [StndID_2 in (2, 3, 4, 5)] {
      mark-mime: 'image/svg';
      mark-size: 64;
      z-index: 0;
      [StndID_2 = 2] {
        mark: url('file://symbols/2525/Frames/0310_Friend.svg');
      };
      [StndID_2 = 3] {
        mark: url('file://symbols/2525/Frames/0410_Neutral.svg');
      };
      [StndID_2 = 4] {
        mark: url('file://symbols/2525/Frames/0510_Suspect_Joker.svg');
      };
      [StndID_2 = 5] {
        mark: url('file://symbols/2525/Frames/0600_Hostile_Faker.svg');
      };
    };
  };
  [Entity = 28] {
    z-index: 1;
    [EntityT = 2] {
      [EntityST = 1] {
        mark: url('file://symbols/2525/Symbol/25280201_Antipersonnel Mine with
Directional Effects.svg');
        mark-mime: 'image/svg';
        mark-size: 40;
        mark-rotation: [Angle - 90];
      };
    };
  };
  [Entity = 11] {
    z-index: 1;
    [EntityT = 4] {
      [EntityST = 0] {
        mark: url('file://symbols/2525/Symbol/11110400_Organization or Group.svg');
        mark-mime: 'image/svg';
        mark-size: 64;
      };
    };
  };
  [Entity = 19] {
    z-index: 1;
    [EntityT = 0] {
      [EntityST = 0] {
        mark: url('file://symbols/2525/Symbol/10190000_Emergency_Operation.svg');
        mark-mime: 'image/svg';
```

```
        mark-size: 64;
      };
    };
  };
}
```

C.1.16. landsat8.css

```
/* @title Landsat 8 */
* {
  raster-channels: 1 2 3;
  raster-contrast-enhancement: normalize;
  raster-contrast-enhancement-algorithm: StretchToMinimumMaximum;
  raster-contrast-enhancement-min: 5000;
  raster-contrast-enhancement-max: 20000;
}
```

Appendix D: CartoCSS Style Encoding

This appendix shows the CartoCSS Style Encoding for the styles described in the chapter 7.

CartoCSS Style Encoding for the styles described in the chapter 7

```
//AgricultureSrf

#agriculturesrf [F_CODE = 'EA010'] [OSM_T_LAN = 'farmland'][zoom > 11]{
  polygon-fill:#F7EDED;
  polygon-pattern-file: url("Green_AP77_Fill.svg");
  polygon-pattern-opacity: 0.5;
  polygon-pattern-gamma: 1;
  polygon-pattern-clip: true;
}

//CultureSrf
#culturesrf [F_CODE = "AL030"][OSM_T_LAN = 'cemetery'][zoom > 11]{
  line-color:#000000;
  line-width:1;
  line-dasharray: 5, 2.5;
}

//Emergency
#emergency [zoom > 11] {
  [type = 'reunification']{
        marker-file: url(civilian-staging-transit.svg);
        marker-width:25;
        marker-height:25;
  }
  [type = 'info_point']{
        marker-file: url(public-information.svg);
        marker-width:25;
        marker-height:25;
  }
}

//HydrographyCrv
// River, Permanent
#hydrographycrv[F_CODE = 'BH140'][OSM_T_WAT != 'drain']{
  [zoom > 11] {
      line-width:3;
      line-color:#00A0C6;
  }
  [zoom >= 15] {
      line-width:5;
      line-color:#00A0C6;
  }
  [zoom >= 17] {
      line-width:10;
```

```
      line-color:#00A0C6;
  }
}
//River, Intermittent
#hydrographycrv[F_CODE = 'BH140'][OSM_T_WAT = 'drain'] {
  [zoom > 11] {
      line-width:3;
      line-dasharray: 10, 2.5;
      line-color:#00A0C6;
  }
  [zoom >= 15] {
      line-width:5;
      line-dasharray: 10, 2.5;
      line-color:#00A0C6;
  }
  [zoom >= 17] {
      line-width:10;
      line-dasharray: 10, 2.5;
      line-color:#00A0C6;
  }
}

//HydrographySrf
//Lake
#hydrographysrf [F_CODE='BH082'][zoom > 11]::z1 {
  line-color:#00A0C6;
  line-width:4;
  polygon-opacity:1;
  polygon-fill:#B0E1ED;
}
//Dam
#hydrographysrf[F_CODE='BI020'][zoom > 11] ::z2{
  line-color:#000000;
  line-width:4;
  marker-file: url('PT_Black_Dam_2.svg');
  marker-placement: interior;
  marker-width: 10;
}

//InformationPnt
#informationpnt[OSM_T_IIN=~'Syria.*'][OSM_T_PLA = 'locality'] {
  [zoom >= 10]{
    text-name: [OSM_T_ENG];
    text-face-name: 'Open Sans Regular','Noto','Hanazono','Unifont','DejaVu';
    text-size: 10;
    text-fill: #000000;
    text-halo-fill: #ffffff;
    text-halo-radius: 1;
  }
}
```

```
//Milstandard2525
#military [zoom > 11]{
  [StndID_1 = 0][StndID_2 != 999] {
    marker-width:30;

    [StndID_2 = 2] {
      marker-file: url('0310_Friend.svg');
    }
     [StndID_2 = 3] {
      marker-file: url('0410_Neutral.svg');
    }

    [StndID_2 = 4] {
      marker-file: url('0510_Suspect_Joker.svg');
    }

      [StndID_2 = 5] {
     marker-file: url('0600_Hostile_Faker.svg');
    }

  }

/*symbol*/
::symbol{
      [Entity = 28]{
        marker-file: url('25280201_Antipersonnel_Mine_with_Directional_Effects.svg');
        marker-transform: rotate([Angle]-90);
        marker-width:20;
        marker-allow-overlap:true;
    }

      [Entity = 19] {
        marker-file: url('10190000_Emergency_Operation.svg');
        marker-width:15;
        marker-allow-overlap:true;
    }
      [Entity = 11] {
        marker-file: url('11110400_Organization_or_Group.svg');
        marker-width:20;
        marker-allow-overlap:true;
    }
  }
}

//RecreationSrf
#recreationsrf [F_CODE="AK040"]{
  [zoom > 13]{
     line-color:#3EA250;
     line-width:5;
     polygon-fill:#E8F3E2;
     line-dasharray: 5, 2.5;
```

```
      line-offset: 2;
      line-opacity:1;
      polygon-opacity:1;
   }
   [zoom > 11] [zoom <= 13]{
      marker-file: url("sports_ground_wilson.svg");
      }
}

//SettlementSrf
#settlementsrf [F_CODE = "AL020"] {
  polygon-opacity:1;
  polygon-fill:#E8C3B2;
  line-color: #000000;
  line-width: 2;
}

//StructurePnt
#structurepnt[F_CODE = 'AL013'] {
    [zoom > 12] {
      marker-file: url('building1.svg');
      marker-width:10;
      marker-height:10;
    }
    [zoom > 10][zoom <= 12]  {
      marker-file:url('building1.svg');
      marker-width:5;
      marker-height:5;
    }
}
#structurepnt[F_CODE = 'AL013'][OSM_T_REL = 'muslim'][OSM_T_AME = 'place_of_worship']
{
  [zoom > 12]{
      marker-file: url(AR_Black_Mosque.svg);
      marker-width:20;
      marker-height:20; }
  [zoom >= 10][zoom <= 12]{
      marker-file: url(PT_Black_Mosque.svg);
      marker-width:5;
      marker-height:5; }
}

#structurepnt[F_CODE = 'AL013'][OSM_T_AME = 'college']  {
  [zoom > 12]{
      marker-file: url(PT_Black_School.svg);
      marker-width:20;
      marker-height:20; }
  [zoom >= 10][zoom <= 12]{
      marker-file: url(PT_Black_School.svg);
      marker-width:5;
      marker-height:5; }
```

```
}

#structurepnt[F_CODE = 'AL013'][OSM_T_AME = 'public_building'] [zoom > 11] {
   marker-file: url(building1.svg);
   marker-width:15;
   marker-height:15;
}

//TransportationGroundCrv
// Primary Road
#transportationgroundcrv [F_CODE = 'AP030'] [RIN_ROI = 3] {
  [zoom >= 19][zoom <= 20] {
    ::case {
      line-width:27;
      line-color:#000000;
    }
    ::fill {
      line-width:22.5;
      line-color:#cb171a;
      }
 }
    [zoom >= 18][zoom < 19] {
    ::case {
      line-width:21;
      line-color:#000000;
    }
    ::fill {
      line-width:17.1;
      line-color:#cb171a;
      }
 }
    [zoom >= 17][zoom < 18] {
    ::case {
      line-width:18;
      line-color:#000000;
    }
    ::fill {
      line-width:14.4;
      line-color:#cb171a;
      }
 }
    [zoom >= 15][zoom < 17] {
    ::case {
      line-width:10;
      line-color:#000000;
    }
    ::fill {
      line-width:7.74;
      line-color:#cb171a;
      }
 }
```

```
    [zoom >= 12][zoom < 15] {
    ::case {
      line-width:8;
      line-color:#000000;
    }
    ::fill {
      line-width:6.12;
      line-color:#cb171a;
      }
  }
    [zoom >= 11][zoom < 12] {
    ::case {
      line-width:6;
      line-color:#000000;
    }
    ::fill {
      line-width:4.5;
      line-color:#cb171a;
      }
  }
    [zoom >= 10][zoom < 11] {
     line-width:2;
     line-color:#cb171a;
  }
    [zoom >= 9][zoom < 10] {
     line-width:1.9;
     line-color:#cb171a;
  }
    [zoom >= 8][zoom < 9] {
     line-width:1.4;
     line-color:#cb171a;
  }
    [zoom >= 7][zoom < 8] {
     line-width:1;
     line-color:#cb171a;
  }
    [zoom >= 5][zoom < 7] {
     line-width:0.8;
     line-color:#cb171a;
  }
   [zoom < 5] {
     line-width:0.5;
     line-color:#cb171a;
  }
}
// Secondary Road
#transportationgroundcrv [F_CODE = 'AP030'] [RIN_ROI = 4] {
    [zoom >= 19][zoom <= 20] {
    ::case {
      line-opacity:0.4;
      line-width:27;
```

```
      line-color:#000000;
    }
    ::fill {
      line-opacity:1;
      line-width:25;
      line-color:#cb171a;
      }
}
 [zoom >= 18][zoom < 19] {
    ::case {
      line-opacity:0.4;
      line-width:21;
      line-color:#000000;
    }
    ::fill {
      line-opacity:1;
      line-width:19;
      line-color:#cb171a;
      }
}
 [zoom >= 17][zoom < 18] {
    ::case {
      line-opacity:0.4;
      line-width:18;
      line-color:#000000;
    }
    ::fill {
      line-opacity:1;
      line-width:16;
      line-color:#cb171a;
      }
}
     [zoom >= 16][zoom < 17] {
    ::case {
      line-opacity:0.4;
      line-width:10;
      line-color:#000000;
    }
    ::fill {
      line-opacity:1;
      line-width:8.5;
      line-color:#cb171a;
      }
}
 [zoom >= 15][zoom < 16] {
    ::case {
      line-opacity:0.4;
      line-width:9;
      line-color:#000000;
    }
    ::fill {
```

```
      line-opacity:1;
      line-width:7.5;
      line-color:#cb171a;
      }
 }
  [zoom >= 12][zoom < 15] {
    ::case {
      line-opacity:0.4;
      line-width:5;
      line-color:#000000;
    }
    ::fill {
      line-opacity:1;
      line-width:4;
      line-color:#cb171a;
      }
 }
  [zoom >= 11][zoom < 12] {
    ::case {
      line-opacity:0.4;
      line-width:0.8;
      line-color:#9eae23;
    }
    ::fill {
      line-opacity:1;
      line-width:0.8;
      line-color:#cb171a;
      }
 }
[zoom >= 5] [zoom < 11]{
    line-width:0.8;
    line-color:#cb171a;
 }
 }
//Street
#transportationgroundcrv [F_CODE = 'AP030'] [RIN_ROI = 5] {
    [zoom >= 19][zoom <= 20] {
    ::case {
      line-width:27;
      line-color:#000000;
    }
    ::fill {
      line-width:25;
      line-color:#ffffff;
      }
 }
    [zoom >= 18][zoom < 19] {
    ::case {
      line-width:21;
      line-color:#000000;
    }
```

```
    ::fill {
      line-width:19;
      line-color:#ffffff;
      }
  }
    [zoom >= 17][zoom < 18] {
    ::case {
      line-width:18;
      line-color:#000000;
    }
    ::fill {
      line-width:16;
      line-color:#ffffff;
      }
  }
   [zoom >= 16][zoom < 17] {
    ::case {
      line-width:10;
      line-color:#000000;
    }
    ::fill {
      line-width:8.5;
      line-color:#ffffff;
      }
  }
    [zoom >= 15][zoom < 16] {
    ::case {
      line-width:9;
      line-color:#000000;
    }
    ::fill {
      line-width:7.5;
      line-color:#ffffff;
      }
  }
    [zoom >= 13][zoom < 15] {
    ::case {
      line-width:4;
      line-color:#000000;
    }
    ::fill {
      line-width:3;
      line-color:#ffffff;
      }
  }
   [zoom >= 12][zoom < 13] {
    ::case {
      line-width:2;
      line-color:#000000;
    }
    ::fill {
```

```
      line-width:1.2;
      line-color:#ffffff;
      }
 }
  [zoom >= 11] [zoom < 12]{
    ::case {
      line-width:1;
      line-color:#000000;
    }
    ::fill {
      line-width:0.8;
      line-color:#ffffff;
      }
 }

}

//TransportationGroundCrv_Highlight
#transportationgroundcrv_highlight [F_CODE = 'AQ040'] [TRS = 13] {
  [zoom >= 14] [zoom < 16]{
    ::background {
       line-offset: 8;
       line-color:#000000;
       line-width: 0.5;
       marker-file: url('horline_half.svg');
       marker-placement: vertex-first;
       marker-transform: 'rotate(-45,0,0) translate(-2.5,0)';
       marker-offset: 8;
       marker-allow-overlap: true;
       ::foreground {
         line-offset: 4;
         line-color:#ffffff;
         line-width: 4;
         marker-file: url('horline_half.svg');
         marker-placement: vertex-last;
         marker-transform: 'rotate(45,0,0) translate(2.5,0)';
         marker-offset: 8;
         marker-allow-overlap: true;
       }
    }
    ::foreground {
      line-offset: -8;
      line-color:#000000;
      line-width: 0.5;
      marker-file: url('horline_half.svg');
      marker-placement: vertex-last;
      marker-transform: 'rotate(-45,0,0) translate(2.5,0)';
      marker-offset: -8;
      marker-allow-overlap: true;
      ::background {
         line-offset: -4;
```

```
        line-color:#ffffff;
        line-width: 4;
        marker-file: url('horline_half.svg');
        marker-placement: vertex-first;
        marker-transform: 'rotate(45,0,0) translate(-2.5,0)';
        marker-offset: -8;
        marker-allow-overlap: true;
      }
    }
  }
  [zoom >= 16] [zoom < 17]{
    ::background {
       line-offset: 9;
       line-color:#000000;
       line-width: 1;
       marker-file: url('horline_one.svg');
       marker-placement: vertex-first;
       marker-transform: 'rotate(-45,0,0) translate(-5,0)';
       marker-offset: 9;
       marker-allow-overlap: true;
       ::foreground {
         line-offset: 5;
         line-color:#ffffff;
         line-width: 4;
         marker-file: url('horline_one.svg');
         marker-placement: vertex-last;
         marker-transform: 'rotate(45,0,0) translate(5,0)';
         marker-offset: 9;
         marker-allow-overlap: true;
       }
    }
    ::foreground {
      line-offset: -9;
      line-color:#000000;
      line-width: 1;
      marker-file: url('horline_one.svg');
      marker-placement: vertex-last;
      marker-transform: 'rotate(-45,0,0) translate(5,0)';
      marker-offset: -9;
      marker-allow-overlap: true;
      ::background {
         line-offset: -5;
         line-color:#ffffff;
         line-width: 4;
         marker-file: url('horline_one.svg');
         marker-placement: vertex-first;
         marker-transform: 'rotate(45,0,0) translate(-5,0)';
         marker-offset: -9;
         marker-allow-overlap: true;
      }
    }
```

```
}
[zoom >= 17] [zoom < 18]{
  ::background {
     line-offset: 12;
     line-color:#000000;
     line-width: 1;
     marker-file: url('horline_one.svg');
     marker-placement: vertex-first;
     marker-transform: 'rotate(-45,0,0) translate(-5,0)';
     marker-offset: 12;
     marker-allow-overlap: true;
     ::foreground {
       line-offset: 10;
       line-color:#ffffff;
       line-width: 2;
       marker-file: url('horline_one.svg');
       marker-placement: vertex-last;
       marker-transform: 'rotate(45,0,0) translate(5,0)';
       marker-offset: 12;
       marker-allow-overlap: true;
     }
  }
  ::foreground {
    line-offset: -12;
    line-color:#000000;
    line-width: 1;
    marker-file: url('horline_one.svg');
    marker-placement: vertex-last;
    marker-transform: 'rotate(-45,0,0) translate(5,0)';
    marker-offset: -12;
    marker-allow-overlap: true;
    ::background {
       line-offset: -10;
       line-color:#ffffff;
       line-width: 2;
       marker-file: url('horline_one.svg');
       marker-placement: vertex-first;
       marker-transform: 'rotate(45,0,0) translate(-5,0)';
       marker-offset: -12;
       marker-allow-overlap: true;
    }
  }
}

[zoom >= 18][zoom <= 20]{
  ::background {
    line-offset: 18;
    line-color:#000000;
    line-width: 2;
    marker-file: url('horline_two.svg');
    marker-placement: vertex-first;
```

```
      marker-transform: 'rotate(-45,0,0) translate(-5,0)';
      marker-offset: 18;
      marker-allow-overlap: true;
      ::foreground {
        line-offset: 14;
        line-color:#ffffff;
        line-width: 6;
        marker-file: url('horline_two.svg');
        marker-placement: vertex-last;
        marker-transform: 'rotate(45,0,0) translate(5,0)';
        marker-offset: 18;
        marker-allow-overlap: true;
      }
    }
    ::foreground {
      line-offset: -18;
      line-color:#000000;
      line-width: 2;
      marker-file: url('horline_two.svg');
      marker-placement: vertex-last;
      marker-transform: 'rotate(-45,0,0) translate(5,0)';
      marker-offset: -18;
      marker-allow-overlap: true;
      ::background {
        line-offset: -14;
        line-color:#ffffff;
        line-width: 6;
        marker-file: url('horline_two.svg');
        marker-placement: vertex-first;
        marker-transform: 'rotate(45,0,0) translate(-5,0)';
        marker-offset: -18;
        marker-allow-overlap: true;
      }
    }
  }
 }


// Cart Track
 #transportationgroundcrv [F_CODE = 'AP010']   {
  [zoom >=17][zoom <=20]{
    ::case {
      line-dasharray: 1.0, 0.0;
      line-width:12;
      line-color:#000000;
    }
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width:8;
        line-color:#f7ab32;
```

```
      }
      ::backgroung{
        line-dasharray: 18.100, 39.820, 6.033, 13.273;
        line-width:8;
        line-color:#ffffff;
      }
    }
  }
  [zoom >=12][zoom < 17]{
    ::case {
      line-dasharray: 1.0, 0.0;
      line-width:6.3;
      line-color:#000000;
    }
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width: 4.655;
        line-color:#f7ab32;
      }
      ::backgroung{
        line-dasharray: 10, 22, 3.333, 7.333;
        line-width:4.655;
        line-color:#ffffff;
      }
    }
  }
  [zoom >=10][zoom < 12]{
    ::case {
      line-dasharray: 1.0, 0.0;
      line-width: 3;
      line-color:#000000;
    }
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width:2.25;
        line-color:#f7ab32;
      }
      ::backgroung{
        line-dasharray: 4.797, 10.577, 1.603, 3.526;
        line-width: 2.25;
        line-color:#ffffff;
      }
    }
  }
  [zoom >= 7][zoom < 10]{
    ::case {
      line-dasharray: 1.0, 0.0;
      line-width: 1.5;
      line-color:#000000;
```

```
    }
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width: 1.14;
        line-color:#f7ab32;
      }
      ::backgroung{
        line-dasharray: 2.415, 5.323, 0.807, 1.774;
        line-width: 1.14;
        line-color:#ffffff;
      }
    }
  }
  [zoom >= 4][zoom < 7]{
    ::case {
      line-dasharray: 1.0, 0.0;
      line-width:1.14;
      line-color:#000000;
    }
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width: 0.9;
        line-color:#f7ab32;
      }
      ::backgroung{
        line-dasharray: 1.870, 4.124, 0.625, 1.375;
        line-width: 0.9;
        line-color:#ffffff;
      }
    }
  }
}

//Trail
#transportationgroundcrv [F_CODE = 'AP050']   {
  [zoom >=17][zoom <=20]{
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width:4.5;
        line-color:#000000;
      }
      ::backgroung{
        line-dasharray: 10, 18, 3, 5;
        line-width:4.5;
        line-color:#ffffff;
      }
    }
  }
```

```
    [zoom >= 12][zoom < 17]{
    ::fill {
      ::foreground{
        line-dasharray: 1.0, 0.0;
        line-width:2.363;
        line-color:#000000;
      }
      ::backgroung{
        line-dasharray: 5, 9, 1.5, 2.5;
        line-width: 2.363;
        line-color:#ffffff;
      }
    }
  }
}

//UtilityInfrastructureCrv
#utilityinfrastructurecrv [F_CODE = 'AT005'][OSM_T_POW = 'line']   {
    [zoom >= 11]{
        line-width:2;
        line-color:#473895;
    }
}

//UtilityInfrastructurePnt
#utilityinfrastructurepnt [F_CODE = 'AT042'][OSM_T_POW = 'tower']   {
    [zoom >= 13] [zoom < 15]{
        marker-file: url(pylon_wilson.svg);
        marker-width:15;
        marker-allow-overlap: true;
    }
    [zoom >= 15] [zoom < 20]{
        marker-file: url(pylon_wilson.svg);
        marker-width:20;
        text-name: 'Pylon';
        text-face-name: 'Open Sans Regular';
        text-size: 12;
        text-fill: #000000;
        text-halo-fill: #ffffff;
        text-halo-radius: 1;
        text-allow-overlap:true;
        text-opacity: 1;
        text-dx: 0;
        text-dy: 18;
    }
}

//VegetationSrf
#vegetationsrf [F_CODE = 'EC015'][OSM_T_LAN = 'forest'] {
  polygon-fill:#C2E4B9;
}
```

Appendix E: GeoPackage Style Encoding

This appendix shows the GeoPackage JSON based Style Encoding for the styles derived from the source encoding.

```
//AgricultureSrf
{
  "symbol-mapping": {
    "table": "AgricultureSrf",
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "EA010"
      }],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 200000,
          "symbols": [{
              "fill": {
                "color": "0xf7eded",
                "opacity": 1.0,
                "image": {
                  "name": "Green_AP77_Fill.svg",
                  "scale": 10,
                  "apply-gpkg-angle": false,
                  "angle-offset": 0,
                  "format": "image/svg+xml"
                }
              }
            },
            {
              "stroke": {
                "color": "0x0",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 5,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
```

```
                }
              }
            }
          ]
        }]
      }
    }]
  }
}

//CultureSrf
{
  "symbol-mapping": {
    "table": "CultureSrf",
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "AL030"
        },
        {
          "operand": "OSM_T_LAN",
          "operator": "=",
          "value": "cemetery"
        }
      ],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 200000,
          "symbols": [{
            "stroke": {
              "color": "0x0",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 5,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [5, 2.5, 0, 0],
              "cap": "",
              "casing": {
                "color": "0x0",
                "opacity": 1.0,
                "width": 0
              }
            }
          }]
```

```
        }]
      }
    }]
  }
}

//Emergency
{
  "symbol-mapping": {
    "table": "emergency",
    "geometry-type": "POINT"
  },
  "style-rule": {
    "conditions": [{
        "criteria": [{
          "operand": "type",
          "operator": "=",
          "value": "reunification"
        }],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 35000,
            "symbols": [{
              "image": {
                "name": "civilian-staging-transit.svg",
                "scale": 1.0,
                "apply-gpkg-angle": false,
                "angle-offset": 0,
                "format": "image/svg+xml"
              }
            }]
          }]
        }
      },
      {
        "criteria": [{
          "operand": "type",
          "operator": "=",
          "value": "info_point"
        }],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 35000,
            "symbols": [{
              "image": {
                "name": "public-information.svg",
                "scale": 1.0,
                "apply-gpkg-angle": false,
                "angle-offset": 0,
```

```
                "format": "image/svg+xml"
              }
            }]
          }]
        }
      }
    ]
  }
}

//HydrographyCrv
{
  "symbol-mapping": {
    "table": "HydrographyCrv",
    "geometry-type": "MULTILINESTRING"
  },
  "style-rule": {
    "conditions": [{
        "criteria": [{
            "operand": "F_CODE",
            "operator": "=",
            "value": "BH140"
          },
          {
            "operand": "OSM_T_LAN",
            "operator": "!=",
            "value": "drain"
          }
        ],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 200000,
            "symbols": [{
              "stroke": {
                "color": "0x00a0c6",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 10,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
                }
              }
            }]
```

```
          }]
        }
      },
      {
        "criteria": [{
            "operand": "F_CODE",
            "operator": "=",
            "value": "BH140"
          },
          {
            "operand": "OSM_T_LAN",
            "operator": "=",
            "value": "drain"
          }
        ],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 200000,
            "symbols": [{
              "stroke": {
                "color": "0x00a0c6",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 10,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [10, 2.5, 0, 0],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
                }
              }
            }]
          }]
        }
      }
    ]
  }
}

//HydrographySrf
{
  "symbol-mapping": {
    "table": "HydrographySrf",
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
```

```
    "conditions": [{
        "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "BH082"
        }],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 200000,
            "symbols": [{
              "fill": {
                "color": "0xb0e1ed",
                "opacity": 1.0,
                "image": {
                  "name": "",
                  "scale": 0,
                  "apply-gpkg-angle": false,
                  "angle-offset": 0,
                  "format": ""
                }
              }
            }]
          }]
        }
      },
      {
        "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "BI020"
        }],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 500000,
            "symbols": [{
              "stroke": {
                "color": "0x333333",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 20,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
```

```
                }
              }
            }]
          }]
        }
      }
    ]
  }
}

//InformationPnt
{
  "symbol-mapping": {
    "table": "InformationPnt",
    "geometry-type": "POINT"
  },
  "style-rule": {
    "conditions": [{
        "criteria": [{
            "operand": "OSM_T_LAN",
            "operator": "Contains",
            "value": "Syria"
          },
          {
            "operand": "OSM_T_PLA",
            "operator": "!=",
            "value": "locality"
          }
        ],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 500000,
            "symbols": [{
              "label": {
                "text": "OSM_T_ENG",
                "color": "0x0",
                "opacity": 1.0,
                "font": "OpenSans",
                "font-size": 10,
                "outline-color": "0xFFFFFF",
                "outline-opacity": 1.0,
                "outline-width": 1,
                "vertical-offset": 0,
                "horizontal-offset": 0
              }
            }]
          }]
        }
      },
      {
```

```
        "criteria": [{
          "operand": "OSM_T_PLA",
          "operator": "=",
          "value": "locality"
        }],
        "style": {
          "map-scales": [{
            "min": 0,
            "max": 500000,
            "symbols": [{
              "label": {
                "text": "OSM_T_ENG",
                "color": "0x0",
                "opacity": 1.0,
                "font": "OpenSans",
                "font-size": 10,
                "outline-color": "0xFFFFFF",
                "outline-opacity": 1.0,
                "outline-width": 1,
                "vertical-offset": 0,
                "horizontal-offset": 0
              }
            }]
          }]
        }
      }
    ]
  }
}

//Milstandard2525
{
  "symbol-mapping": {
    "table": "MilStandard2525_Project",
    "geometry-type": "POINT"
  },
  "style-rule": {
    "conditions": [{
        "criteria": [{
            "operand": "StndID_1",
            "operator": "=",
            "value": 0
          },
          {
            "operand": "StndID_2",
            "operator": "=",
            "value": 2
          }
        ],
        "style": {
          "map-scales": [{
```

```
          "symbols": [{
            "image": {
              "name": "0310_Friend.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
      "criteria": [{
          "operand": "StndID_1",
          "operator": "=",
          "value": 0
        },
        {
          "operand": "StndID_2",
          "operator": "=",
          "value": 3
        }
      ],
      "style": {
        "map-scales": [{
          "symbols": [{
            "image": {
              "name": "0410_Neutral.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
      "criteria": [{
          "operand": "StndID_1",
          "operator": "=",
          "value": 0
        },
        {
          "operand": "StndID_2",
          "operator": "=",
          "value": 4
        }
      ],
      "style": {
```

```
        "map-scales": [{
          "symbols": [{
            "image": {
              "name": "0510_Suspect_Joker.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
      "criteria": [{
          "operand": "StndID_1",
          "operator": "=",
          "value": 0
        },
        {
          "operand": "StndID_2",
          "operator": "=",
          "value": 5
        }
      ],
      "style": {
        "map-scales": [{
          "symbols": [{
            "image": {
              "name": "0600_Hostile_Faker.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
      "criteria": [{
          "operand": "Entity",
          "operator": "=",
          "value": 11
        },
        {
          "operand": "EntityT",
          "operator": "=",
          "value": 4
        },
        {
```

```
          "operand": "EntityST",
          "operator": "=",
          "value": 0
        }
      ],
      "style": {
        "map-scales": [{
          "symbols": [{
            "image": {
              "name": "11110400_Organization or Group.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
      "criteria": [{
          "operand": "Entity",
          "operator": "=",
          "value": 19
        },
        {
          "operand": "EntityT",
          "operator": "=",
          "value": 0
        },
        {
          "operand": "EntityST",
          "operator": "=",
          "value": 0
        }
      ],
      "style": {
        "map-scales": [{
          "symbols": [{
            "image": {
              "name": "10190000_Emergency_Operation.svg",
              "scale": 0.15,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }]
      }
    },
    {
```

```
        "criteria": [{
            "operand": "Entity",
            "operator": "=",
            "value": 28
          },
          {
            "operand": "EntityT",
            "operator": "=",
            "value": 2
          },
          {
            "operand": "EntityST",
            "operator": "=",
            "value": 1
          }
        ],
        "style": {
          "map-scales": [{
            "symbols": [{
              "image": {
                "name": "25280201_Antipersonnel Mine with Directional Effects.svg",
                "scale": 0.15,
                "apply-gpkg-angle": true,
                "angle-offset": -90,
                "format": "image/svg+xml"
              }
            }]
          }]
        }
      }
    ]
  }
}

//RecreationSrf
{
  "symbol-mapping": {
    "table": "RecreationSrf",
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AK040"
      }],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 200000,
```

```
          "symbols": [{
             "fill": {
               "color": "0xe8f3e2",
               "opacity": 1.0,
               "image": {
                 "name": "",
                 "scale": 0,
                 "apply-gpkg-angle": false,
                 "angle-offset": 0,
                 "format": ""
               }
             }
           },
           {
             "stroke": {
               "color": "0x3ea250",
               "opacity": 1.0,
               "background-color": "0xFFFFFF",
               "background-opacity": 1.0,
               "width": 5,
               "vertical-offset": 2,
               "horizontal-offset": 0,
               "dashes": [5, 2.5, 0, 0],
               "cap": "",
               "casing": {
                 "color": "0x0",
                 "opacity": 1.0,
                 "width": 0
               }
             }
           },
           {
             "fill": {
               "color": "0x0",
               "opacity": 0,
               "image": {
                 "name": "sports_ground_wilson.svg",
                 "scale": 6,
                 "apply-gpkg-angle": false,
                 "angle-offset": 0,
                 "format": "image/svg+xml"
               }
             }
           }
         ]
       }]
     }
   }]
  }
}
```

```
//SettlementSrf
{
  "symbol-mapping": {
    "table": "SettlementSrf",
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AL020"
      }],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 200000,
          "symbols": [{
              "fill": {
                "color": "0xe8c3b2",
                "opacity": 1.0,
                "image": {
                  "name": "",
                  "scale": 0,
                  "apply-gpkg-angle": false,
                  "angle-offset": 0,
                  "format": ""
                }
              }
            },
            {
              "stroke": {
                "color": "0x000",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 5,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "",
                "casing": {
                  "color": "0x000",
                  "opacity": 1.0,
                  "width": 0
                }
              }
            }
          ]
        }]
      }
```

```
    }]
  }
}

//StructurePnt
{
  "symbol-mapping": {
    "table": "StructurePnt",
    "geometry-type": "POINT"
  },
  "style-rule": {
    "conditions": [{
        "criteria": [{
            "operand": "F_CODE",
            "operator": "=",
            "value": "AL013"
          },
          {
            "operand": "OSM_T_AME",
            "operator": "!=",
            "value": "place_of_worship"
          },
          {
            "operand": "OSM_T_REL",
            "operator": "!=",
            "value": "muslim"
          },
          {
            "operand": "OSM_T_AME",
            "operator": "!=",
            "value": "public_building"
          },
          {
            "operand": "OSM_T_AME",
            "operator": "!=",
            "value": "gas_station"
          }
        ],
        "style": {
          "map-scales": [{
            "min": 100000,
            "max": 200000,
            "symbols": [{
              "image": {
                "name": "building.svg",
                "scale": 0.07,
                "apply-gpkg-angle": false,
                "angle-offset": 0,
                "format": "image/svg+xml"
              }
            }]
```

```
        }, {
          "min": 0,
          "max": 100000,
          "symbols": [{
            "image": {
              "name": "building.svg",
              "scale": 0.14,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }],
      }
    },
    {
      "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "AL013"
        },
        {
          "operand": "OSM_T_AME",
          "operator": "=",
          "value": "place_of_worship"
        },
        {
          "operand": "OSM_T_REL",
          "operator": "=",
          "value": "muslim"
        }
      ],
      "style": {
        "map-scales": [{
          "min": 100000,
          "max": 200000,
          "symbols": [{
            "image": {
              "name": "PT_Black_Mosque.svg",
              "scale": 1.0,
              "apply-gpkg-angle": false,
              "angle-offset": 0,
              "format": "image/svg+xml"
            }
          }]
        }, {
          "min": 0,
          "max": 100000,
          "symbols": [{
            "image": {
              "name": "AR_Black_Mosque.svg",
```

```
            "scale": 8.6,
            "apply-gpkg-angle": false,
            "angle-offset": 0,
            "format": "image/svg+xml"
          }
        }]
      }]
    }
  },
  {
    "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AL013"
      },
      {
        "operand": "OSM_T_AME",
        "operator": "=",
        "value": "public_building"
      }
    ],
    "style": {
      "map-scales": [{
        "min": 0,
        "max": 100000,
        "symbols": [{
          "image": {
            "name": "building.svg",
            "scale": 1.0,
            "apply-gpkg-angle": false,
            "angle-offset": 0,
            "format": "image/svg+xml"
          }
        }]
      }]
    }
  },
  {
    "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AL013"
      },
      {
        "operand": "OSM_T_AME",
        "operator": "=",
        "value": "gas_station"
      }
    ],
    "style": {
      "map-scales": [{
```

```
            "min": 0,
            "max": 200000,
            "symbols": [{
                "fill": {
                  "color": "0x3d3d3f",
                  "opacity": 0.8,
                  "image": {
                    "name": "",
                    "scale": 0,
                    "apply-gpkg-angle": false,
                    "angle-offset": 0,
                    "format": ""
                  },
                  "diameter": 16
                }
              },
              {
                "stroke": {
                  "color": "0xFF",
                  "opacity": 1.0,
                  "background-color": "0xFFFFFF",
                  "background-opacity": 1.0,
                  "diameter": 16,
                  "width": 2,
                  "vertical-offset": 0,
                  "horizontal-offset": 0,
                  "dashes": [],
                  "cap": "",
                  "casing": {
                    "color": "0x0",
                    "opacity": 1.0,
                    "width": 0
                  }
                }
              }
            ]
          }]
        }
      }
    ]
  }
}

//TransportationGroundCrv
{
  "symbol-mapping": {
    "table": "TransportationGroundCrv",
    "geometry-type": "MULTILINESTRING"
  },
  "style-rule": {
    "conditions": [{
```

```
        "criteria": [{
            "operand": "F_CODE",
            "operator": "=",
            "value": "AP030"
          },
          {
            "operand": "RIN_ROI",
            "operator": "<",
            "value": 5
          },
          {
            "operand": "RIN_ROI",
            "operator": ">",
            "value": 2
          }
        ],
        "style": {
          "map-scales": [{
              "min": 2500000,
              "max": 25000000,
              "symbols": [{
                "stroke": {
                  "color": "0xcb171a",
                  "opacity": 1.0,
                  "background-color": "0xFFFFFF",
                  "background-opacity": 1.0,
                  "width": 0.9,
                  "vertical-offset": 0,
                  "horizontal-offset": 0,
                  "dashes": [],
                  "cap": "",
                  "casing": {
                    "color": "0x0",
                    "opacity": 1.0,
                    "width": 0.12
                  }
                }
              }]
            },
            {
              "min": 500000,
              "max": 2500000,
              "symbols": [{
                "stroke": {
                  "color": "0xcb171a",
                  "opacity": 1.0,
                  "background-color": "0xFFFFFF",
                  "background-opacity": 1.0,
                  "width": 1.14,
                  "vertical-offset": 0,
                  "horizontal-offset": 0,
```

```
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0x0",
              "opacity": 1.0,
              "width": 0.18
            }
          }
        }]
      },
      {
        "min": 100000,
        "max": 500000,
        "symbols": [{
          "stroke": {
            "color": "0xcb171a",
            "opacity": 1.0,
            "background-color": "0xFFFFFF",
            "background-opacity": 1.0,
            "width": 2.25,
            "vertical-offset": 0,
            "horizontal-offset": 0,
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0x0",
              "opacity": 1.0,
              "width": 0.375
            }
          }
        }]
      },
      {
        "min": 5000,
        "max": 100000,
        "symbols": [{
          "stroke": {
            "color": "0xcb171a",
            "opacity": 1.0,
            "background-color": "0xFFFFFF",
            "background-opacity": 1.0,
            "width": 4.665,
            "vertical-offset": 0,
            "horizontal-offset": 0,
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0x0",
              "opacity": 1.0,
              "width": 0.8225
            }
```

```
            }
          }]
        },
        {
          "min": 0,
          "max": 5000,
          "symbols": [{
            "stroke": {
              "color": "0xcb171a",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 8,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [],
              "cap": "",
              "casing": {
                "color": "0x0",
                "opacity": 1.0,
                "width": 2
              }
            }
          }]
        }
      ]
    }
  },
  {
    "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AP030"
      },
      {
        "operand": "RIN_ROI",
        "operator": "=",
        "value": 5
      }
    ],
    "style": {
      "map-scales": [{
          "min": 2500000,
          "max": 25000000,
          "symbols": [{
            "stroke": {
              "color": "0x0",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 0.65,
```

```
            "vertical-offset": 0,
            "horizontal-offset": 0,
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0xFFFFFF",
              "opacity": 1.0,
              "width": 0.09
            }
          }
        }]
      },
      {
        "min": 500000,
        "max": 2500000,
        "symbols": [{
          "stroke": {
            "color": "0x0",
            "opacity": 1.0,
            "background-color": "0xFFFFFF",
            "background-opacity": 1.0,
            "width": 0.885,
            "vertical-offset": 0,
            "horizontal-offset": 0,
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0xFFFFFF",
              "opacity": 1.0,
              "width": 0.135
            }
          }
        }]
      },
      {
        "min": 100000,
        "max": 500000,
        "symbols": [{
          "stroke": {
            "color": "0x0",
            "opacity": 1.0,
            "background-color": "0xFFFFFF",
            "background-opacity": 1.0,
            "width": 1.688,
            "vertical-offset": 0,
            "horizontal-offset": 0,
            "dashes": [],
            "cap": "",
            "casing": {
              "color": "0xFFFFFF",
              "opacity": 1.0,
```

```
                "width": 0.281
              }
            }
          }]
        },
        {
          "min": 5000,
          "max": 100000,
          "symbols": [{
            "stroke": {
              "color": "0x0",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 3.491,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [],
              "cap": "",
              "casing": {
                "color": "0xFFFFFF",
                "opacity": 1.0,
                "width": 0.617
              }
            }
          }]
        },
        {
          "min": 0,
          "max": 5000,
          "symbols": [{
            "stroke": {
              "color": "0x0",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 6,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [],
              "cap": "",
              "casing": {
                "color": "0xFFFFFF",
                "opacity": 1.0,
                "width": 1.5
              }
            }
          }]
        }
      ]
    }
```

```
    },
    {
      "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "AQ040"
        },
        {
          "operand": "TRS",
          "operator": "=",
          "value": "13"
        },
        {
          "operand": "viz-pass",
          "operator": "!=",
          "value": "1"
        }
      ],
      "style": {
        "map-scales": [{
            "min": 5000,
            "max": 100000,
            "symbols": [{
              "stroke": {
                "color": "0x0",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 12,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "square",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
                }
              }
            }]
          },
          {
            "min": 0,
            "max": 5000,
            "symbols": [{
              "stroke": {
                "color": "0x0",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 20,
```

```
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "square",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
                }
              }
            }]
          }
        ]
      }
    },
    {
      "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "AQ040"
        },
        {
          "operand": "TRS",
          "operator": "=",
          "value": "13"
        },
        {
          "operand": "viz-pass",
          "operator": "=",
          "value": "1"
        }
      ],
      "style": {
        "map-scales": [{
            "min": 5000,
            "max": 100000,
            "symbols": [{
              "stroke": {
                "color": "0xFFFFFF",
                "opacity": 1.0,
                "background-color": "0xFFFFFF",
                "background-opacity": 1.0,
                "width": 10,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [],
                "cap": "round",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0
```

```
              }
            }
          }]
        },
        {
          "min": 0,
          "max": 5000,
          "symbols": [{
            "stroke": {
              "color": "0xFFFFFF",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 14,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [],
              "cap": "round",
              "casing": {
                "color": "0x0",
                "opacity": 1.0,
                "width": 0
              }
            }
          }]
        }
      ]
    }
  },
  {
    "criteria": [{
      "operand": "F_CODE",
      "operator": "=",
      "value": "AP010"
    }],
    "style": {
      "map-scales": [{
          "min": 2500000,
          "max": 25000000,
          "symbols": [{
            "stroke": {
              "color": "0xFFFFFF",
              "opacity": 1.0,
              "background-color": "0xf7ab82",
              "background-opacity": 1.0,
              "width": 0.9,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [1.87, 4.124, 0.625, 1.375],
              "cap": "",
              "casing": {
```

```
            "color": "0x0",
            "opacity": 1.0,
            "width": 0.12
          }
        }
      }]
    },
    {
      "min": 500000,
      "max": 2500000,
      "symbols": [{
        "stroke": {
          "color": "0xFFFFFF",
          "opacity": 1.0,
          "background-color": "0xf7ab82",
          "background-opacity": 1.0,
          "width": 1.14,
          "vertical-offset": 0,
          "horizontal-offset": 0,
          "dashes": [2.415, 5.323, 0.807, 1.774],
          "cap": "",
          "casing": {
            "color": "0x0",
            "opacity": 1.0,
            "width": 0.18
          }
        }
      }]
    },
    {
      "min": 100000,
      "max": 500000,
      "symbols": [{
        "stroke": {
          "color": "0xFFFFFF",
          "opacity": 1.0,
          "background-color": "0xf7ab82",
          "background-opacity": 1.0,
          "width": 2.25,
          "vertical-offset": 0,
          "horizontal-offset": 0,
          "dashes": [4.797, 10.577, 1.603, 3.526],
          "cap": "",
          "casing": {
            "color": "0x0",
            "opacity": 1.0,
            "width": 0.375
          }
        }
      }]
    },
```

```
          {
            "min": 5000,
            "max": 100000,
            "symbols": [{
              "stroke": {
                "color": "0xFFFFFF",
                "opacity": 1.0,
                "background-color": "0xf7ab82",
                "background-opacity": 1.0,
                "width": 4.655,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [10, 22, 3.333, 7.333],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 0.8225
                }
              }
            }]
          },
          {
            "min": 0,
            "max": 5000,
            "symbols": [{
              "stroke": {
                "color": "0xFFFFFF",
                "opacity": 1.0,
                "background-color": "0xf7ab82",
                "background-opacity": 1.0,
                "width": 8,
                "vertical-offset": 0,
                "horizontal-offset": 0,
                "dashes": [18.1, 39.82, 6.033, 13.273],
                "cap": "",
                "casing": {
                  "color": "0x0",
                  "opacity": 1.0,
                  "width": 2
                }
              }
            }]
          }
        ]
      }
    },
    {
      "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
```

```
          "value": "AP050"
        }],
        "style": {
          "map-scales": [{
              "min": 5000,
              "max": 100000,
              "symbols": [{
                "stroke": {
                  "color": "0xFFFFFF",
                  "opacity": 1.0,
                  "background-color": "0x0",
                  "background-opacity": 1.0,
                  "width": 2.363,
                  "vertical-offset": 0,
                  "horizontal-offset": 0,
                  "dashes": [5, 9, 1.5, 2.5],
                  "cap": "",
                  "casing": {
                    "color": "0x0",
                    "opacity": 1.0,
                    "width": 0
                  }
                }
              }]
            },
            {
              "min": 0,
              "max": 5000,
              "symbols": [{
                "stroke": {
                  "color": "0xFFFFFF",
                  "opacity": 1.0,
                  "background-color": "0x0",
                  "background-opacity": 1.0,
                  "width": 4.5,
                  "vertical-offset": 0,
                  "horizontal-offset": 0,
                  "dashes": [10, 18, 3, 5],
                  "cap": "",
                  "casing": {
                    "color": "0x0",
                    "opacity": 1.0,
                    "width": 0
                  }
                }
              }]
            }
          ]
        }
      }
    ]
```

```
  }
}

//UtilityInfrastructureCrv
{
  "symbol-mapping": {
    "table": "UtilityInfrastructureCrv",
    "geometry-type": "MULTILINESTRING"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
        "operand": "F_CODE",
        "operator": "=",
        "value": "AT005"
      }],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 200000,
          "symbols": [{
            "stroke": {
              "color": "0x473895",
              "opacity": 1.0,
              "background-color": "0xFFFFFF",
              "background-opacity": 1.0,
              "width": 20,
              "vertical-offset": 0,
              "horizontal-offset": 0,
              "dashes": [],
              "cap": "",
              "casing": {
                "color": "0x0",
                "opacity": 1.0,
                "width": 0
              }
            }
          }]
        }]
      }
    }]
  }
}

//UtilityInfrastructurePnt
{
  "symbol-mapping": {
    "table": "UtilityInfrastructurePnt",
    "geometry-type": "POINT"
  },
  "style-rule": {
```

```
    "conditions": [{
       "criteria": [{
           "operand": "F_CODE",
           "operator": "=",
           "value": "AT042"
         },
         {
           "operand": "HGT",
           "operator": "<",
           "value": 46
         }
       ],
       "style": {
         "map-scales": [{
             "min": 0,
             "max": 200000,
             "symbols": [{
               "image": {
                 "name": "pylon_wilson.svg",
                 "scale": 5,
                 "apply-gpkg-angle": false,
                 "angle-offset": 0,
                 "format": "image/svg+xml"
               }
             }]
           },
           {
             "min": 0,
             "max": 35000,
             "symbols": [{
               "label": {
                 "text": "'Pylon'",
                 "color": "0x0",
                 "opacity": 1.0,
                 "font": "OpenSans",
                 "font-size": 12,
                 "outline-color": "0xFFFFFF",
                 "outline-opacity": 1.0,
                 "outline-width": 1,
                 "vertical-offset": -3,
                 "horizontal-offset": 0
               }
             }]
           }
         ]
       }
     },
     {
       "criteria": [{
           "operand": "F_CODE",
           "operator": "=",
```

```
            "value": "AT042"
          },
          {
            "operand": "HGT",
            "operator": ">=",
            "value": 46
          }
        ],
        "style": {
          "map-scales": [{
              "min": 0,
              "max": 200000,
              "symbols": [{
                "image": {
                  "name": "PT_Blue072_VO.svg",
                  "scale": 6,
                  "apply-gpkg-angle": false,
                  "angle-offset": 0,
                  "format": "image/svg+xml"
                }
              }]
            },
            {
              "min": 0,
              "max": 35000,
              "symbols": [{
                "label": {
                  "text": "'Pylon'",
                  "color": "0x0",
                  "opacity": 1.0,
                  "font": "OpenSans",
                  "font-size": 12,
                  "outline-color": "0xFFFFFF",
                  "outline-opacity": 1.0,
                  "outline-width": 1,
                  "vertical-offset": -3,
                  "horizontal-offset": 0
                }
              }]
            }
          ]
        }
      }
    ]
  }
}

//VegetationSrf
{
  "symbol-mapping": {
    "table": "VegetationSrf",
```

```
    "geometry-type": "MULTIPOLYGON"
  },
  "style-rule": {
    "conditions": [{
      "criteria": [{
          "operand": "F_CODE",
          "operator": "=",
          "value": "EC015"
        },
        {
          "operand": "OSM_T_LAN",
          "operator": "=",
          "value": "forest"
        }
      ],
      "style": {
        "map-scales": [{
          "min": 0,
          "max": 1000000,
          "symbols": [{
            "fill": {
              "color": "0xf7eded",
              "opacity": 1.0,
              "image": {
                "name": "",
                "scale": 0,
                "apply-gpkg-angle": false,
                "angle-offset": 0,
                "format": ""
              }
            }
          }]
        }]
      }
    }]
  }
}
```

Appendix F: Revision History

NOTE	*Example History (Delete this note).* replace below entries as needed

Table 24. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
Nov 5, 2018	Sara Saeedi	.1	all	initial version
Nov 21, 2018	Sara Saeedi	.9	all	comments integrate
Nov 28, 2018	Sara Saeedi	1.0	various	preparation for publication

Appendix G: Bibliography

1. Yutzler, J., Cass, R.: OGC Portrayal Concept Development Study. OGC 17-094r1,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-094r1.html (2018).
2. Field, K.: A Cacophony of Cartography. The Cartographic Journal. 51, 1–10 (2014).
3. Fellah, S.: OGC Testbed 11 Symbology Mediation Engineering Report. OGC 15-058,Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=64385 (2015).
4. Fellah, S.: OGC Testbed 11 Implementing Linked Data and Semantically Enabling OGC Services Engineering Report. OGC 15-054,Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=64405 (2015).
5. Fellah, S.: OGC Testbed 13 Portrayal Engineering Report. OGC 17-045,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-045.html (2018).
6. Bruce, C.: OGC OWS-6 Symbology Encoding (SE) Changes Engineering Report. OGC 09-016,Open Geospatial Consortium, https://portal.opengeospatial.org/files/?artifact_id=33515 (2009).