OGC Testbed-15

Encoding and Metadata Conceptual Model for Styles
Engineering Report
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Subject</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>Executive Summary</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.1. Document contributor contact points</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.2. Foreword</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>References</td>
<td>7</td>
</tr>
<tr>
<td>4.</td>
<td>Terms and definitions</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>4.1. Abbreviated terms</td>
<td>9</td>
</tr>
<tr>
<td>5.</td>
<td>Overview</td>
<td>10</td>
</tr>
<tr>
<td>6.</td>
<td>Metadata and encoding model</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>6.1. An overview of the model</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>6.2. The Style</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>6.3. The Style Metadata</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>6.4. Stylesheets</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>6.5. Styleable layer sets</td>
<td>16</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Style Metadata JSON Encoding</td>
<td>21</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Style examples</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>B.1. Generic vector style</td>
<td>23</td>
</tr>
<tr>
<td>Appendix C</td>
<td>Revision History</td>
<td>25</td>
</tr>
<tr>
<td>Appendix D</td>
<td>Bibliography</td>
<td>26</td>
</tr>
</tbody>
</table>
Publication Date: 2019-12-11

Approval Date: 2019-11-22

Submission Date: 2019-10-08

Reference number of this document: OGC 19-023r1

Reference URL for this document: http://www.opengis.net/doc/PER/t15-D011

Category: OGC Public Engineering Report

Editor: Andrea Aime

Title: OGC Testbed-15: Encoding and Metadata Conceptual Model for Styles Engineering Report

OGC Public Engineering Report

COPYRIGHT

Copyright © 2019 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

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

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

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

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

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

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

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

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.
Chapter 1. Subject

This OGC Testbed 15 Engineering Report (ER) describes a style encoding and metadata conceptual model that provides information for understanding styles intended usage, availability, compatibility with existing layers, and supporting style search. A style is a sequence of rules of symbolizing instructions to be applied by a rendering engine on one or more features and/or coverages.

The model also provides a way to express and locate multiple encodings for each style described. For example, the Styled Layer Descriptor (SLD) 1.0 [1], Symbology Encoding (SE) 1.1 [2], Cascading Style Sheets (CSS) [3], and Mapbox GL [4]) styles.

This document builds upon previous OGC work, in particular:

- The "OGC Symbology Conceptual Model: Core part" [5] candidate standard which defines common portrayal concepts shared across various style encodings.
- The OGC Vector Tiles Pilot [6] initiative that defined a prototype of a Styles API that is independent of the style encoding.
Chapter 2. Executive Summary

Styles in the OGC standards baseline have traditionally played two different roles:

- As opaque identifiers in the implementations of OGC Web Map Service (WMS) and Web Map Tile Service (WMTS) standards. These standards enable selection of different visual appearances for server side rendered map layers.

- As Extensible Markup Language (XML) based standards such as the OGC Styled Layer Descriptor/Symbology Encoding Standard (SLD/SE) that serve as an exchange format between systems. This approach enables a translation from a system's internal styling data structure to an external (interoperable) form, and back, perhaps into a different system.

Evolutions in web based mapping systems require more robust support for styling. In particular:

- The emerging client-side style capabilities in web mapping require a way to locate a suitable style on the server side, determine the style's applicability to the current displayed layers, and retrieve the style.

- The same environment often requires style editing.

- Style catalogs and style reuse require a way to describe styles (what kind of symbolization is used, what layers are involved, what attributes are needed).

- Both client and server applications are increasingly supporting a wider variety of open styling encodings. Multiple style encodings can be made available either through hand setup, or through automated conversion.

The conceptual model described in this ER focuses on supporting the above use cases, with an accent on simplicity and ease of implementation. As such, only the minimum information necessary to locate, reason about, and retrieve styles is included.

2.1. Document contributor contact points

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

Contacts

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrea Aime</td>
<td>GeoSolutions</td>
<td>Editor</td>
</tr>
<tr>
<td>Jeff Harrison</td>
<td>AGC</td>
<td>Contributor</td>
</tr>
<tr>
<td>Olivier Ertz</td>
<td>HEIG-VD</td>
<td>Contributor</td>
</tr>
</tbody>
</table>

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.
Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.
Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC API - Features - Part 1: Core, version 1.0.0-draft.2 [http://docs.opengeospatial.org/DRAFTS/17-069r2.html]
Chapter 4. Terms and definitions

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

**style**

a sequence of rules of symbolizing instructions to be applied by a rendering engine on one or more features and/or coverages

**style encoding**

specification to express a style as one or more files

**NOTE** In Testbed-15 Mapbox Styles, OGC SLD versions 1.0 and 1.1 are used.

**stylesheet**

representation of a style in a style encoding

**style metadata**

esential information about a style needed to support users to discover and select styles for rendering their data and for visual style editors to create user interfaces for editing a style

**layer**

A layer is an abstraction of reality specified by a geographic data model (feature, coverage…) and represented using a set of symbols (Style) to plot it. A layer contributes to a single geographic subject and may be a theme.

**Web API**

API using an architectural style that is founded on the technologies of the Web

**Features API**

OGC API Features provides API building blocks to create, modify and query features on the Web.

**Coverages API**

OGC API Coverages provides API building block to access coverages as defined by the Coverage Implementation Schema (CIS) 1.1 on the Web.

**Maps API**

OGC API Maps provides API building block to describe, build and retrieve web maps.

**Tiles API**

OGC API Tiles provides API building block to describe, build and retrieve tiles from any resource that can be subdivided in a regular set of tiles (e.g., maps, features and coverages)

**Styles API**

OGC API Styles is a Web API that enables map servers and clients as well as visual style editors to manage and fetch styles.
4.1. Abbreviated terms

API
  Application Programming Interface

OGC
  Open Geospatial Consortium

SLD
  OGC Styled Layer Descriptor

SE
  Symbology Encoding
Chapter 5. Overview

Section 6 introduces the style metadata and encoding model, and explains each element in detail.

Annex A provides a sample JSON implementation of the style metadata model.

Annex B provides includes samples styles.

Annex C contains the document revision history.

Annex D contains bibliographic references.
Chapter 6. Metadata and encoding model

6.1. An overview of the model

The conceptual model covering style metadata and multiple encodings is summarized in the following Unified Markup Language (UML) class diagram.

Figure 1. Style metadata and encoding conceptual model

The model can be succinctly divided in three main components:

- The Style is at the center of the model. Every other major component of the model is related to Style.
- The StyleMetadata, connected to the style, describes the Style with a simple metadata entry, as well as some structural information, such as which inner layers make up the style and what attributes are needed.
- Finally, the Stylesheet class describes the actual style encodings available.

Note that the organization of the model into these three components, makes it complementary with the Symbology Core draft specification (18-067r2). This organization minimizes overlap and duplication between the two models in order to facilitate future convergence between them.

### 6.2. The Style

A Style is a sequence of rules for symbolizing instructions that are to be applied by a rendering engine to one or more features and/or coverages.

On a layer by layer basis the WMS and WMTS standards list available style names in their capabilities document. This allows clients to identify the desired style using the STYLES parameter in a GetMap request. However, the WMS and WMTS standards do not offer support for a client to provide their own styles. The SLD standard addressed this limitation in two ways:

- By providing the styling rules through a URL passed to the server using the SLD parameter in a GetMap request, or
- By providing the styling rules to the server through an SLD document passed through the SLD_BODY parameter in a GetMap request.

Implementations including the SLD extensions for WMS also allow extracting the style encoded as SLD, and modifying the style (getStyle and PutStyle SLD extension for the WMS protocol).

Moreover, the association between layers and styles is modelled as a one-to-many containment, from the layer to the style.

Traditionally, web maps were often built by stacking multiple layers and choosing the desired style for each (layers and styles WMS parameters). However, it was also possible to provide full basemaps either by using the WMS capabilities tree structure (a layer containing other layers) or by simply leveraging the services "opaque" nature to hide a group of layers with a coordinated symbology under a single layer name.

In the new emerging OGC API specifications currently being developed [https://www.opengeospatial.org/blog/2996](https://www.opengeospatial.org/blog/2996) there is a desire to also support explicit knowledge of the associations between styles and data sources (collections). This is true whether the layers are vector or raster data, while also retaining the ability to keep on rendering fully opaque layers that the server can build using whatever approach is deemed most appropriate. This is in particular useful to support client side rendering, as well as client side style editing.

This scenario involves bidirectional associations between a style and collections:

- A collection from a Features, Coverages, Maps or Tiles API can link to one or more styles as a way to indicate a default styling that client side rendering engine can use.
- A style can link to one or more collections as examples of suitable data sources.

The relationship of the OGC API - Features standard to style usage is straightforward because a collection can be rendered client-side using styles that the API might provide, or a client might be looking for a suitable style for the layer on a shared style catalog. With the exception of simple
styles, for rendering to succeed attributes used by the style need to be present. Thus, knowing their name, expected data type, and semantics is important.

The relationship of the OGC API - Coverages draft specification to styles is similar to the Features one, with two exceptions:

- With the exception of simple geo-referenced images having Red-Green-Blue (RGB) colors (either explicitly, or through an embedded palette), coverage rendering can involve a number of operations on the raster bands, including band selection, contrast stretching, color map application and hill shading.
- Client-side rendering of raw raster data, while possible, has not yet reached widespread support as the vector case.

The relationship of the OGC API - Tiles draft specification to styles is more interesting, as it can involve both client and server-side rendering:

- If Tiles API is simply returning server side rendered tiles, then a link from the tiles collection to the style may be a way to describe how the rendering is done. This eventually allows a style editor to alter the style and submit it back to the server. See the Styles API also developed in Testbed 15 (OGC 19-010).
- If the Tiles API is also returning tiled vector data, such as multi-layer Mapbox vector tiles, then the link can be used by a client to setup a suitable default rendering for the tiled vector data in question.

At the time of writing there is no tile encoding standard supporting the transfer of both vector and raster information. However, that would not change the conceptual model. The style linked from the collection could simply contain symbolization directives for both data types.

An implementation of the OGC API - Maps draft specification would behave the same as an implementation of the Tiles API in a server side rendering scenario, and would allow a style editor to eventually modify the style used for rendering.

### 6.3. The Style Metadata

The Style metadata associated components represents most of the UML diagram real estate. The information represented in the model is actually designed to be simple and compact and most of the content is there to provide the possible values of a few enumerations.

At the core of the style metadata the following information is enumerated:

- **Id**: the style identifier, same as the Style class.
- **Title**: a title for the style, suitable for indicating the style usage in a style listing.
- **Abstract**: a longer description of the style, including all information needed to facilitate understanding, such as applicability, classification methods, reference to norms and whatever other information the user should be aware of before using the style.
- **Keywords**: a list of well-known keywords that would help locating the style in a catalog.
- **PointOfContact**: information that can be used to contact the authors or custodians for the style.
• Version: a version number for the style.
• Scope: with a fixed value of Style.
• StyleableLayerSet: indicating association with a well known group of layers to be styled (more on this later in this chapter).
• AccessConstraints: an indication about the availability of the style that the user with access to the style needs to be aware of before using or redistributing the style in question. Possible values are confidential, restricted, secret, topSecret, unclassified

The style metadata is then associated with a thumbnail, a link to a visual representation of a map rendered with the style (not to be confused with a legend) and suitable for presenting a carousel of styles in a client. An example taken from the OpenMapTiles [https://openmaptiles.org/] web site follows:

![Figure 2. A style carousel showing a number of style thumbnails](image)

A style can be associated with multiple dates, in particular:

• Creation: the timestamp when the style was first produced.
• Publication: the timestamp when the style was first made available to the users.
• Revision: the timestamp of the last style change/revision.
• ValidTill: the timestamp marking the future validity of the style (the style may no longer be applicable at this date, or that a new revision of the style is going to be issued).
• ReceivedOn: when the style was received from an external provider.

Finally, the style metadata provides indications of the layers involved in the symbolization (which might be just one), and the attributes used by the style for each layer (could be none). While part of this information could be retrieved from the stylesheets themselves, this is a succinct representation enumerating the minimum set of data the style needs in order to actually symbolize data.

In particular, for each layer the following information is provided:

• A layer id, typically the same identifier used in the style to refer to the layer
• A layer type, with the possible values of Point, Line, Polygon, Vector, Raster. Vector can be used as a generic way to refer to vector data when the precise geometry type is not known, or the
style is designed to work with multiple types (an example of such style is available in the Styles examples annex.

- A layer description, which can be used to understand the role of the layer in the style
- An eventual list of attributes that the style needs out of the layer in order to perform filtering, classification, symbolization

When present, each attribute holds the following information:

- **Id**: the name of the attribute used in the style (e.g., from an SLD `<PropertyName>` tag)
- **Description**: an optional textual description of the attribute, which a style editor can show to provide more semantics around the attribute usage
- **Type**: the attribute type, with possible values of `String`, `Numeric`, `Boolean`, `Date`, `DateTime`

The attribute ids can be derived via a simple automatic scan of the style contents. The type can be loosely inferred by the rendering machinery needs and context of where the attribute is used (e.g. an attribute used directly as a rotation needs to be a number). However, both a more precise type and a description also have to do with the semantics of what the style is depicting, thus, manual intervention will be needed to fill those fields with accuracy.

### 6.4. Stylesheets

Each style can be considered as an abstract set of instructions to render a particular map, or a portion of the map. The instructions can then be delivered in a particular encoding. In Testbed 15 SLD 1.0, SLD 1.1, Mapbox Styles were used along with other system specific language implementations like GeoServer GeoCSS and GNOSIS CMSS.

Each style language features different characteristics, styling abilities, and different intended target audience, for example:

- **SLDs** are based on XML and meant for machine interchange, though they can also be hand edited, and offer a wide variety of features adapting themselves for both tiled output and custom, dynamic, Map API like map making.
- **Mapbox Styles** are based on JSON and geared towards client-side rendering of vector tiles, and assume the "Web Mercator" CRS and the associated tile matrix. These characteristics make using the Mapbox Styles awkward to use in other scenarios.
- **GeoServer GeoCSS and GNOSIS CMSS** offer similar capabilities to SLDs with a more compact syntax and richer rule handling mechanics, gearing them towards hand editing of complex styles.

Some servers have developed the ability to automatically convert between formats. However, the different nature and possibilities of the languages often make for a lossy conversion, and it is sometimes possible to create a better matching output by hand editing the style instead.

Finally, styling languages are not static, but evolve over time, thus it might be necessary to indicate the version of the style language used in the encoding.

Given these considerations, a style metadata can link to one or more stylesheets, that is, encoding of
a style, with the following attributes:

- **Title**: the title of the encoding language, e.g. Mapbox Style, SLD, GeoCSS or CMSS.
- **Version**: the version of the encoding language used, e.g., 1.0
- **Specification**: a link to the style encoding specification, e.g. https://docs.mapbox.com/mapbox-gl-js/style-spec/
- **Native**: true or false, indicating if this is the native encoding of the style, possibly hand prepared, or an on-the-fly conversion, with potential loss of details in the process.
- **Link**: a JSON link to the stylesheet, with the usual URL, media type, and a relation of stylesheet.

### 6.5. Styleable layer sets

A "Styleable layer set" is a group of layers that are typically styled together, as a group, typically to form a basemap.

Common examples of these sets are OpenStreetMap [http://www.openstreetmap.org], or Natural Earth [https://www.naturalearthdata.com].

These data sets can be styled in different ways. For example in Testbed 15 the various actors worked with a base map style, a night variant that map, and a "raster backdrop" version of the same basemap. The different conditions of light (day vs night) and the differences in background color call for a different rendering of the overlayed elements (roads, buildings, points of interest).

The metadata model currently identifies a styleable layer set by the "styleableLayerSet" string attribute in the style metadata. Two styles sharing the same value for said attributes are meant to be alternatives of each other, based on same set of layers. The night and day cases use the exact same set of layers.

![Figure 3. Day view, with solid background (vector data rendered client side by MapStore)](image)
Figure 4. Night view, with solid background (vector data rendered client side by MapStore)

The overlay one adds a digital elevation model image as a backdrop.

Figure 5. Day view with hillshade background, zoomed out (vector data rendered client side by Mapstore, raster hill shade rendered server side by GeoServer)
Examples of the same concept can be found online. For example, both OpenMapTiles and MapBox render their respective sets of layers with different styles depending on the necessities of the customer. Below are some visual examples:

Please note that the same set of layers and the same style name can come with significantly different representations from different vendors, and while being based on similar data sets, the amount of information in each can vary significantly (and quite likely, the actual sets of layers and attributes available in each changes with vendor implementations).

Here are examples from the same area in Chicago, rendered using an "OpenStreetMap" data set, using a style called "OSM Bright", from three different mapping websites (one which is OpenStreetMap proper):
With this in mind, it is best to consider the styleable layer set as an indication provided by a single Styles API to label variants of the same style, on the same dataset, to be used under different conditions.

A more general and interoperable notion of a styleable layer could be built by advertising the set of layers and available attributes. This information could use the same structure as the style metadata, but with an important semantic difference. While the style metadata reports the layers and attributes the style needs to operate on, a styleable layer set would instead report all the attributes available to eventual styles for rendering purposes.
Figure 11. Possible description of a styleable layer set

Such an explicit description would allow for sharing styles across servers and catalogs providing the same styleable layer set.
Appendix A: Style Metadata JSON Encoding

The following is an example JSON document implementing the style metadata, as well providing links to the various style encodings.

**Style metadata example**

```
{
  "id": "night",
  "title": "Topographic night style",
  "description": "This topographic basemap style is designed to be used in situations with low ambient light. The style supports datasets based on the TDS 6.1 specification.",
  "keywords": [
    "basemap",
    "TDS",
    "TDS 6.1",
    "OGC API"
  ],
  "pointOfContact": "John Doe",
  "accessConstraints": "unclassified",
  "dates": {
    "creation": "2019-01-01T10:05:00Z",
    "publication": "2019-01-01T11:05:00Z",
    "revision": "2019-02-01T11:05:00Z",
    "validTill": "2019-02-01T11:05:00Z",
    "receivedOn": "2019-02-01T11:05:00Z"
  },
  "scope": "style",
  "version": "1.0.0",
  "stylesheets": [
    {
      "title": "Mapbox Style",
      "version": "8",
      "specification": "https://docs.mapbox.com/mapbox-gl-js/style-spec/",
      "native": true,
      "tilingScheme": "GoogleMapsCompatible",
      "link": {
        "href": "https://example.org/catalog/1.0/styles/night?f=mapbox",
        "rel": "stylesheet",
        "type": "application/vnd.mapbox.style+json"
      }
    },
    {
      "title": "OGC SLD",
      "version": "1.0",
      "native": false,
      "link": {
        "href": "https://example.org/catalog/1.0/styles/night?f=sld10",
        "rel": "stylesheet"
      }
    }
  ]
}
```
"type": "application/vnd.ogc.sld+xml;version=1.0"
}
]
"layers": [
{
"id": "vegetationsrf",
"type": "polygon",
"sampleData": {
"href": "https://services.interactive-instruments.de/vtp/daraa/collections/vegetationsrf/items?f=json&limit=100",
"rel": "data",
"type": "application/geo+json"
}
},
{
"id": "hydrographycrv",
"type": "line",
"sampleData": {
"href": "https://services.interactive-instruments.de/vtp/daraa/collections/hydrographycrv/items?f=json&limit=100",
"rel": "data",
"type": "application/geo+json"
},
"attributes": [
{
"id": "f_code",
"type": "string"
}
]
},
"links": [
{
"rel": "preview",
"type": "image/png",
"href": "https://services.interactive-instruments.de/t15/daraa/resources/night-thumbnail.png"
}
]
Appendix B: Style examples

This annex contains example of styles mentioned in the description.

B.1. Generic vector style

The following style uses a custom function called dimensions to symbolize all types of geometry found, making this a good example of a style using the generic type Vector in the style layer metadata. GeoServer uses this style by default when the geometry type is unknown due to lack of information, or when the geometry is actually generic and different features can contain different geometric types.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0"
  xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
  xmlns="http://www.opengis.net/sld"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <Name>generic</Name>
    <UserStyle>
      <Title>Generic</Title>
      <Abstract>Generic style</Abstract>
      <FeatureTypeStyle>
        <Rule>
          <Name>Polygon</Name>
          <Title>Polygon</Title>
          <ogc:Filter>
            <ogc:PropertyIsEqualTo>
              <ogc:Function name="dimension">
                <ogc:Function name="geometry"/>
              </ogc:Function>
              <ogc:Literal>2</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#AAAAAA</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#000000</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
        <Rule>
          <Name>Line</Name>
          <Title>Line</Title>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```
## Appendix C: Revision History

### Table 1. Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Editor</th>
<th>Release</th>
<th>Primary clauses modified</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 26, 2019</td>
<td>A. Aime</td>
<td>0.1</td>
<td>all</td>
<td>initial version</td>
</tr>
<tr>
<td>May 24, 2019</td>
<td>A. Aime</td>
<td>0.2</td>
<td>all</td>
<td>Added references, diagrams, sample JSON encoding</td>
</tr>
<tr>
<td>Aug 14, 2019</td>
<td>A. Aime</td>
<td>0.3</td>
<td>all</td>
<td>Added descriptive text in the various sections</td>
</tr>
<tr>
<td>September 20, 2019</td>
<td>A. Aime</td>
<td>0.4</td>
<td>all</td>
<td>Applying reviewer’s feedback</td>
</tr>
<tr>
<td>October 2, 2019</td>
<td>A. Aime</td>
<td>0.5</td>
<td>all</td>
<td>Applying reviewer’s feedback</td>
</tr>
<tr>
<td>December 4, 2019</td>
<td>C. Reed</td>
<td>.6</td>
<td>Subject and Exec Summary</td>
<td>Minor edits for publication.</td>
</tr>
</tbody>
</table>
Appendix D: Bibliography


