OGC® DOCUMENT: 18-067R3

External identifier of this OGC® document: http://www.opengis.net/doc/IS/SymCore/1.0

OGC®
Making location count.

OGC SYMBOLOGY CONCEPTUAL MODEL: CORE PART

STANDARD
Conceptual model & encoding

APPROVED

Version: 1.0
Submission Date: 2018-09-07
Approval Date: 2020-08-24
Publication Date: 2020-10-15
Editor: Erwan Bocher, Olivier Ertz

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.

Suggested additions, changes and comments on this standard are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

Copyright notice

Note

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.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This document presents the requirements for defining the Symbology Conceptual Core Model (SymCore), the conceptual basis to define symbology rules for the portrayal of geographical data. It is modular and extensible (one core model, many extensions), also encoding agnostic (one symbology model, many encodings). It contains a minimal set of abstract classes representing explicit extension points of the model.

Note, that this document does not define any extensions.

The SWG work that led to this proposal has been done in the continuation of the Symbology Encoding standard (SE 1.1). While portrayal concerns the complete picture of what can be called a “cartographic ecosystem,” the description of symbology rules rather concerns the subpart of it about the instructions to be applied by a rendering engine to symbolize geodata. That’s why this proposal has a focus on symbology versus concerns close to WMS Style Layer Descriptor profile (SLD) considerations. In other words, while a set of “layer styles” describe the links between some geodata and some styles to build a map, each of these styles are built of a set of symbology rules in accordance with SymCore. The overall motivation that lead to this proposal is related to the issue “how to make richer the symbology abilities.” The first answer is modularity which comes with extensibility. SE 1.1 is not modular per se, while this proposal is designed to be so with a core model extensible to host the diversity of such abilities in relation to various data models. It means that the core model is somewhat abstract and does not define concrete visualizations (e.g., red dashed line of wide thickness to draw features of type cable car). The second answer that follows from the first is about extensions. As soon as the conceptual basis is set out in a specification document (this document), then extensions have to document the concrete symbology concepts to portray geodata structured according to a given data model. It is worth to notice that conceptually, the core model is not related to any specific underlying data model to be represented. It is up to an extension to define styling abilities in relation to a specific data model (e.g., FeatureTypeStyle). The third answer is about encodings. As soon as the conceptual basis is established and extensions packaged, then the task is to define encodings to formalize the concrete conceptual symbology abilities. In summary, SymCore concerns the first answer with a consistent approach to:

- provide the flexibility required to achieve adequate symbology rules for a variety of information communities; e.g., aviation symbols, weather symbols, thematic maps, etc,; and

- achieve high-level styling interoperability without encoding dependencies.

As a consequence, this document follows the same motivation that split up SLD 1.0 (SLD 1.1 and SE 1.1). This document puts together parts that are not specific to any service (e.g., Web Map Service), are independent, and allow the concepts to be reused by other standards willing to address aspects related to cartography. So a more general and portable symbology model is defined for use across the broad OGC standards baseline, to be applied to geospatial datasets as well as online geospatial data and mapping services. Potential implementations of SymCore are expected to enhance OGC standards such as the Web Map Service, Web Feature Service, GeoPackage, and others. By sharing a common core, and using an extension mechanism,

integration of these standards for the purposes of cartographic representation could be greatly simplified.

ii

KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, style, symbology, conceptual, core, modular, neutral, portrayal

iii SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- HEIG-VD (School of Management and Engineering Vaud)
- CNRS (National Center for Scientific Research)
- Strategic ACI
- IGN

iv SUBMITTERS

All questions regarding this submission should be directed to the editor or the submitters:

NAME	AFFILIATION
Olivier Ertz	HEIG-VD (School of Management and Engineering Vaud) https://heig-vd.ch
Erwan Bocher	CNRS (National Center for Scientific Research) http://www.cnrs.fr
Matt Sorenson	Strategic ACI https://www.strategicaci.com/
Dimitri Sarafinof	IGN http://www.ign.fr/

v OTHER CONTRIBUTORS

Comments on the content of this standard were given both informally and formally by members of the OGC. The OGC OAB (OGC Architecture Board) reviewed this document and provided comments. Written comments were submitted by the following individual members during the public review.

NAME	AFFILIATION
Carl Reed	Carl Reed and Associates

NAME	AFFILIATION
Jeff Yutzler	Image Matters LLC
Chris Little	Met Office
Antony Cooper	University of Pretoria
Sean Eagles	Canada Centre for Mapping and Earth Observation
Patty Zhao	Canada Centre for Mapping and Earth Observation
Jan Hjelmager	Specialist consultant
Davis, Ethan	University Corporation for Atmospheric Research (UCAR)

vi

SECURITY CONSIDERATIONS

No security considerations have been made for this standard.

1

SCOPE

1 SCOPE

This document presents the requirements that define the Symbology Conceptual Core Model (SymCore). The SymCore is a conceptual, modular, neutral model for the portrayal of geographical data. The model contains a minimal set of abstract classes representing explicit extension points of the model. Note: this document does not define any extensions.

Even though the SymCore could be extended in many places (such as Color or ParameterValue), the abstract Style class must be considered as the root element for portraying a geographic data model and for defining a concrete style model, e.g., specific style dedicated to render 2D vector data or a style for 3D, Virtual Reality, or Augmented Reality topics (Figure 1).

The SymCore is a new approach (Bocher E., Ertz O., 2018, see Annex B):

- to provide the flexibility required to achieve adequate cartographic styling and fill the needs of a variety of information communities; e.g., aviation symbols, weather symbols, thematic maps, etc.; and
- to achieve high level styling interoperability without encoding dependencies.

Figure 1 — The core model and its potential extensions

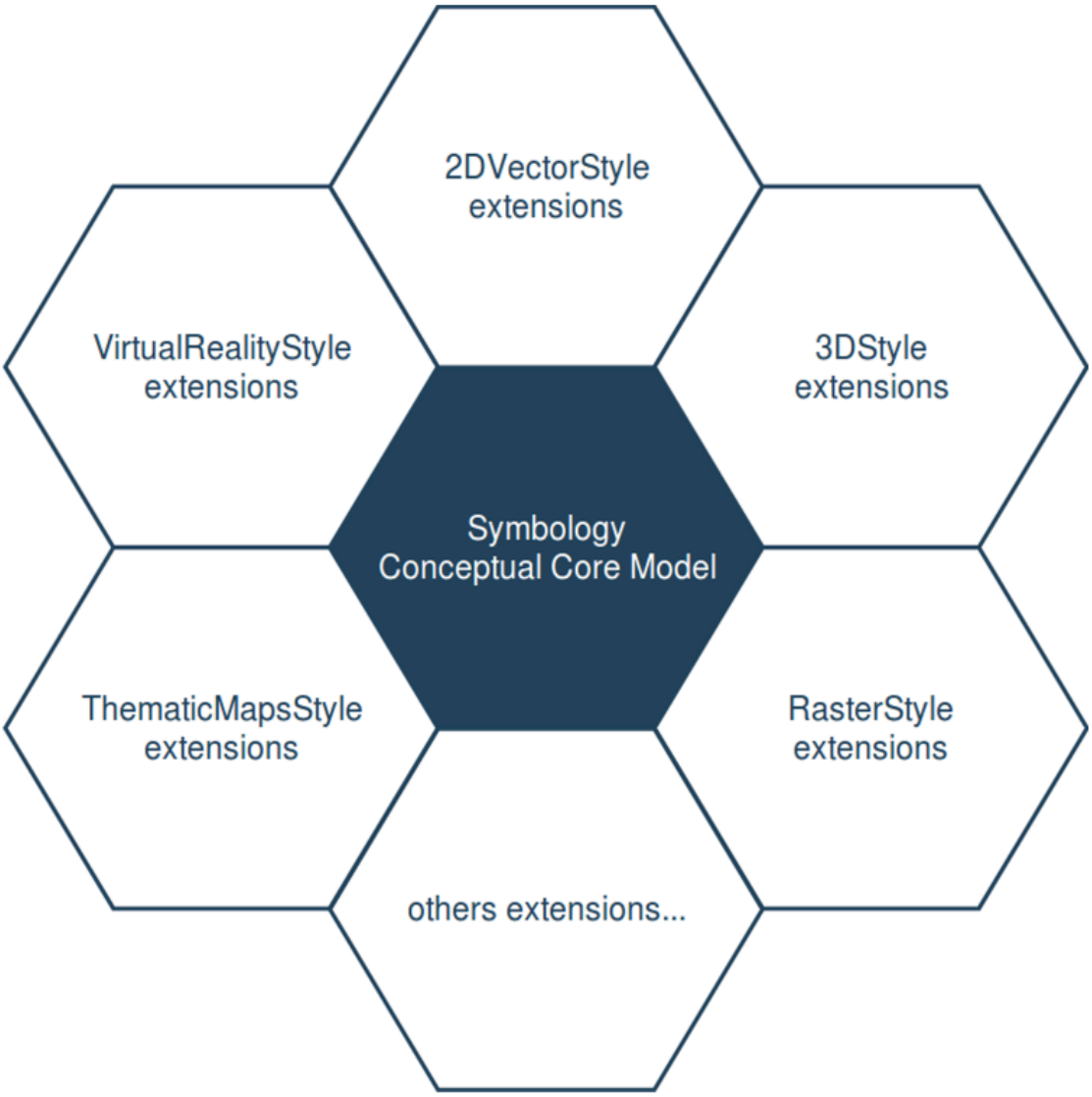
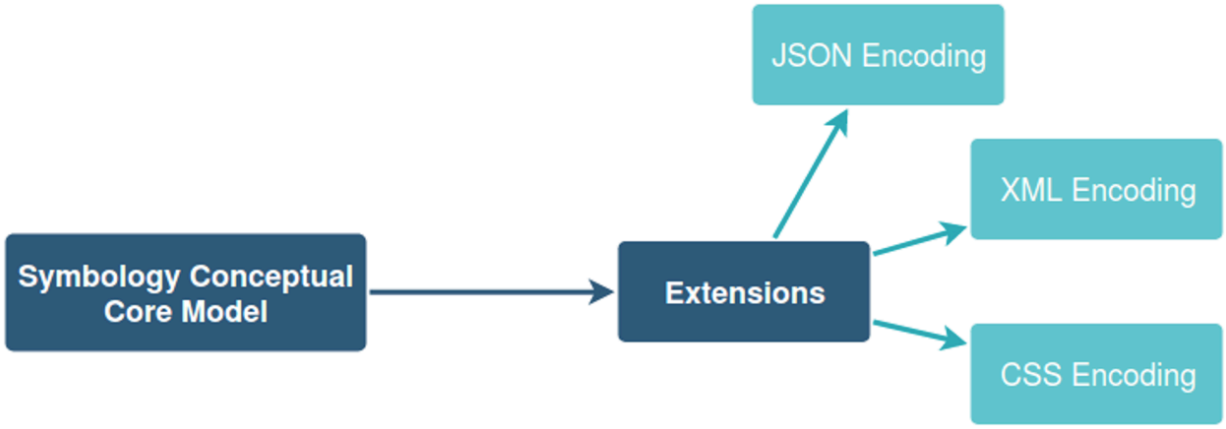


Figure 2 explains the relation between the core, the potential extensions, and their encodings. A community style extension must be based on a core element and will be encoding-independent. An extension should have a concrete encoding. As the figure shows encoding could be implemented in various formats.

Figure 2 — From core and extensions to encodings: principles of implementation



2

CONFORMANCE

2 CONFORMANCE

This document defines a standardization target for encodings that implement the OGC Symbology Conceptual Core Model. The goal is to allow different encodings to have equivalent content and semantics so that they can be interoperable. This document establishes a core requirements class with a URI of http://www.opengis.net/spec/symbology/2.0/req/core.

Requirements and conformance test URIs defined in this document are relative to http://www.opengis.net/spec/symbology/2.0. All requirements in this standard are part of the core requirement class stated above.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site[1].

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

[1] www.opengeospatial.org/cite

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Jeff de La Beaujardiere: OGC 06-042, *OpenGIS Web Map Service (WMS) Implementation Specification*. Open Geospatial Consortium (2006). https://portal.opengeospatial.org/files/?artifact_id=14416

Markus Lupp: OGC 05-078r4, *OpenGIS Styled Layer Descriptor Profile of the Web Map Service Implementation Specification*. Open Geospatial Consortium (2007). https://portal.opengeospatial.org/files/?artifact_id=22364

Dr. Markus Mueller: OGC 05-077r4, *OpenGIS Symbology Encoding Implementation Specification*. Open Geospatial Consortium (2007). https://portal.opengeospatial.org/files/?artifact_id=16700

Panagiotis (Peter) A. Vretanos: OGC 09-026r2, *OGC Filter Encoding 2.0 Encoding Standard – With Corrigendum* . Open Geospatial Consortium (2014). http://docs.opengeospatial.org/is/09-026r2/09-026r2.html

Policy SWG: OGC 08-131r3, *The Specification Model – Standard for Modular specifications*. Open Geospatial Consortium (2009). https://portal.opengeospatial.org/files/?artifact_id=34762&version=2

A. Phillips, M. Davis: IETF RFC 4646, *Tags for Identifying Languages*. Internet Engineering Task Force, Fremont, CA (2006-09). https://xml2rfc.tools.ietf.org/public/rfc/bibxml/reference.RFC.4646.xml

ISO: ISO 19117:2012, *Geographic information – Portrayal*. International Organization for Standardization, Geneva (2012). https://www.iso.org/standard/46226.html

The Unified Code for Units of Measure (UCUM), 2017 http://unitsofmeasure.org/ucum.html

W3C CSS Fonts chapter, 2016 https://www.w3.org/TR/CSS2/fonts.html#font-styling

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

For the purposes of this document, the following terms and definitions apply.

This document used the terms defined in Policy Directive 49[2], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

4.1. Portrayal

Presentation of information to humans (Note 1 to entry: Within the scope of this International Standard, portrayal is restricted to the portrayal of geographic information). [SOURCE: ISO 19117:2012, 4.20]

4.2. Layer

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

4.3. Style

Set of rules of symbolizing instructions to be applied by a rendering engine on a layer of geographic features (e.g., feature, coverage…).

[2] https://portal.ogc.org/public_ogc/directives/directives.php

4.4. Rendering engine

Automated process that produces graphics using a pipeline of layers and styles as inputs.

4.5. Render

Conversion of digital graphics data into visual form (EXAMPLE Generation of an image on a video display) [SOURCE: ISO 19117:2012, 4.27]

5

CONVENTIONS

5 CONVENTIONS

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. ABBREVIATED TERMS

The abbreviated terms clause gives a list of the abbreviated terms necessary for understanding this document.

Table 1 — Abbreviations

IETF	Internet Engineering Task Force, https://ietf.org/
ISO	International Organization for Standardization, https://www.iso.org
OGC	Open Geospatial Consortium, www.opengeospatial.org
UML	Unified Modeling Language, http://www.uml.org/

6

CORE CONCEPTUAL MODEL

6 CORE CONCEPTUAL MODEL

6.1. OVERVIEW

The requirements described in this section define the symbology core requirement class. The UML diagram (Figure 3) shows the fundamental concepts of the Symbology Conceptual Core Model.

Figure 3 — UML Class Diagram of the Symbology core

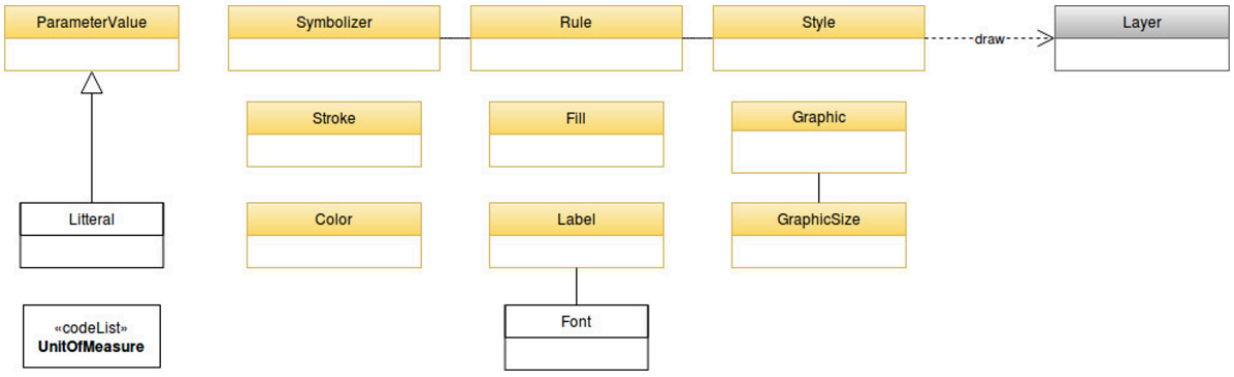


Table 2 — Requirements Class

REQUIREMENTS CLASS CORE	
http://www.opengis.net/spec/symbology/2.0/req/core	
Target type	Token
Dependencies	none
REQ 1	http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass Implementations shall support the encoding of all properties of the StyleClass and meet all of the tabulated constraints and notes.
REQ 2	http://www.opengis.net/spec/symbology/2.0/req/core/RuleClass Implementations shall support the encoding of all RuleClass properties and meet all of the tabulated constraints and notes.
REQ 3	http://www.opengis.net/spec/symbology/2.0/req/core/SymbolizerClass Implementations shall support the encoding of all SymbolizerClass properties and meet all of the tabulated constraints and notes.
REQ 4	http://www.opengis.net/spec/symbology/2.0/req/core/ParameterValueClassI Implementations shall support the encoding of all ParameterValue parameters class and meet all of the tabulated constraints and notes.

REQUIREMENTS CLASS CORE	
REQ 5	http://www.opengis.net/spec/symbology/2.0/req/core/LiteralClass Implementations shall support the encoding of all parameters of the LiteralClass and meet all of the tabulated constraints and notes.
REQ 6	http://www.opengis.net/spec/symbology/2.0/req/core/UOMClass Implementations shall support the encoding of all properties of the UOMClass and meet all of the tabulated constraints and notes.
REQ 7	http://www.opengis.net/spec/symbology/2.0/req/core/ColorClass Implementations shall support the encoding of all properties of the ColorClass and meet all of the tabulated constraints and notes.
REQ 8	http://www.opengis.net/spec/symbology/2.0/req/core/FillClass Implementations shall support the encoding of all properties of the FillClass and meet all of the tabulated constraints and notes.
REQ 9	http://www.opengis.net/spec/symbology/2.0/req/core/StrokeClass Implementations shall support the encoding of all properties of the StrokeClass and meet all of the tabulated constraints and notes.
REQ 10	http://www.opengis.net/spec/symbology/2.0/req/core/GraphicClass Implementations shall support the encoding of all properties of the GraphicClass and meet all of the tabulated constraints and note.
REQ 11	http://www.opengis.net/spec/symbology/2.0/req/core/GraphicSizeClass Implementations shall support the encoding of all properties of the GraphicSizeClass and meet all of the tabulated constraints and notes.
REQ 12	http://www.opengis.net/spec/symbology/2.0/req/core/LabelClass Implementations shall support the encoding of all properties of the LabelClass and meet all of the tabulated constraints and notes.
REQ 13	http://www.opengis.net/spec/symbology/2.0/req/core/FontClass Implementations shall support the encoding of all properties of the FontClass and meet all of the tabulated constraints and notes.

Each concept in the model can be extended according to two main extension principles:

- either globally related to an abstract class (in yellow); and
- or locally related to the extension property defined within each class of the model.

The role of each class and property in the model above is described in the tables below.

6.2. CLASS STYLE

Req 1 Implementations shall support the encoding of all properties of the StyleClass and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass

This class is the root concept of the Symbology Conceptual Core Model. This class organizes the rules of symbolizing instructions to be applied by a rendering engine on a layer of geographic features (e.g., vector based spatial data or raster data). As an abstract class, it is designed to be extended (e.g., the FeatureTypeStyle extension for vector data).

Please note that the graphic pipeline of the rendering engine must be expressed unambiguously for each concrete implementation of a Style in order to enable cartographic portrayal interoperability.

The StyleClass properties are documented in the following table.

Table 3 — StyleClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Style	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One
abstract	Human readable description	ParameterValue data type	Zero or one
rule	Rule(s) that drive(s) the rendering engine	Rule	One or more
extension	Any encoding should allow the user to extend the class to include custom items	Any	Zero or more

6.3. CLASS RULE

Req 2 Implementations shall support the encoding of all RuleClass properties and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/RuleClass

This core class describes the concept of a rule in the Symbology model. Rules are used to organize symbolizing instructions and potentially to define conditions of application of these associated symbolizers (e.g., feature-property conditions or map scales).

The RuleClass properties are documented in the following table.

Table 4 — RuleClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Rule	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
abstract	Human readable description	ParameterValue data type	Zero or one
symbolizer	Symbolize(s) to apply by the rendering engine	Symbolizer	One or more
extension	Any encoding should allow the user to extend the class to include custom properties	Any	Zero or more

6.4. CLASS SYMBOLIZER

> **Req 3** Implementations shall support the encoding of all SymbolizerClass properties and meet all of the tabulated constraints and notes.
> http://www.opengis.net/spec/symbology/2.0/req/core/SymbolizerClass

This class describes how to portray geographic data given a shape (e.g., area fill, line stroke, point marker, etc.) and graphical properties (e.g., color, opacity, font-family, etc.). As an abstract class, it is designed to be extended.

The SymbolizerClass properties are documented in the following table.

Table 5 — SymbolizerClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Symbolizer	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One
abstract	Human readable description	ParameterValue data type	Zero or one
uom	Unit of measure to apply to all graphical properties of a Symbolizer	uom code	Zero or one
extension	Any encoding should allow the user to extend the class to include custom items	Any	Zero or more

To understand what the symbolizer concept is, consider a “Lake” feature type represented by a Polygon that is to be symbolized as a “blue” filled polygon with its boundary drawn as a “black” line. As symbolizer is an abstract class: a concrete extension called here, for example, AreaSymbolizer, which must be provided to render an interior “fill” and an outlining “stroke.” Consequently, the AreaSymbolizer extension will implement concrete extensions of the abstract Stroke and Fill classes of the conceptual model.

Depending on the type of geographical object, a set of symbolizer extensions can be conceived. For example a LineSymbolizer to draw a river, a PointSymbolizer to represent the “Hospitals,” or a LabelSymbolizer to render the road name along a line.

6.5. CLASS PARAMETERVALUE

> **Req 4** Implementations shall support the encoding of all ParameterValue parameters class and meet all of the tabulated constraints and notes.
> http://www.opengis.net/spec/symbology/2.0/req/core/ParameterValueClass

The ParameterValue class represents a gateway that provides the value to be used by a parameter in a styling context of use (almost all styling parameters such as width, opacity, displacement, etc. are “parameter-values”). This class has a similar meaning to Expression as defined in the OGC Filter Encoding 2.0 standard. As an abstract class, it is designed to be extended (e.g., Literal).

The ParameterValue properties are documented in the following table.

Table 6 — ParameterValue

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
language	Language identifier for the ParameterValue element. (a)	Character String. This language identifier shall be as specified in IETF RFC 4646.	zero or more
extension	Any encoding should allow the ability to extend the class to include custom items	Any	zero or more

(a) The language identifier should offer a way to adapt the ParameterValue to a specified language, e.g., display the title of a Rule element both in English and French.

6.6. CLASS LITERAL

> **Req 5** Implementations shall support the encoding of all parameters of the LiteralClass and meet all of the tabulated constraints and notes.
> http://www.opengis.net/spec/symbology/2.0/req/core/LiteralClass

The Literal class is a concrete implementation of the ParameterValue class. LiteralClass represents a typed atomic literal value as a constant explicitly specified. It was originally defined in the OGC Filter Encoding 2.0 standard section 7.5.1.

LiteralClass properties are documented in the following table.

Table 7 — LiteralClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
value	A value for the literal data	Any	one

6.7. CLASS UOM CODELIST

> **Req 6** Implementations shall support the encoding of all properties of the UOMClass and meet all of the tabulated constraints and notes.
> http://www.opengis.net/spec/symbology/2.0/req/core/UOMClass

For styling parameters that define sizing and positioning of graphical objects (width, displacement, etc.) the unit of measure needs to be provided for the rendering engine. Therefore, for different levels of elements (e.g., Symbolizer, Stroke, Fill, GraphicSize, etc.) the model allows using different *uom* codes. Consequently, either the unit of measure is determined through the *uom* code directly associated to each element or it is determined by the innermost parent *uom* code (e.g., an *uom* code defined at the Symbolizer level implies that this unit is applied for all sizing and positioning values inside the Symbolizer).

Below is the list of allowed units of measure as per UCUM (except for pixel): * portrayal units: pixel, millimeter, inch, percentage; and * ground units: meter, foot.

The portrayal unit “pixel” is the default unit of measure. If available, the pixel size depends on the viewer client resolution, otherwise it is equal to 0.28mm * 0.28mm (~ 90 DPI).

6.8. CLASS COLOR

> **Req 7** Implementations shall support the encoding of all properties of the ColorClass and meet all of the tabulated constraints and notes.
> http://www.opengis.net/spec/symbology/2.0/req/core/ColorClass

The *ColorClass* allows the definition of color. As an abstract class and part of the base of the core graphical concepts, this class is a global point of extension for specifying concrete definitions of colors (e.g., RGBColor extension).

The ColorClass properties are documented in the following table.

Table 8 — ColorClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
extension	Any encoding should allow the extension of *ColorClass* with custom items	Any type	zero or more

6.9. CLASS FILL

Req 8 Implementations shall support the encoding of all properties of the *FillClass* and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/FillClass

FillClass defines the graphical symbolizing parameters required to draw the filling of a two-dimensional shape such as a polygon. As an abstract class and part of the base of the core graphical concepts, FillClass is a global point of extension for specifying concrete definitions for shape fill operations (e.g., the *SolidFill* and *GraphicFill* extensions).

The *FillClass* properties are documented in the following table.

Table 9 — FillClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties within a Fill	uom code	zero or one
extension	Any encoding should allow the extension of a Fill operation with custom items	Any type	zero or more

6.10. CLASS STROKE

Req 9 Implementations shall support the encoding of all properties of the StrokeClass and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/StrokeClass

StrokeClass defines the graphical symbolizing parameters for drawing an outline (e.g., for linear geometries or the exterior of a polygon geometry). As an abstract class and part of the base of the core graphical concepts, *StrokeClass* is a global point of extension to specify concrete ways to draw outlines (e.g., the PenStroke and GraphicStroke extensions). The StrokeClass properties are documented in the following table.

Table 10 — StrokeClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties inside a Stroke	uom code	zero or one
extension	Any encoding should allow to extend a Stroke with custom items	Any type	zero or more

6.11. CLASS GRAPHIC

Req 10 Implementations shall support the encoding of all properties of the *GraphicClass* and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/GraphicClass

The Graphic class defines the parameters for drawing a graphic symbol such as shape, color(s), and size. A *graphic* can be informally defined as "a little picture" and can be either a bitmap or scaled vector (the term "graphic" is used instead of the term "symbol" to avoid confusion with Symbolizer, which is used in a different context in this model). As an abstract class and part of the base of the core graphical concepts, *GraphicClass* is a global point of extension to specify concrete ways to draw "graphic symbol" (e.g., ExternalGraphic and MarkGraphic extensions).

The GraphicClass properties are documented in the following table.

Table 11 — GraphicClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties within a Graphic	uom code	zero or one
graphicSize	Rendering size of the graphic	GraphicSize data type	zero or one
extension	Any encoding should allow to extend a Graphic with custom items	Any type	zero or more

6.12. CLASS GRAPHICSIZE

Req 11 Implementations shall support the encoding of all properties of the *GraphicSizeClass* and meet all of the tabulated constraints and notes.

http://www.opengis.net/spec/symbology/2.0/req/core/GraphicSizeClass

The *GraphicSize* class determines the size of the graphic when it is rendered. As an abstract class, it is designed to be extended to support the various ways the size could be specified such as by a single value, a rectangular box, or by a three-dimensional cube. The GraphicSize properties are documented in the following table.

Table 12 — GraphicSize

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
extension	Any encoding should allow to extend a GraphicSize with custom items	Any type	zero or more

6.13. CLASS LABEL

Req 12 Implementations shall support the encoding of all properties of the LabelClass and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/LabelClass

LabelClass defines the graphical symbolizing properties for drawing a text label. As an abstract class and part of the base of the core graphical concepts, LabelClass is a point of extension to specify concrete ways to draw text label according to placement behaviors (e.g., a PointLabel or LineLabel).

LabelClass properties are documented in the following table.

Table 13 — LabelClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to the affected graphical properties within a Label	uom code	zero or one
labelText	Text-label content to draw	ParameterValue data type String	one
font	Font definition to draw the text-label content	Font data type Default value: system-dependent	zero or one
fill	Filling style to draw the glyphs	Fill data type	zero or one
extension	Any encoding should allow to extend a Label with custom items	Any type	zero or more

6.14. CLASS FONT

Req 13 Implementations shall support the encoding of all properties of the FontClass and meet all of the tabulated constraints and notes.
http://www.opengis.net/spec/symbology/2.0/req/core/FontClass

The FontClass describes the font properties to apply for the rendering of a text string. It refers to the W3C CSS Fonts chapter.

FontClass properties are documented in the following table.

Table 14 — FontClass

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to the affected graphical properties within a Font	uom code	zero or one
fontFamily	Font family name (a)	ParameterValue data type CharacterString	zero or more
fontSize	Font size when applying the font to a text string (b)	ParameterValue data type Float	zero or one
fontWeight	Amount of weight or boldness to use for a font	ParameterValue data type CharacterString	zero or one
fontStyle	Style to use for a font	ParameterValue data type CharacterString	zero or one
extension	Any encoding should allow to extend a Font with custom items	Any type	zero or more

(a) Any number of FontFamily parameters may be given and they are assumed to be in preferred order. (b) The size unit is specified by the uom code if defined or by the innermost parent unit of measure definition otherwise.

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A Symbology Encoding implementation shall satisfy the following characteristics to be conformant with this specification.

The OGC URI identifier of this conformance class is: http://www.opengis.net/spec/symbology/2.0/conf/core.

It is the root of the test identifiers described below.

A.1. IMPLEMENTS THE STYLE CLASS.

Table A.1 — A.1 Implements the Style class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/StyleClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass
TEST METHOD:	Review all rows of the Style class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class of the Style class has an encoding rule defined.

A.2. IMPLEMENTS THE RULE CLASS.

Table A.2 — A.2 Implements the Rule class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/RuleClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass

TEST METHOD: Review all rows of the Rule class and confirm that an encoding is defined for each element in the encoding specification.

A.3. IMPLEMENTS THE SYMBOLIZER CLASS.

Table A.3 — A.3 Implements the Symbolizer Class

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/SymbolizerClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/SymbolizerClass

TEST METHOD: Review all rows of the Symbolizer class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class of the Symbolizer class (defined by the symbology conceptual model) has an encoding rule defined.

A.4. IMPLEMENTS THE PARAMETERVALUE CLASS.

Table A.4 — A.4 Implements the ParameterValue Class

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/ParameterValueClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/ParameterValueClass

TEST METHOD: Review all rows of the ParameterValue class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the ParameterValue abstract class is implemented.

A.5. IMPLEMENTS THE LITERAL CLASS.

Table A.5 — A-5 Implements the Literal class

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/LiteralClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/LiteralClass

TEST METHOD: Review all rows of the Literal class and confirm that an encoding rule is defined for each element in the encoding specification.

A.6. IMPLEMENTS THE UOM CODELIST.

Table A.6 — A.6 Implements the uom codelist

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/UOMClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/UOMClass

TEST METHOD: Check that at least the pixel portrayal unit of measure is implemented.

A.7. IMPLEMENTS THE COLOR CLASS.

Table A.7 — A.7 Implements the Color Class

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/ColorClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/req/core/ColorClass

TEST METHOD: Review all rows of the Color class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the Color abstract class is implemented.

A.8. IMPLEMENTS THE FILL CLASS.

Table A.8 — A.8 Implements the Fill Class

TEST ID: http://www.opengis.net/spec/symbology/2.0/conf/core/FillClass

TEST PURPOSE: To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/FillClass

TEST METHOD: Review all rows of the Fill class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the Fill abstract class is implemented.

A.9. IMPLEMENTS THE STROKE CLASS.

Table A.9 — A.9 Implements the Stroke Class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/StrokeClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/StrokeClass
TEST METHOD:	Review all rows of the Stroke class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the Stroke abstract class is implemented.

A.10. IMPLEMENTS THE GRAPHIC CLASS.

Table A.10 — A.10 Implements the Graphic Class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/GraphicClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/GraphicClass
TEST METHOD:	Review all rows of the Graphic class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the Graphic abstract class is implemented.

A.11. IMPLEMENTS THE GRAPHICSIZE CLASS.

Table A.11 — A.11 Implements the GraphicSize Class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/GraphicSizeClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/GraphicSizeClass
TEST METHOD:	Review all rows of the GraphicSize class and confirm that an encoding rule is defined for each element in the encoding specification.

A.12. IMPLEMENTS THE LABEL CLASS.

Table A.12 — A.12 Implements the Label Class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/LabelClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/LabelClass
TEST METHOD:	Review all rows of the Label class and confirm that an encoding rule is defined for each element in the encoding specification. Check that at least one concrete class (defined by the symbology conceptual model) of the Label abstract class is implemented.

A.13. IMPLEMENTS THE FONT CLASS.

Table A.13 — A.13 Implements the Font Class

TEST ID:	http://www.opengis.net/spec/symbology/2.0/conf/core/FontClass
TEST PURPOSE:	To test requirement http://www.opengis.net/spec/symbology/2.0/conf/req/FontClass
TEST METHOD:	Review all rows of the Font class and confirm that an encoding rule is defined for each element in the encoding specification.

B

ANNEX B (NORMATIVE) EXAMPLE IMPLEMENTATION

B

ANNEX B (NORMATIVE) EXAMPLE IMPLEMENTATION

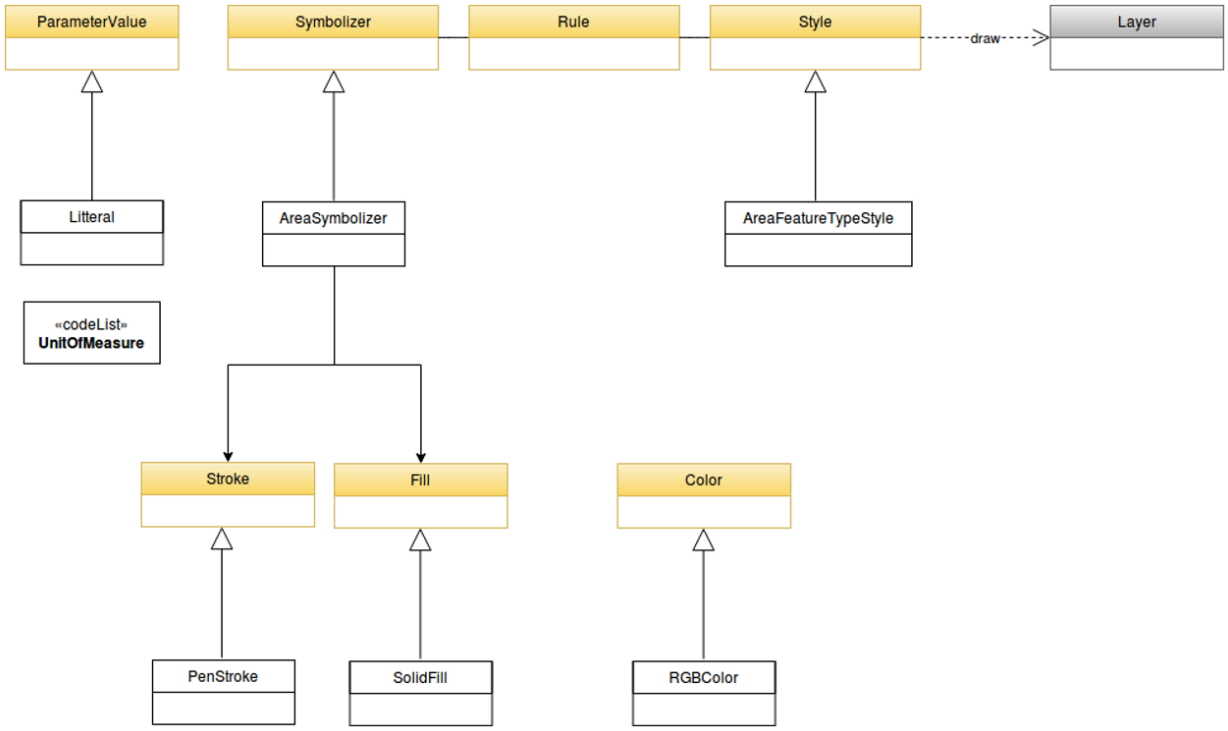
The Symbology Conceptual Core Model standard implies that a concrete style extension is implemented to control how the features of a certain data model are rendered. The following example is given for information only. It must be read and understood as a fictitious implementation of an extension.

Let us introduce the AreaFeatureTypeStyle extension which holds a simple and classical symbolizer, call it the AreaSymbolizer extension which describes the graphical parameters for drawing polygonal features with outlined and filled surface areas.

Figure B.1 shows the fundamental concepts of the Symbology Conceptual Core Model that must be implemented to create the AreaFeatureTypeStyle extension.

Note that this extension doesn't contain any rule mechanisms such as MinScaleDenominator and MaxScaleDenominator or/and filter operators.

Figure B.1 — UML Class Diagram of the AreaFeatureTypeStyle extension



AreaFeatureTypeStyle extension

The AreaFeatureTypeStyle extension defines the styling that is to be applied to a single feature type.

This class extends the abstract Style class described in the core. It defines the ability to portray of a Layer built of N instances of GML AbstractFeatureType[3] with the ability to access features according to Simple Feature SF-2[4].

The rendering engine is driven according to the following execution: all features are applied to each symbolizer in sequence as they appear nested in rule and following the "painter's model" with the first item in a list being the first item plotted and hence being on the "bottom".

Table B.1 — AreaFeatureTypeStyle extension — featureTypeName

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
featureTypeName	Identifies the specific feature type that the feature-type style is for	Literal data type	zero or one

To define the access to the exact feature type geometry to style, the extension adds a geometry property to the AreaSymbolizer with the related behaviours (described below for each requirement).

Table B.2 — AreaFeatureTypeStyle extension — Geometry

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
geometry	Geometry attribute or sub-element of a geometry attribute (a) added to the Symbolizer	ParameterValue data type Geometry[a]	one

[a] from http://www.opengis.net/doc/IS/SFA/6.1.2

(a) Add to symbolizer the geometry attribute, (b) Features according GML Simple Feature (SF-2 Level).

AreaSymbolizer extension

[3]Portele C. 2007. OpenGIS® geography markup language (GML) encoding standard, version 3.2.1 (OGC 07-036) (accessed august 2018).

[4]Van den Brink L, Portele C, Vretanos PA. 2012. Geography markup language (GML) simple features profile—with corrigendum (OGC 10-100r3) Wayland: Open Geospatial Consortium, Inc. Technical report (accessed august 2018).

An AreaSymbolizer is used to symbolize a geometry into an area including filling its interior and stroking its outline. It is typically used for a 2-dimensional geometry (e.g., Polygon).

Table B.3 — AreaSymbolizer extension

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
fill	Filling style to draw the interior area	Fill data type	Zero or one
stroke	Stroke style to draw the outline	Stroke data type	Zero or one

(a) If a geometry has "holes," then they are not filled, but the borders around the holes are stroked in the usual way if a Stroke parameter is mentioned. "Islands" within holes are filled and stroked, and so on. If a point is used, then a small, square, orthogonal-normal area should be constructed for rendering. If a line is used, then the line (string) is closed for filling (only) by connecting its end point to its start point, any line crossings are corrected in some way, and only the original line is stroked. (b) A missing Fill property means that the geometry will not be filled. A missing Stroke property means that the geometry will not be stroked. When both are used, the filling is rendered first and then the stroking rendered on top of the filling.

SolidFill extension

The SolidFill class is a concrete implementation of the Fill class and allows to formulate a filling of an area (e.g., a polygon geometry or any kind of symbol).

Table B.4 — SolidFill extension

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
color	The color to fill the area	Color data type	zero or one
opacity	Opacity of the Color	ParameterValue type (Float) Value: [0;1] (1 means 100% opaque) Default value: 0	zero or one

Note: Any fill implementation can be imagined as for example a TextureFill extension to add a level of artistic control and realism to a surface. For complex Texture properties, a specific Symbolizer could be proposed.

PenStroke extension

The PenStroke extension is a concrete implementation of the Stroke class. It allows to draw a line (e.g., a 1-dimensional geometry, the outline of a marker, etc.) analogously to how a pen is used with ink, that is to say by filling the area formed by the thickness of the line.

Table B.5 — PenStroke extension

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
width	Thickness of the line which gives form to an area to fill (a)	ParameterValue data type (Float) Value: [0;+∞) Default value: 1px	zero or one
fill	The filling style to draw the linear area	Fill data type	zero or one

(a) The Width parameter is in the context of a UnitOfMeasure code (that may be inherited from a parent element).

RGBColor extension

The RGBColor extension is a concrete implementation of the Color class where the color is expressed as three integer properties in conformance with the sRGB standardized color space.

Table B.6 — RGBColor extension

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
red	The red value of the color	ParameterValue data type (Integer) Value: (0;255) Default value: 0	one
green	The green value of the color	ParameterValue data type (Integer) Value: (0;255) Default value: 0	one
blue	The blue value of the color	ParameterValue data type (Integer) Value: (0;255) Default value: 0	one

C

ANNEX C (NORMATIVE) A REDESIGN OF OGC SYMBOLOGY ENCODING STANDARD FOR SHARING CARTOGRAPHY

C

ANNEX C (NORMATIVE) A REDESIGN OF OGC SYMBOLOGY ENCODING STANDARD FOR SHARING CARTOGRAPHY

The ins and outs of the SymCore conceptual model are explained through the research article, below. Also, this article helps the reader to understand how the conceptual model can be extended (core and extensions approach).

Bocher E, Ertz O. 2018. A redesign of OGC Symbology Encoding standard for sharing cartography. PeerJ Computer Science 4:e143 https://doi.org/10.7717/peerj-cs.143

The abstract of the article is imported here.

> Despite most Spatial Data Infrastructures offering service-based visualization of geospatial data, requirements are often at a very basic level leading to poor quality of maps. This is a general observation for any geospatial architecture as soon as open standards as those of the Open Geospatial Consortium (OGC) are applied. To improve the situation, this paper does focus on improvements at the portrayal interoperability side by considering standardization aspects. We propose two major redesign recommendations. First to consolidate the cartographic theory at the core of the OGC Symbology Encoding standard. Secondly to build the standard in a modular way so as to be ready to be extended with upcoming future cartographic requirements. Thus, we start by defining portrayal interoperability by means of typical-use cases that frame the concept of sharing cartography. Then we bring to light the strengths and limits of the relevant open standards to consider in this context. Finally we propose a set of recommendations to overcome the limits so as to make these use cases a true reality. Even if the definition of a cartographic-oriented standard is not able to act as a complete cartographic design framework by itself, we argue that pushing forward the standardization work dedicated to cartography is a way to share and disseminate good practices and finally to improve the quality of the visualizations.

D

ANNEX D (NORMATIVE) REVISION HISTORY

D

ANNEX D (NORMATIVE) REVISION HISTORY

Table D.1 — Revision history

DATE	RELEASE	AUTHOR	PARAGRAPH MODIFIED	DESCRIPTION
2020-08-27		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Edits considering received and answered comments from TC vote
2019-09-18		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Edits considering received and answered comments (18-067r2)
2019-08-21		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Update abstract to be more comprehensive
2019-03-29		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Edits considering received and answered comments
2018-09-07		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Release for early public comment (18-067)
2018-06-05		E. Bocher (CNRS) O. Ertz (HEIG-VD)		Initial document