Open Geospatial Consortium

Saved 2018/09/05 16:43

Reference URL for this document: www.opengeospatial.net/doc/IS/?

Internal reference number of this OGC ® document: 18-067

Version: 1.0

Category: OGC ® Standard Specification

Editor(s): Erwan Bocher, Olivier Ertz

OGC Symbology Conceptual Model: Core part

Document type: OGC ® Implementation Standard

Document subtype: Conceptual Model

Document stage: Draft

Document language:English

Table of Contents

i. Abstract

This document presents the requirements for defining the Symbology Conceptual Core Model (SCCM). The SCCM is an abstract, 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, that this document does not define any extensions.

ii. Keywords

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

Ogcdoc, style, symbology, conceptual, core, modular, neutral, portrayal

iii. Preface

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.

iv. Submitting organizations

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

HEIG-VD (School of Management and Engineering Vaud) https://heig-vd.ch/

CNRS (National Center for Scientific Research) http://www.cnrs.fr

v. Submitters

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

Name	Affiliation
Olivier Ertz	HEIG-VD
Erwan Bocher	CNRS

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 made both types of comments. Written comments were submitted by the following individual members curing the public review:

Name	Affiliation
Carl Reed	Carl Reed and Associates
Jeff Yutzler	Image Matters LLC
Chris Little	Met Office

1 Scope

This document presents the requirements that define the Symbology Conceptual Core Model (SCCM). The SCCM is an abstract, 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, that this document does not define any extensions.

Even though the SCCM 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, Augmented Reality topics (Figure 1).

The SCCM is a new approach (Bocher E, Ertz O, 2018) :

- 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, ...
- to achieve a high level styling interoperability without encoding dependencies.

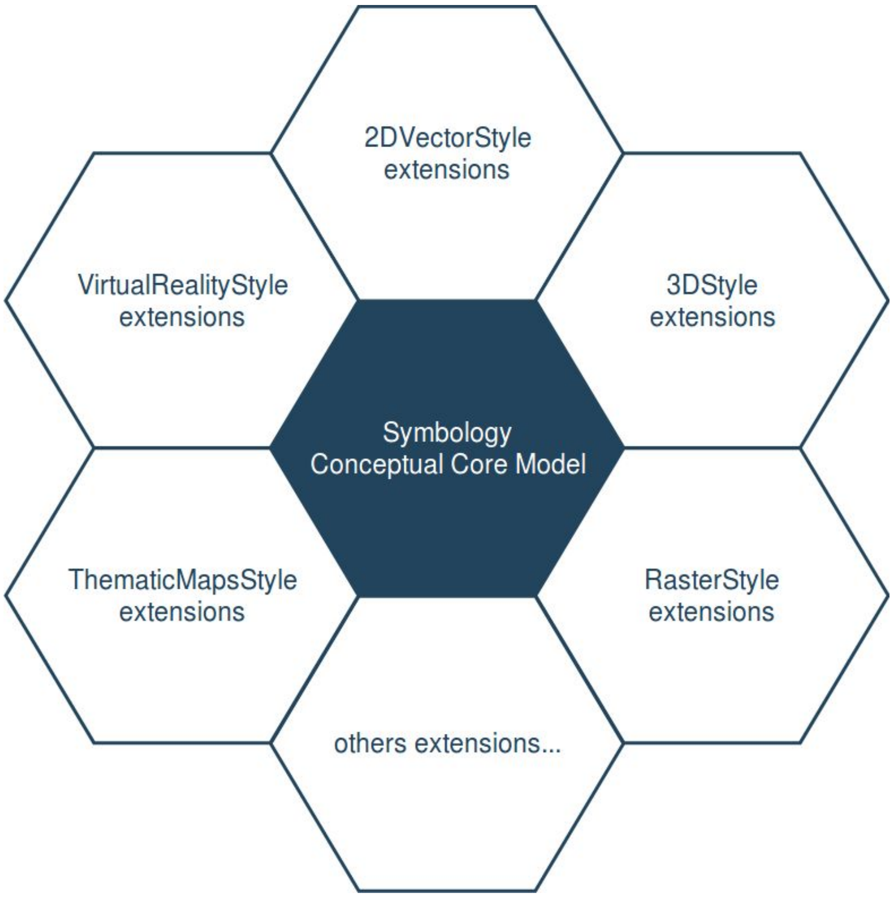
2DVectorStyle extensions
VirtualRealityStyle extensions
3DStyle extensions
Symbology Conceptual Core Model
ThematicMapsStyle extensions
RasterStyle extensions
others extensions...

Figure 1. The core model and its potential extensions

Note :

2 Conformance

This document defines a standardisation target for extensions 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 stated above.

Conformance with this standard shall be checked using all the relevant tests specified in Chapter 6 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.

3 Normative References

OGC 06-042, OpenGIS Web Map Service (WMS) Implementation Specification, Version 1.3.0 http://www.opengeospatial.org/standards/wms

OGC 05-078r4, OpenGIS Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0 http://www.opengeospatial.org/standards/sld

OGC 05-077r4, OpenGIS Symbology Encoding Implementation Specification, Version 1.1.0 http://www.opengeospatial.org/standards/se

OGC 09-026r2, OGC Filter Encoding 2.0 Encoding Standard - With Corrigendum, http://www.opengeospatial.org/standards/filter

OGC 08-131r3, The Specification Model - A Standard for Modular specifications. https://portal.opengeospatial.org/files/?artifact_id=34762

IETF RFC 4646: Tags for Identifying Languages, https://datatracker.ietf.org/doc/rfc4646/

ISO 19117:2012, Geographic information - Portrayal https://www.iso.org/standard/46226.html

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

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

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

4 Definitions and Abbreviations

4.1 Definitions

4.1.1 Portrayal

The ISO defines portrayal as presentation of information for human. See ISO 19117:2012, Geographic information - Portrayal https://www.iso.org/standard/46226.html

4.1.2 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.

4.1.3 Rendering engine

A rendering engine is an automated process that produces graphics using a pipeline of layers and styles as inputs. A rendering engine is commonly found in desktop or server-based geographic information systems.

4.1.4 Render

Conversion of digital graphics data into visual form. See ISO 19117:2012, Geographic information - Portrayal https://www.iso.org/standard/46226.html

4.2 Abbreviations

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

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/

5 Core Conceptual Model

5.1 Overview

The requirements described in this section define the symbology core requirement class. The UML diagram (Figure 2) shows the fundamental concepts of the Symbology Conceptual Core Model.

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

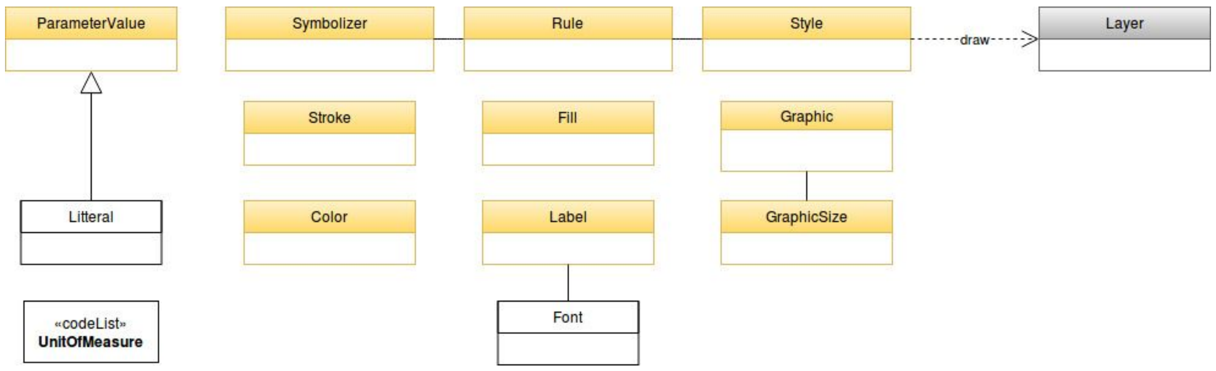

Figure 2. UML Class Diagram of the Symbology core

Scope: All requirements in this subsection relate to the above requirement class.

Dependencies: None

Description: The Symbology conceptual model is shown in the UML Diagram below.

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

- either globally related to an abstract class (in yellow/italic)
- 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.

5.2 Class Style

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass
Requirement Txt: Implementations shall support the encoding of all properties of the Style class and meet all of the tabulated constraints and notes.

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 i 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 Style class properties are documented in the following table.

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

5.3 Class Rule

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/RuleClass

Requirement Txt: Implementations shall support the encoding of all *RuleClass* properties and meet all of the tabulated constraints and notes.

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.

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
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

5.4 Class Symbolizer

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/SymbolizerClass

Requirement Txt: Implementations shall support the encoding of all *SymbolizerClass* properties and meet all of the tabulated constraints and notes.

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.

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 we want to symbolize 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 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 locate the “Hospitals” or a LabelSymbolizer to render the road name along a line.

5.5 Class ParameterValue

Requirement ID:
http://www.opengis.net/spec/symbology/2.0/req/core/ParameterValueClass

Requirement Txt: Implementations shall support the encoding of all *ParameterValue* parameters class and meet all of the tabulated constraints and notes.

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.

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.

5.6 Class Literal

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/LiteralClass
Requirement Txt: Implementations shall support the encoding of all parameters of the Literal class and meet all of the tabulated constraints and notes.

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.

Name	Definition	Data type and value	Multiplicity
value	A value for the literal data	Any	one

5.7 UOM Codelist

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

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 (eg. Symbolizer, Stroke, Fill, GraphicSize...) 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
- 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).

5.8 Class Color

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/ColorClass
Requirement Txt: Implementations shall support the encoding of all properties of the Color class and meet all of the tabulated constraints and notes.

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.

Name	Definition	Data type and value	Multiplicity
extension	Any encoding should allow the extension of *ColorClass* with custom items	Any type	zero or more

5.9 Class Fill

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

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.

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

5.10 Class Stroke

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

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.

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

5.11 Class Graphic

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

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.

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

5.12 Class GraphicSize

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

Requirement Txt: Implementations shall support the encoding of all properties of the *GraphicSizeClass* and meet all of the tabulated constraints and notes.

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.

Name	Definition	Data type and value	Multiplicity
extension	Any encoding should allow to extend a GraphicSize with custom items	Any type	zero or more

5.13 Class Label

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/LabelClass

Requirement Txt: Implementations shall support the encoding of all properties of the *LabelClass* and meet all of the tabulated constraints and notes.

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 behaviours (e.g., a PointLabel or LineLabel).

LabelClass properties are documented in the following table.

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

5.14 Class Font

Requirement ID: http://www.opengis.net/spec/symbology/2.0/req/core/FontClass

Requirement Txt: Implementations shall support the encoding of all properties of the *FontClass* and meet all of the tabulated constraints and notes.

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

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

6 Abstract Test Suite (Normative)

An 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.

6.1 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

6.7 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.

6.8 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.

6.9 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.

6.10 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.

6.11 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.

6.12 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.

6.13 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.

Annexe A Example implementation

The Symbology Conceptual Core Model standard implies that a concrete style extension is implemented to control how the features of a certains 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.

The figure 3 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 as MinScaleDenominator and MaxScaleDenominator or/and filter operators.

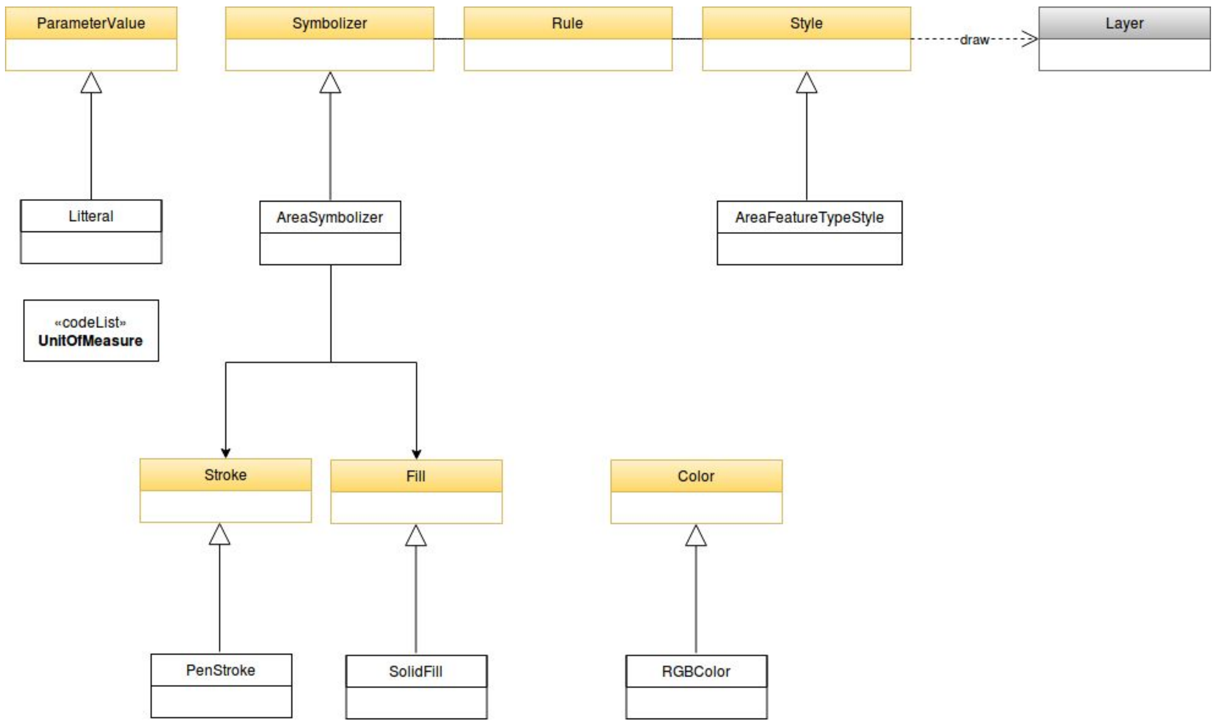

Figure 3. UML Class Diagram of the AreaFeatureTypeStyle extension

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 (Portele, 2007) with the ability to access features according to Simple Feature SF-2 (Van den Brink et al., 2012).

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 "painters model" with the first item in a list being the first item plotted and hence being on the "bottom".

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).

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[1]	one

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

AreaSymbolizer extension

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).

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.

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

(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).

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 a 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.

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.

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)	one

		Default value: 0	

Annexe references

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

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)