S-100 - Part 9

Portrayal

10 April 2013

1 Inhalt

2 Scope

This part of the standard describes a Portrayal Catalogue and its contents. The concept in this standard is that feature data is modelled with a focus on content and portrayal or display of a feature is accomplished using rules or functions that map the content to the appropriate symbols and display characteristics. This concept allows the same content to be displayed in different ways and allows the display mapping rules to be maintained without having to modify all the content data.

The Portrayal Catalogue contains portrayal functions that map the features to symbology it also contains symbol definitions, colour definitions, portrayal parameters and portrayal management concepts such as viewing groups. The goal in S-100 is to provide a mechanism where, for a given product, the portrayal catalogue can be delivered as data in a machine readable form such that a compliant implementation can display the product feature data using the given Portrayal Catalogue.

3 Conformance

This part of the specification conforms to ISO 19117 : 2012 (E) according to the Annex A Abstract test suite.

EditorialNote: Conceptually conformance seems complete however classes and terms in 19117 are different than in this spec. We should either adopt the 19117 terms or create a crosswalk between this and 19117.

4 Normative references

ISO 19117 : 2012 (E)

5 Symbols and abbreviated terms

To be done

6 General portrayal model

The general portrayal model is illustrated in figure 1.

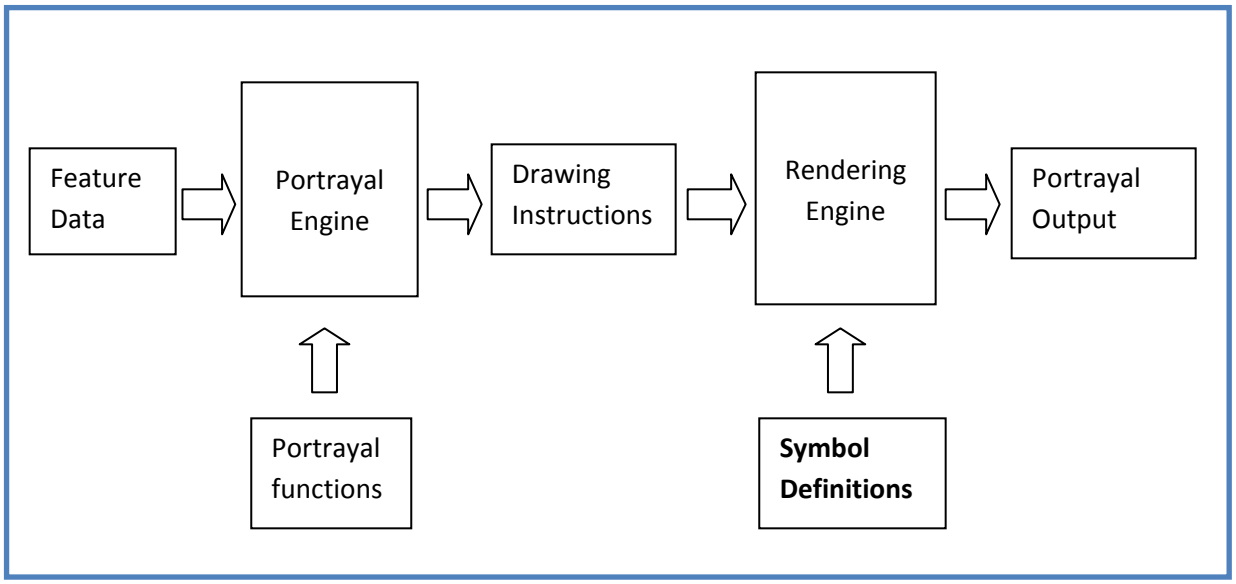


Figure 1

This part of S-100 defines a feature-centred function-based portrayal mechanism. Instances of features are portrayed based on portrayal functions, which make use of geometry and attribute information. The relationship between the feature instances, attributes, and the underlying spatial geometry is specified in a product specification based on the General Feature Model of S-100.

Portrayal information is needed to portray a dataset containing geographic data. The portrayal information is defined as drawing instructions created by specific portrayal functions. The portrayal mechanism makes it possible to portray the same dataset in different ways without altering the dataset itself.

The drawing instructions are intermediate data used by the rendering engine to produce the portayal output. During the rendering process, the rendering engine uses the symbol definitions to create the output according to the output device.

The symbol definitions contain the details of all graphical elements used for the portrayal. The model of the symbol definitions is described in this document.

All about 2D (some words needed)

6.1 The portrayal process

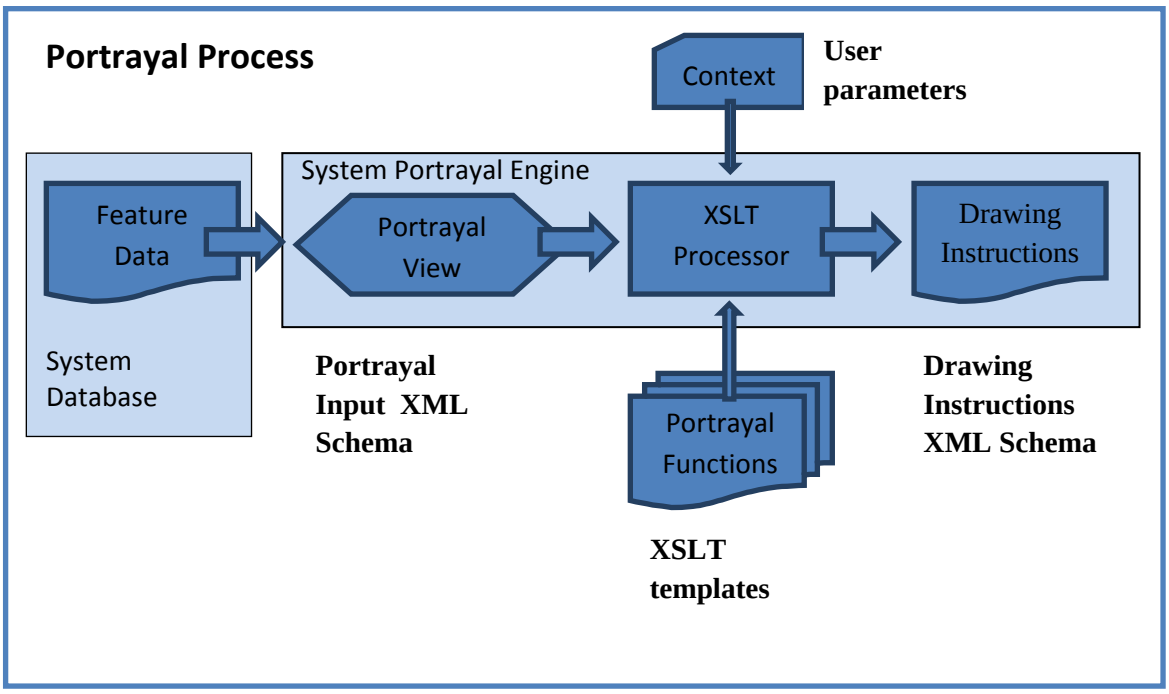


Figure 2

The system has Feature data within its internal database that needs to be portrayed. The System Portrayal Engine transforms the Feature data into drawing instructions. Drawing instructions include such things as references to symbol definitions, priority and filtering information. The drawing instructions are further processed by the rendering engine to produce the final display.

In this process, feature data needs to be exposed to the XSLT processor as XML content. The XSLT processor applies the best matching template or portrayal function to each feature. The portrayal function uses the defined logic to transform the input feature content along with related context information into drawing instructions which are output as XML.

The functionality of the System Portrayal Engine is defined in terms of XSLT. XSLT is a declarative language. An XSLT processor transforms XML input into XMLoutputs. Contextual and user parameters can be fed into the XSLT processor for use by the portrayal functions. Portrayal functions in XSLT can range from simple lookup or best match templates to complex conditional logic. XSLT is defined to work on an XML node treehowever there are implementations that interface the XSLT processor directly with internal structures or relational database tables. Although there are newer versions of XSLT, XSLT 1.0 (http://www.w3.org/TR/xslt) has been chosen for this portrayal specification as the most commonly supported.

This portrayal specification defines how machine readable portrayal transformation functions are implemented as XSLT templates disseminated in XSL files. Since XSLT is defined to operate on XML and

produce XML the XML input and output schemas are defined as part of this specification. A conformant System Portrayal Engine must operate consistently with XSLT in order to process the machine readable XSL files and produce equivalent output.

7 Data input schema

7.1 Introduction

The data input schema describes how the data is presented to the XSLT processor. The data can be transformed to an XML document or a presentation of such a document e.g. a DOM-tree. It is also possible to model the data to look like XML and use a special software interface to present such data to the XSLT processor.

Whatever method is used this schema describes how the data must be organized. In this standard only the base types are described. The actual feature types of a data product must be specified in a schema that will be part of the product specification. The data types of such a schema will correspond to the portrayal rules of the same product specification. All feature types in a product must be based on the types specified in this schema.

The schema contains also data types for spatial objects and for associations. Whenever such types are not sufficient for a specific data product, appropriate types can be derived from the types in this standard. This may be the case for spatial objects that needs to have associations to quality information types.

Note:

It is assumed for the examples in this section that types of this schema are in the namespace `s100`.

Editorial note:

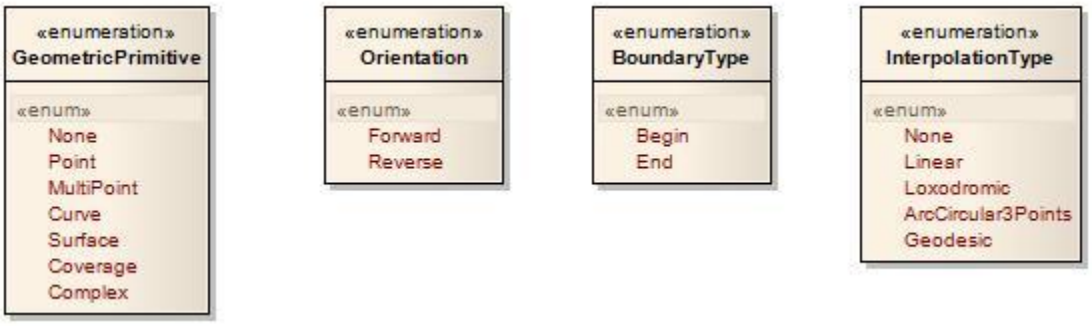
The intent is to map this schema to the corresponding S-100 GML elements. Since GML is XML it can be used as input to the XSLT processor. We need more support from the group that has developed the GML profile to understand and to document how to do this.

One example is how to define the relationships between features and geometry including direction and masking.

The S-100 GML profile should be based on the latest version of the GFM. It should support attributes at associations, shared geometry and associations from spatial types to information types.

7.2 Enumerations

For the use in this schema the following enumeration types are defined:



GeometricPrimitive

This enumeration describes the type of geometric primitive that is used by a feature object. If the feature object uses different geometric primitives the value `Complex` has to be used.

```
<xs:simpleType name="GeometricPrimitive">
<xs:restriction base="xs:string">
<xs:enumeration value="None"/>
<xs:enumeration value="Point"/>
<xs:enumeration value="MultiPoint"/>
<xs:enumeration value="Curve"/>
<xs:enumeration value="Surface"/>
<xs:enumeration value="Coverage"/>
<xs:enumeration value="Complex"/>
</xs:restriction>
</xs:simpleType>
```

Orientation

The enumeration`Orientation` is used to specify the orientation of a referenced geometry that is used by a feature object or by a complex curve.

```
<xs:simpleType name="Orientation">
<xs:restriction base="xs:string">
<xs:enumeration value="Forward"/>
<xs:enumeration value="Reverse"/>
</xs:restriction>
</xs:simpleType>
```

BoundaryType

This enumeration describes the type of a topologic boundary.

```
<xs:simpleType name="BoundaryType">
<xs:restriction base="xs:string">
<xs:enumeration value="Begin"/>
<xs:enumeration value="End"/>
</xs:restriction>
</xs:simpleType>
```

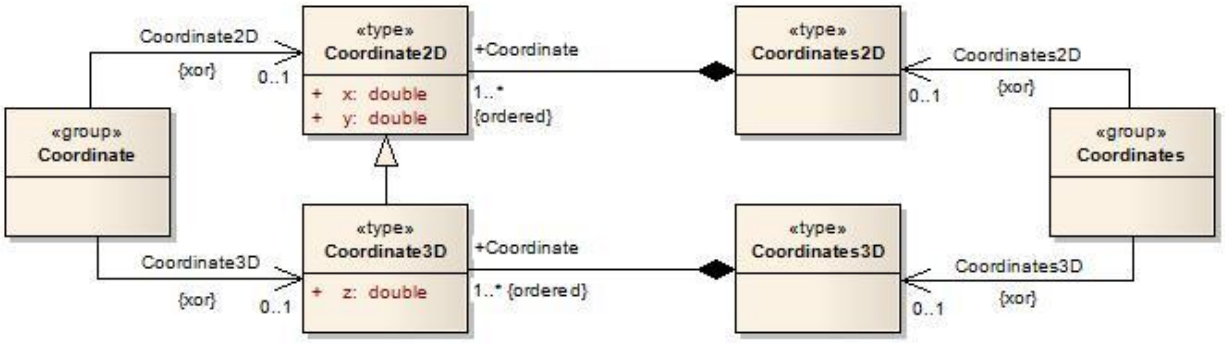
`InterpolationType`

This enumeration describes the mathematical interpolation method between two control points in a line segment. Note that the methods depend on the underlying coordinate reference system and not all of them are valid for all types of CRS. The product specification should specify the details of the use of interpolation.

```
<xs:simpleType name="InterpolationType">
<xs:restriction base="xs:string">
<xs:enumeration value="Linear"/>
<xs:enumeration value="Loxodrome"/>
<xs:enumeration value="Orthodrome"/>
</xs:restriction>
</xs:simpleType>
```

7.3 Coordinates

In case that coordinates have to be presented to the XSLT processor the following typeshave to be used.



The types `Coordinate2D` and `Coordinate3D` are for a simple coordinate tuple. They are defined as:

```
<xs:complexType name="Coordinate2D">
<xs:sequence>
<xs:element name="x" type="xs:double"/>
<xs:element name="y" type="xs:double"/>
```

```
</xs:sequence>
</xs:complexType>

<xs:complexType name="Coordinate3D">
<xs:complexContent>
<xs:extension base="Coordinate2D">
<xs:sequence>
<xs:element name="z" type="xs:double"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Note that the type `Coordinate3D` is an extension of the type `Coordinate2D`.

Example:

```
<s100:Coordinate2D>
<s100:x>9.12345</s100:x>
<s100:y>52.56789</s100:y>
</s100:Coordinate2D>
```

And

```
<s100:Coordinate2D>
<s100:x>9.12345</s100:x>
<s100:y>52.56789</s100:y>
<s100:z>12.5</s100:z>
</s100:Coordinate2D>
```

A group `Coordinate` is defined where coordinate tuple can be used mutually exclusive.

```
<xs:group name="Coordinate">
<xs:choice>
<xs:element name="Coordinate2D" type="Coordinate2D"/>
<xs:element name="Coordinate3D" type="Coordinate3D"/>
</xs:choice>
</xs:group>
```

For a sequence of coordinate tuples the types `Coordinates2D` and `Coordinates3D` are defined. Sequences of coordinates are used for point sets and line segments. The definition looks like:

```
<xs:complexType name="Coordinates2D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate2D" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="Coordinates3D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate3D" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

Note that the default value for minOccurs is 1, i.e. at least one coordinate tuple must be in the list.

Example:

```
<s100:Coordinates3D>
<s100:Coordinate>
<s100:x>12.345</s100:x>
<s100:y>53.456</s100:y>
<s100:z>5.3</s100:z>
</s100:Coordinate>
<s100:Coordinate>
<s100:x>12.458</s100:x>
<s100:y>53.789</s100:y>
<s100:z>7</s100:z>
</s100:Coordinate>
</s100:Coordinates3D>
```

A group is specified for sequences of coordinates just like for single coordinate tuples.

```
<xs:group name="Coordinates">
<xs:choice>
<xs:element name="Coordinates2D" type="Coordinates2D"/>
<xs:element name="Coordinates3D" type="Coordinates3D"/>
</xs:choice>
</xs:group>
```

7.4 Associations

According to the general feature model there are two types of associations:

For each association a separate type is defined in the schema:

```
<xs:complexType name="InformationAssociation">
<xs:attribute name="informationRef" type="xs:string" use="required"/>
<xs:attribute name="role" type="xs:string" use="required"/>
</xs:complexType>
```

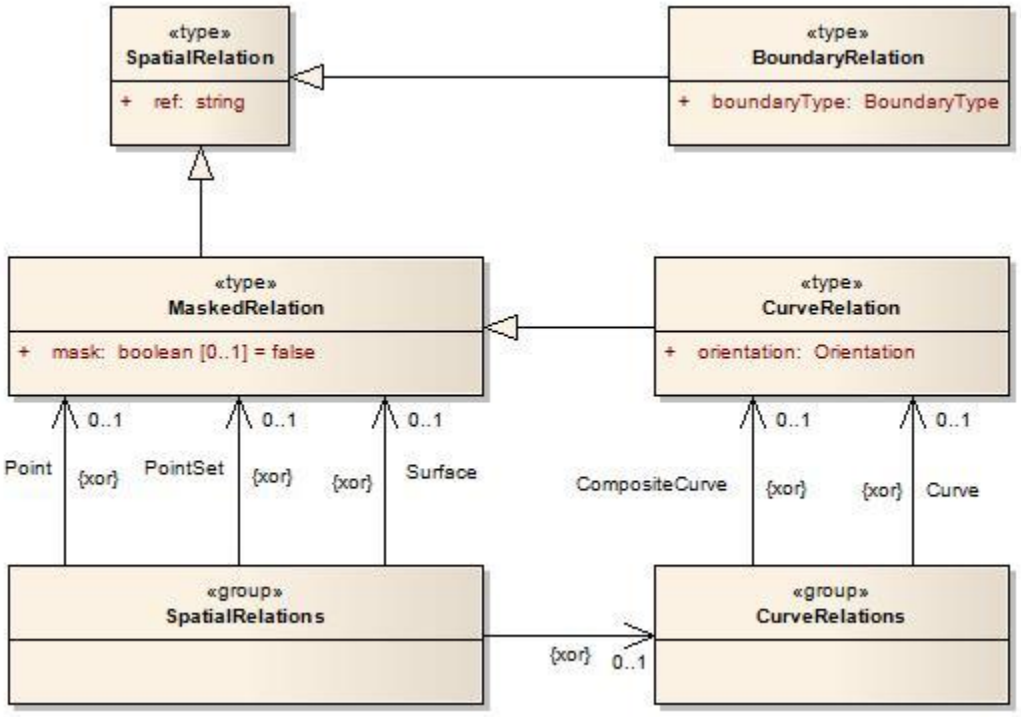
```
<xs:complexType name="FeatureAssociation">
<xs:attribute name="featureRef" type="xs:string" use="required"/>
<xs:attribute name="role" type="xs:string" use="required"/>
</xs:complexType>
```

The attributes `informationRef` and `featureRef` correspond to the attribute `id` of the referenced information respective feature object. See the section on objects for more details.

If a product specification requires attributes for an association a specific type must be sub-classed in the schema of the product specification.

7.5 Spatial relations

In the general feature model different relations are modelled between feature types and spatial types but also between spatial types. For such relations the following types are defined by this schema.



The type `SpatialRelation` is the base type for all relations to spatial objects. It defines only one attribute `ref` that corresponds to the attribute `id` of the spatial object.

```
<xs:complexType name="SpatialRelation">
<xs:attribute name="ref" type="xs:string" use="required"/>
</xs:complexType>
```

The other relation types are derived from this type and add information according to the specific use of that relation. The type `MaskedRelation` adds an attribute `mask` that specifies if a referenced spatial object should not be used for portrayal.

```
<xs:complexType name="MaskedRelation">
<xs:complexContent>
<xs:extension base="SpatialRelation">
<xs:attribute name="mask" type="xs:boolean" default="false"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Note that the attribute mask is not mandatory but has a default value for the case of its absence.

The type `BoundaryRelation` adds a boundary type to the relation and is used when the relation describes a topological relation e.g. the relation to a bounding node of a curve.

```
<xs:complexType name="BoundaryRelation">
<xs:complexContent>
<xs:extension base="SpatialRelation">
<xs:attribute name="boundaryType" type="BoundaryType" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The type `CurveRelation` is used whenever a curve is referenced by a spatial relation since it is necessary to specify if the curve is used in the same direction as it is defined or in the reverse order. The type is derived from `MaskedRelation` since each curve can be a subject of masking.

```
<xs:complexType name="CurveRelation">
<xs:complexContent>
<xs:extension base="MaskedRelation">
<xs:attribute name="orientation" type="Orientation" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Two groups are defined for Spatial relations. One group defines the possible relations two curves the other defines all possible spatial relations.

```
<xs:group name="CurveRelations">
<xs:choice>
<xs:element name="Curve" type="CurveRelation"/>
<xs:element name="CompositeCurve" type="CurveRelation"/>
</xs:choice>
</xs:group>
```

```
<xs:group name="SpatialRelations">
<xs:choice>
<xs:element name="Point" type="MaskedRelation"/>
<xs:element name="PointSet" type="MaskedRelation"/>
<xs:element name="Surface" type="MaskedRelation"/>
<xs:group ref="CurveRelations"/>
</xs:choice>
</xs:group>
```

How these groups are used is demonstrated in the section on schemas for a specific product.

7.6 Objects

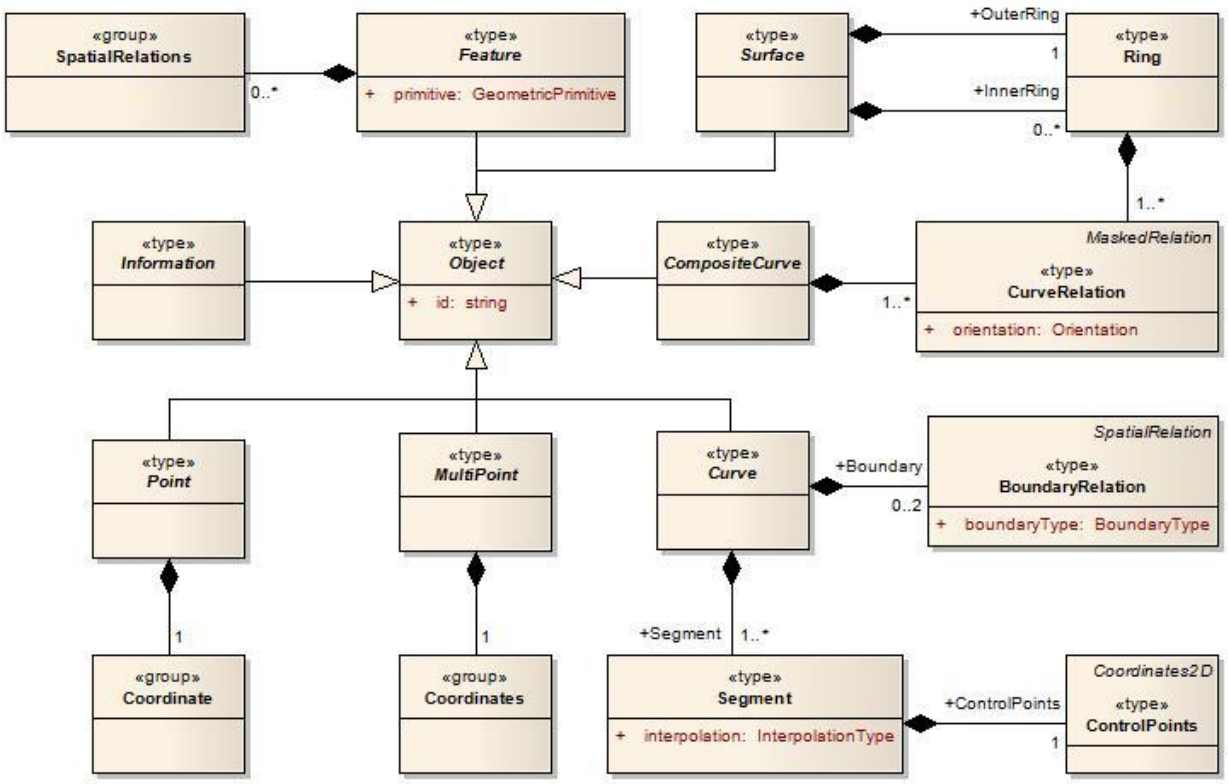
All objects in a data set are based on the type `Object` which carries the common properties of all objects. The only commonality on objects is the identifier. Each objects needs to be identifiable within a data set. This is done by the attribute `id`.

In the product specific schemas constraints can be made on this identifiers, in particular by the use of the <xs:key> and <xs:keyref> elements.

```
<xs:complexType name="Object" abstract="true">
<xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
```

Note that the type of the identifier is xs:string to be as general as possible with respect to different methods used for identification.

The model of all objects is given in following diagram.



7.7 Spatial objects

7.7.1 Preface

Spatial objects in a data set carry the geometric location of a feature object. The following types are supported by this standard:

- Point
- MultiPoint
- Curve
- Composite curve
- Surface

All types described here are abstract and have to be sub-classed in the product schema. Additional properties can then be added in the derived types. Such properties may be association to quality information types or other associations. Attributes are not permitted for spatial objects by the GFM. All types are derived from the type Object i.e. they have an identifier.

7.7.2 Point

A point carries a single coordinate tuple, 2D or 3D. The definition looks like.

```
<xs:complexType name="Point"abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
```

```
<xs:group ref="Coordinate"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Note that the group `Coordinate` is used within the definition to allow both `Coordinate2D` and `Coordinate3D` elements.

7.7.3 MultiPoint

Similar to Point this type defines point geometry for a feature object. The difference is that a set of tuples can be defined. Therefore the group that is referenced in the definition is `Coordinates` allowing both `Coordinates2D` and `Coordinates3D` elements.

```
<xs:complexType name="MultiPoint"abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="Coordinates"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

7.7.4 Curve

Curves describe the line geometry of a feature object. They are made of segments where each segment has a sequence of control points and an interpolation method. The latter defines the geometry between the control points according to the used coordinate reference system.

The type `ControlPoints` restricts the type `Coordinates2D` to have at least 2 coordinate tuples.

```
<xs:complexType name="ControlPoints">
<xs:complexContent>
<xs:restriction base="Coordinates2D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate2D" minOccurs="2" maxOccurs="unbounded"/>
</xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>
```

The type `Segment` simply combines the control points with an interpolation method.

```
<xs:complexType name="Segment">
<xs:sequence>
<xs:element name="ControlPoints" type="ControlPoints"/>
```

```
</xs:sequence>
<xs:attribute name="interpolation" type="InterpolationType" use="required"/>
</xs:complexType>
```

The type `Curve` finally combines a sequence of segments with the topological boundary. The topological boundary of a curve is the beginning and end node implemented by a `Point` object.

```
<xs:complexType name="Curve"abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:element name="Boundary" type="BoundaryRelation" minOccurs="0" maxOccurs="2"/>
<xs:element name="Segment" type="Segment" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

7.7.5 CompositeCurve

A composite curve describes the line geometry of a feature object just like a 'simple' curve. But instead using coordinates to define the geometry it is using a sequence of other curves, including other composite curves. With other words it is a sequence of relations to other curves.

```
<xs:complexType name="CompositeCurve"abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="CurveRelations" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

7.7.6 Surface

Surfaces describe the area geometry of a feature object. The surface itself is defined by its boundary. The boundary consists of an outer ring and optionally a number of inner rings. The inner rings describe holes in the area. Each ring is a closed polygon made from one or many curves. That means that a ring is very similar to a composite curve but unlike the composite curve it is not derived from `Object` because it does not need to be identifiable. The definition of a ring simply looks like:

```
<xs:complexType name="Ring">
<xs:group ref="CurveRelations" minOccurs="1" maxOccurs="unbounded"/>
</xs:complexType>
```

And the definition of a surface finally is:

```
<xs:complexType name="Surface"abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:element name="OuterRing" type="Ring"/>
<xs:element name="InnerRing" type="Ring" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

7.8 Information objects

Information object are identifiable and sharable pieces of information within a data set. In the model an abstract type `Information`is derived from the type `Object`. Although no additional properties are added this type is useful for semantic reasons. Information types in a product specification can be derived from the type `Information` to indicate that they are information types.

```
<xs:complexType name="Information" abstract="true">
<xs:complexContent>
<xs:extension base="Object"/>
</xs:complexContent>
</xs:complexType>
```

Note that the type is abstract. Any realization of an information type in a data product has to be derived from this type.

7.9 Feature objects

Feature objects are abstractions of real world phenomena. This schema defines the abstract base type for any feature type. The type Feature is derived from Object and adds a sequence of spatial relations and a geometric primitive attribute to the properties from the base class. All feature types in a product specification will be derived from this type. They can carry additional elements for feature attributes, feature associations or information associations.

```
<xs:complexType name="Feature" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="SpatialRelations" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="primitive" type="GeometricPrimitive" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

All feature types in a product has to be derived from this abstract type. How to do this is described in an annex of this standard.

For schema definition see A.1 Input Schema

8 Portrayal processing

This specification is referencing XSLT 1.0 which is a W3C recommendation, http://www.w3.org/TR/xslt

XSLT uses XPath 1.0 to address components of a document.http://www.w3.org/TR/xpath/

XSLT (XSL Transformations) is a language expressed as a well formed XML document. The intended purpose of using XSLT in portrayal is to transform the data into drawing instructions. Since XSLT is expressed in XML it is useful for interchange as a machine readable transformation language. XSLT is widely used across many domains but is perhaps most commonly used to transform XML documents into HTML for web page displays. There are many tutorials, books and reference material available for XSLT. There are also several web sites where questions can be posted and examples can be found.

XSLT uses templates to process nodes in the input XML tree and generate nodes as output XML, other SGML formats or even plain text. There are two types of templates a matching template and a named template.

Matching templates use a matching expression using XPATH to specify what elements in the input document should be processed by that template. XPATH (XML Path Language) is an expression language used to address or find components in an XML document. The path capability makes it especially useful when dealing with a hierarchy of content such as nested complex attributes. Only one matching template can match an element from the input document. Matching templates have a built in priority calculation and conflict resolution method that is used to determine which one to use in the case where multiple templates match the same element. Priority numbers can be explicitly assigned as an attribute of a matching template in order to override the default conflict resolution behaviour.

Named templates are called by another template along with the data to be processed. Named templates can also have parameters. These are useful for formatting or other operations that are commonly used in a transformation. A named template can even call itself (recursion), which can be useful for operations such as string token parsing.

A template can loop over a set of nodes that match an XPath expression using an “xsl:apply-templates “ or “xsl:for-each” instruction element. The nodes can also be sorted before being processed. Conditional processing is available by using a simple “xsl:if” instruction or an “xsl:choose” instruction. The choose instruction allows a set of expressions to be tested such that only the first one matching is processed and if no match is found an optional otherwise statement is used to handle a default. This is useful for testing enumerated data such that a different output is generated depending upon the enumeration value.

XLST also includes the ability to have parameters passed at the top level and accessed within any of the templates. These parameters can be useful to provide contextual information to the transformation. There are also variables in XSLT but they can only be assigned data as part of their definition,unlike other languages where variables can be reassigned.Variables are useful to collect data or decision results that can be passed as parameters to another template or used in conditional statements.

XSLT can include or import other XSLT documents. This capability can be useful for management of templates and reuse of templates by multiple top level XLST documents.

Examples

Given the example XML below

```
<BeaconCardinal id="2">
<s100:Point ref="3"/>
<categoryOfCardinalMark>3</categoryOfCardinalMark>
</BeaconCardinal>
<BeaconCardinal id="3">
<s100:Point ref="3"/>
<categoryOfCardinalMark>2</categoryOfCardinalMark>
</BeaconCardinal>
```

A simple matching XSLT template used as a portrayal function

```
<xsl:template match="BeaconCardinal">
        <!—This is a comment. This template matches a BeaconCardinal node and the body of the template can examine data and output results -->

</xsl:template>
```

The above template will be used to process all of the BeaconCardinal objects.

The`choose`instruction can be used to do conditional processing within the template.

```
<xsl:template match="BeaconCardinal">

        <xsl:choose>
        <xsl:when test="categoryOfCardinalMark = '2'">
                <!--  Output symbol for BeaconCardinal categoryOfCardinalMark =2 here -->
        </xsl:when>
        <xsl:when test="categoryOfCardinalMark = '3'">
                <!--  Output symbol for BeaconCardinal categoryOfCardinalMark =3 here -->
        </xsl:when>
        <xsl:otherwise>
                <!--  Output default symbol here -->
        </xsl:otherwise>
</xsl:choose>

</xsl:template>
```

A more advanced XPath expression can be used to refine the match.

```
<xsl:template match="BeaconCardinal[categoryOfCardinalMark=2]">
        <!—This is a comment.  Output symbol for BeaconCardinal categoryOfCardinalMark =2 here -->

</xsl:template>
```

9 Drawing Instructions

9.1 The concepts of drawing instructions

9.1.1 General concept

The output of the portrayal engine is a set of drawing instructions, which link the feature type to a symbol reference. The geometry is either taken from the feature type or can be generated by the portrayal functions. The latter is supported by the concept of augmented geometry.

9.1.2 Portrayal Coordinate Reference Systems

In this context coordinate reference systems refer to the portrayal geometry only.

There are different CRS related to the portrayal:

- Geographic CRS
- Portrayal CRS
- Local CRS
- Line CRS
- Area CRS
- Tile CRS
- Hatch CRS

Geographic CRS are used in the geographic dataset that are portrayed. They will be mapped by means of projections and affine transformation to the Portrayal CRS. Nevertheless rotations of symbols may be still defined relative to the North-Axis of the Geographic CRS.

The Portrayal CRS defines the coordinates at the output device e.g. a screen or pixmap.

Line symbols have two kinds of coordinate reference systems. The line coordinate reference system is a 1-axis coordinate system perpendicular to the geometry of the curve and allows for the specification of line widths and offsets. The second kind of coordinate reference system is a local coordinate reference system which is defined for every location along a curve. This coordinate reference system has an x-axis that is tangential to the curve and a y-axis perpendicular to the x-axis.

An area symbol defines coordinate reference systems for its boundary and for its interior. The boundary coordinate reference systems are those defined for line symbols. The interior of the area symbol has its own coordinate reference system.

For tiled pattern and hatch patterns own CRS are defined.

9.1.3 Viewing Groups and Display Mode

The viewing group is a concept to control the content of the display. It work as an on/off switch for any drawing instruction assigned to the corresponding viewing group. The concept can be seen as a filter on the list of drawing instructions.

Viewing groups can be aggregated into Display Modes.

9.1.4 Display Planes

Display planes are a concept to split the output of the portrayal functions into separate lists. An example of this is the separation of chart information drawn under a radar image and chart information drawn over a radar image.

9.1.5 Display Priorities

Display priorities control the order in which the output of the portrayal functions is processed by the rendering engine. Priorities with smaller numerical values will be processed first.

9.1.6 Null Instruction

This is an instruction to indicate that a feature is intentionally not portrayed.

9.1.7 Point Instruction

9.1.7.1 Overview

The Point Instruction defines the drawing of a symbol. The symbol can be parameterized. This includes rotation, scaling and offset. The details are described in the documentation of the Symbol package.

9.1.7.2 Point Geometry

When the Point Instruction references point geometry the symbol is drawn using its position.

9.1.7.3 Multi Point Geometry

The symbol is repeated using each position of the Multi Point.

9.1.7.4 Curve Geometry

The symbol is drawn on each referenced curved from either the spatialReference or if this is not used on each curve directly referenced by the feature type. The placement of the symbol is controlled by the linePlacement element of the symbol. The details are described in the documentation of the Symbol package.

9.1.7.5 Surface Geometry

The symbol is drawn at a representative position within the surface. How this position is obtained is controlled by the areaPlacement member of the symbol. The details are described in the documentation of the Symbol package.

9.1.8 Line Instruction

9.1.8.1 Overview

The Line Instruction defines the drawing of a line style. Line styles include Simple and Complex Line Styles. The line style can be parameterized. The details are described in the documentation of the

LineStyles package. The geometry is defined by the referenced spatial types. Only curve or surface geometry is supported. For the latter the boundary of the surface defines the geometry. The geometry defines the direction of drawing for the line style.

9.1.8.2 Suppression

When features shares curve geometry multiple line instructions may reference the same curve.

If suppression is set to true (the default) another line instruction with a higher display priority will suppress the drawing of this line instruction. If suppression is set to false this instruction cannot be suppressed.

9.1.9 Area Instruction

9.1.9.1 Overview

The Area Instruction defines the drawing of an area fill. Area Fills include Color Fills and different Pattern Fills. The area fill can be parameterized. The details are described in the documentation of the AreaFills package. Only surface geometry is supported.

9.1.10 Text Instruction

9.1.10.1 Overview

The Text Instruction defines the drawing of text. The text can be parameterized. This includes fonts, colour and size. The details are described in the documentation of the Text package.

9.1.10.2 Point Geometry

When the Text Instruction references a point geometry the text is drawn using its position. Only TextPoint elements are supported.

9.1.10.3 Multi Point Geometry

The text is repeated using each position of the Multi Point.Only TextPoint elements are supported.

9.1.10.4 Curve Geometry

The text is drawn on each referenced curved from either the spatialReference or if this is not used on each curve directly referenced by the feature type. Both TextPoint and TextLine elements are supported. The first is to draw text at a position on the referenced curve relative to the local CRS at that position. The latter is to draw text that follows the shape of the referenced curve. More details can be found in the documentation of the text package.

9.1.10.5 Surface Geometry

The text is drawn at a representative position within the surface. Only TextPoint elements are supported. How this position is obtained is controlled by the areaPlacement member of the TextPoint. The details are described in the documentation of the Text package.

9.1.11 Coverage Instruction

An instruction to portray data coverages like gridded bathymetry, satellite images, etc.

Further input is required from domain experts.

9.1.12 Augmented Geometry

9.1.12.1 Overview

In case the required geometry for a drawing instruction is not explicitly given in a geographic dataset the portrayal function will generate this “augmented” geometry. A set of classes for such augmented geometries are part of the model. All positions used in this classes refer to a given coordinate reference system. Three types of CRS are supported:

1. Geographic CRS
   The coordinates are geographic coordinates.
2. Portrayal CRS
   The coordinate are referenced to the output device of the portrayal.
3. Local CRS
   The coordinates referring to a coordinate system with axes parallel to the Portrayal CRS but the origin shifted to the position of the referenced feature. Only point feature are supported for that type of CRS.

Note: The generated geometry only exists temporary for the purpose of portrayal and is not part of the dataset.

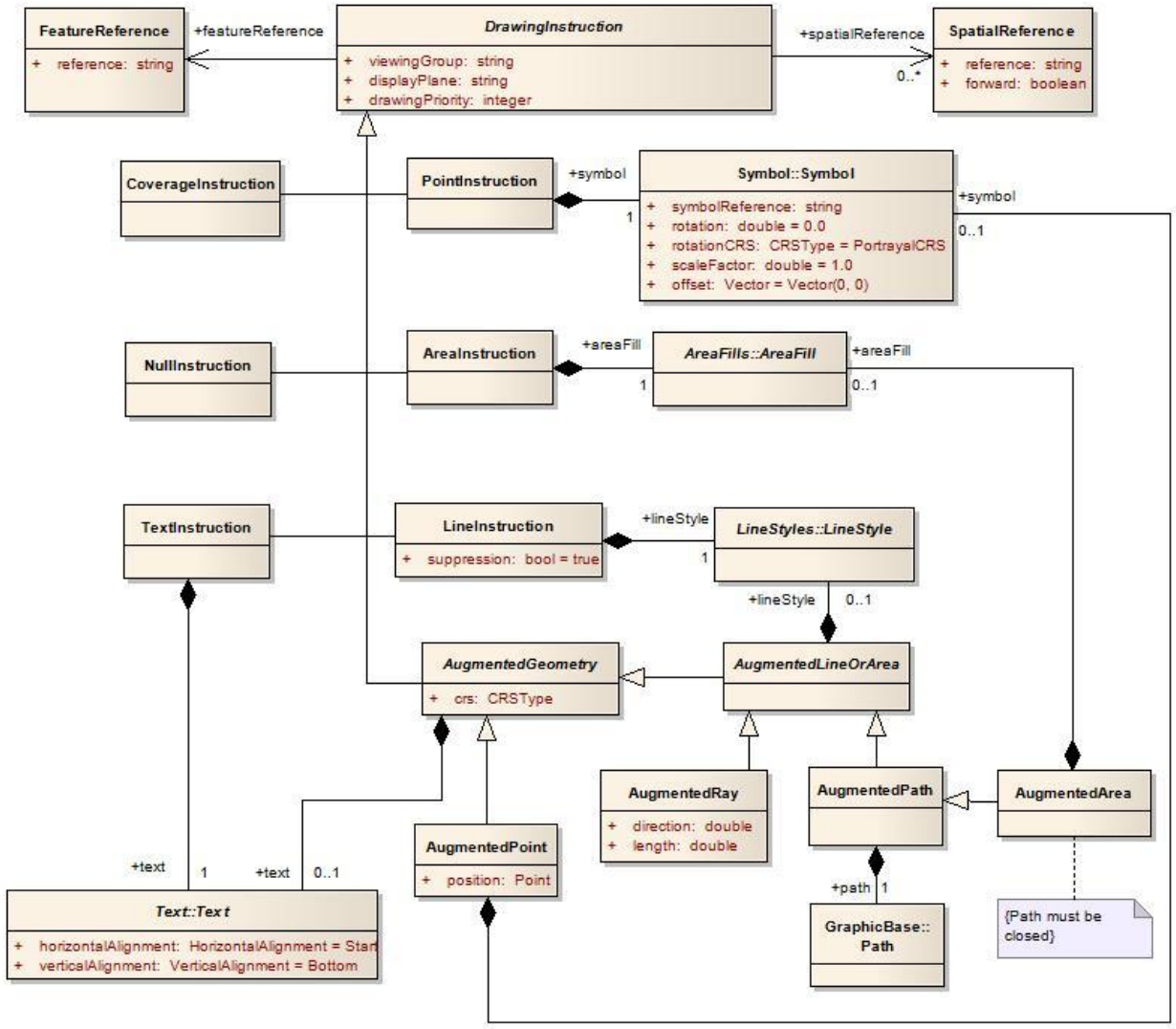
All types of augmented geometries can be used for the portrayal of text.

Examples

More details can be found in the documentation of the drawing instruction model.

9.2 Model of the Drawing Instruction Package

This package contains classes which describe the output of the portrayal functions. Display instructions link the feature types and their geometry to elements from the Symbol Elements package. The next diagram shows the model.



9.2.1 DrawingInstruction

Role Name	Name	Description	Mult.	Type
Class	DrawingInstruction	Abstract base class for all drawing instructions.	-	-
Attribute	viewingGroup	The viewing group the instruction is assigned to.	1	string
Attribute	displayPlane	The display plane the instruction is assigned to	1	string
Attribute	drawingPriority	The priority that defines the order of drawing.	1	integer

Role	featureReference	The reference to the feature type that will be depicted by the instruction.	1	FeatureReference
Role	spatialReference	The reference(s) to the spatial type components of the feature that defines the geometry used for the depiction.Not used when the entire geometry of the feature should be depicted.	0..*	SpatialReference

9.2.2 FeatureReference

Role Name	Name	Description	Mult.	Type
Class	FeatureReference	A reference to a feature type	-	-
Attribute	reference	The identifier of the feature type	1	string

9.2.3 SpatialReference

Role Name	Name	Description	Mult.	Type
Class	SpatialReference	A reference to a spatial type.	-	-
Attribute	reference	The identifier of the spatial type.	1	string
Attribute	forward	If true the spatial object is used in the direction in which it is stored in the data. Only applies to curves.	1	boolean

9.2.4 NullInstruction

Role Name	Name	Description	Mult.	Type
Class	NullInstruction	An instruction that indicates that no portrayal is required for the referenced feature.	-	-

9.2.5 PointInstruction

Role Name	Name	Description	Mult.	Type
Class	PointInstruction	A drawing instruction for point symbol.	-	-
Role	symbol	The symbol to be depicted.	1	Symbol::Symbol

9.2.6 LineInstruction

Role Name	Name	Description	Mult.	Type
Class	LineInstruction	A drawing instruction for line geometry.	-	-
Role	lineStyle	The line style used for the depiction.	1	LineStyles::LineStyle

9.2.7 AreaInstruction

Role Name	Name	Description	Mult.	Type
Class	AreaInstruction	A drawing instruction for area geometry.	-	-
Role	areaFill	The area fill used for the depiction.	1	AreaFills::AreaFill

9.2.8 TextInstruction

Role Name	Name	Description	Mult.	Type
Class	TextInstruction	A drawing instruction for depicting text.	-	-
Role	text	The text to be depicted.	1	Text::Text

9.2.9 CoverageInstruction

Role Name	Name	Description	Mult.	Type
Class	CoverageInstruction	A drawing instruction for depicting coverages of data.	-	-
Role	tbd.			

9.2.10 AugmentedGeometry

Role Name	Name	Description	Mult.	Type
Class	AugmentedGeometry	A base class for drawing instructions that uses geometry not available in the dataset. The geometry is generated by the portrayal functions according to a defined CRS.	-	-
Attribute	crs	The coordinate reference system of the generated geometry. One of • Geographic CRS • Portrayal CRS • Local CRS For detailed description see the documentation of the GraphicsBase package.	1	GraphicBase::CRSType
Role	text	A text to be depicted by the instruction. The rules for text apply depending on the type of geometry used by the instruction.	0..1	Text::Text

9.2.11 AugmentedPoint

Role Name	Name	Description	Mult.	Type
Class	AugmentedPoint	A drawing instruction for a point symbol where the position is not given by the feature type.	-	-
Attribute	position	The position of the symbol.	1	GraphicBase::Point
Role	symbol	The symbol to be depicted.	0..1	Symbol::Symbol

9.2.12 AugmentedLineOrArea

Role Name	Name	Description	Mult.	Type
Class	AugmentedLineOrArea	A base class for linear augmented geometry.	-	-
Role	lineStyle	The line style to be depicted.	0..1	LineStyles::LineStyle

9.2.13 AugmentedRay

Role Name	Name	Description	Mult.	Type
Class	AugmentedRay	A drawing instruction that defines a line from the position of a point feature to another position.	-	-

		The position is defined by the direction and the length attributes. It can be used for drawing line styles or line texts.		
Attribute	direction	The direction of the ray relative to the used CRS.	1	double
Attribute	length	The length of the ray. The units depending on the used CRS.	1	double

9.2.14 AugmentedPath

Role Name	Name	Description	Mult.	Type
Class	AugmentedPath	A drawing instruction for a line. It can be used for drawing line styles or line texts.	-	-
Role	path	The path defining the line geometry.	1	GraphicsBase::Path

9.2.15 AugmentedArea

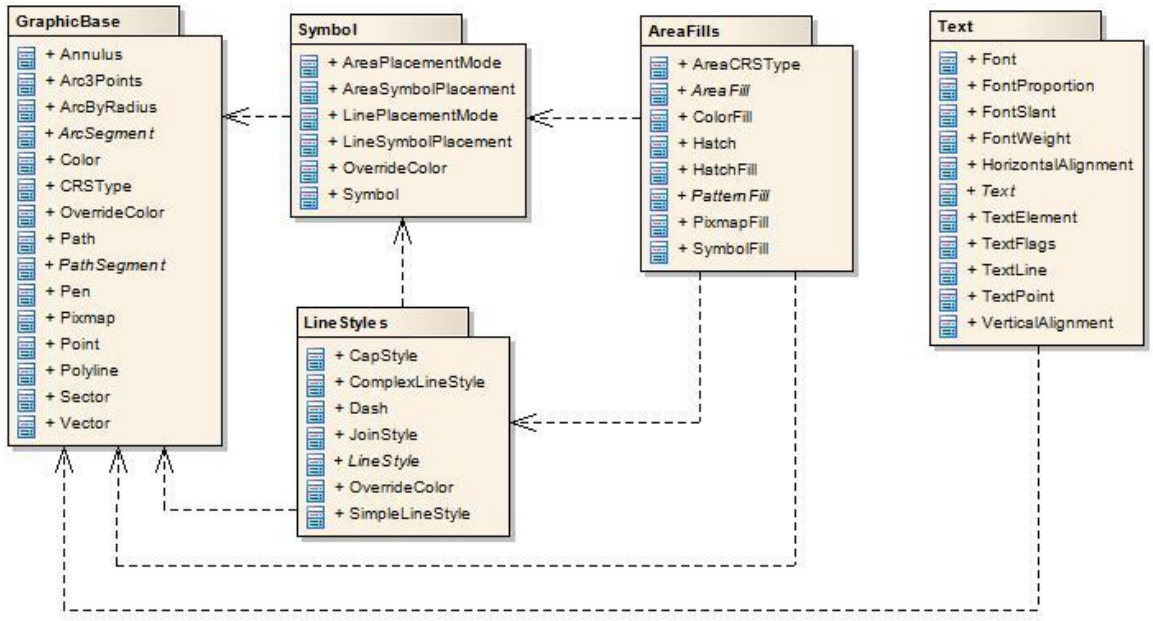
Role Name	Name	Description	Mult.	Type
Class	AugmentedArea	A drawing instruction for an area. It can be used for drawing line styles, area fills, or area texts. The used path must be closed.	-	-
Role	areaFill	The area fill to be depicted.	0..1	AreaFills::AreaFill

For schema definition see A.3 Presentation Schema

10 Symbol Definitions

10.1 Overview

The SymbolDefinition package describes the graphic primitives used for the portrayal. Parts of the primitives are defined externally by using SVG definitions. Those external parts will be referenced from the types in this model. The package diagram is shown in the following figure.

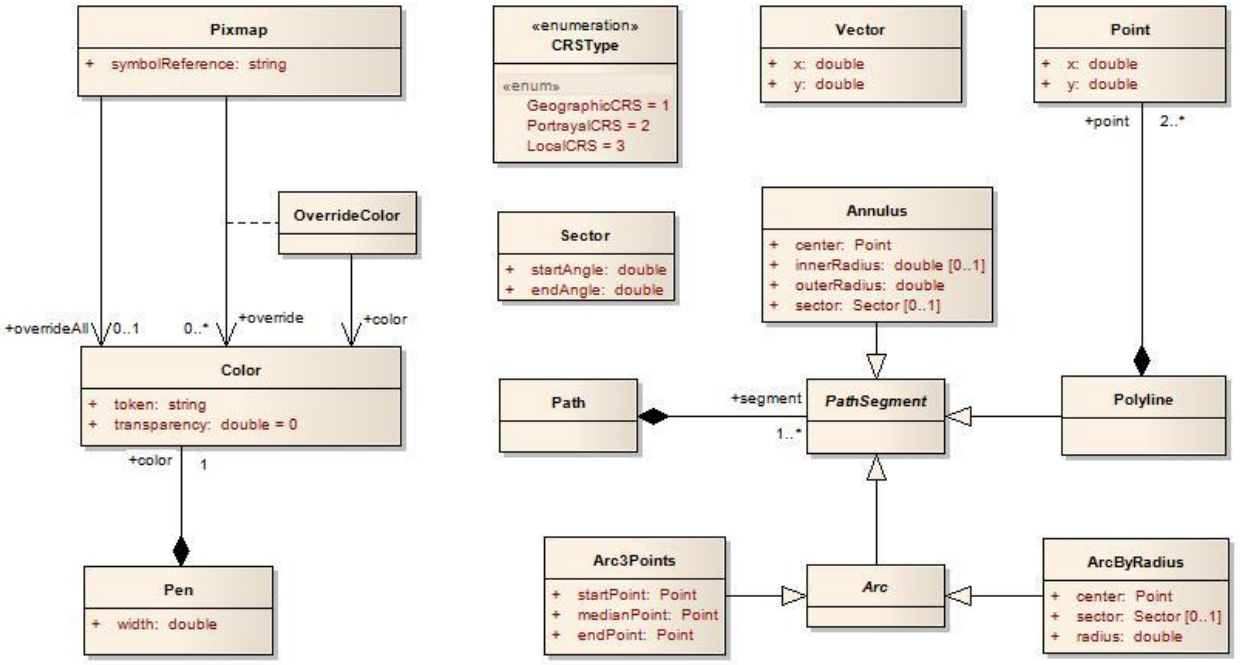


10.2 The GraphicBase package

10.2.1 Overview

This package contains graphic base types for the use in other packages.

10.2.2 Model



10.2.2.1 Point

Role Name	Name	Description	Mult.	Type
Class	Point	A zero-dimensional geometric object in a two-dimensional coordinate space. The coordinate will refer to a coordinate reference system.	-	-
Attribute	x	The x-coordinate of the point. In case the CRS is a geographic CRS this refers to the longitude.	1	double
Attribute	y	The y-coordinate of the point. In case the CRS is a geographic CRS this refers to the latitude.	1	double

10.2.2.2 Vector

Role Name	Name	Description	Mult.	Type
Class	Vector	A geometric object that has both a magnitude and a direction. It is limited to Cartesian coordinate reference systems.	-	-
Attribute	x	The x-coordinate of the vector.	1	double
Attribute	y	The y-coordinate of the vector.	1	double

10.2.2.3 Color

Role Name	Name	Description	Mult.	Type
Class	Color	Representing a colour according to the colour	-	-

		model		
Attribute	token	The token specifies either an element in a colour table or a colour definition in the RGB space.	1	string
Attribute	transparency	The value specifies the transparency; between 0 (opaque) and 1 (full transparent)	1	double

10.2.2.4 Pen

Role Name	Name	Description	Mult.	Type
Class	Pen	A tool for drawing lines.	-	-
Attribute	width	The width of the pen in mm.	1	Double
Role	color	The colour of the pen comprises the actual colour and the transparency.	1	Color

10.2.2.5 Pixmap

Role Name	Name	Description	Mult.	Type
Class	Pixmap	A two dimensional array of pixels defining an image.	-	-
Attribute	symbolReference	A reference to an external definition of the pixmap.	1	string
Role	overrideAll	A colour that override all none fully transparent colours used within the pixmap.	0..1	Color
Association	override	A colour to be replaced by another colour.	0..*	OverrideColor

10.2.2.6 OverrideColor

Role Name	Name	Description	Mult.	Type
Class	OverrideColor	Association class for the replacement of an existing colour in the pixmap with another colour.	-	-
Role	color	The colour that is used to replace the existing colour in the pixmap.	1	Color

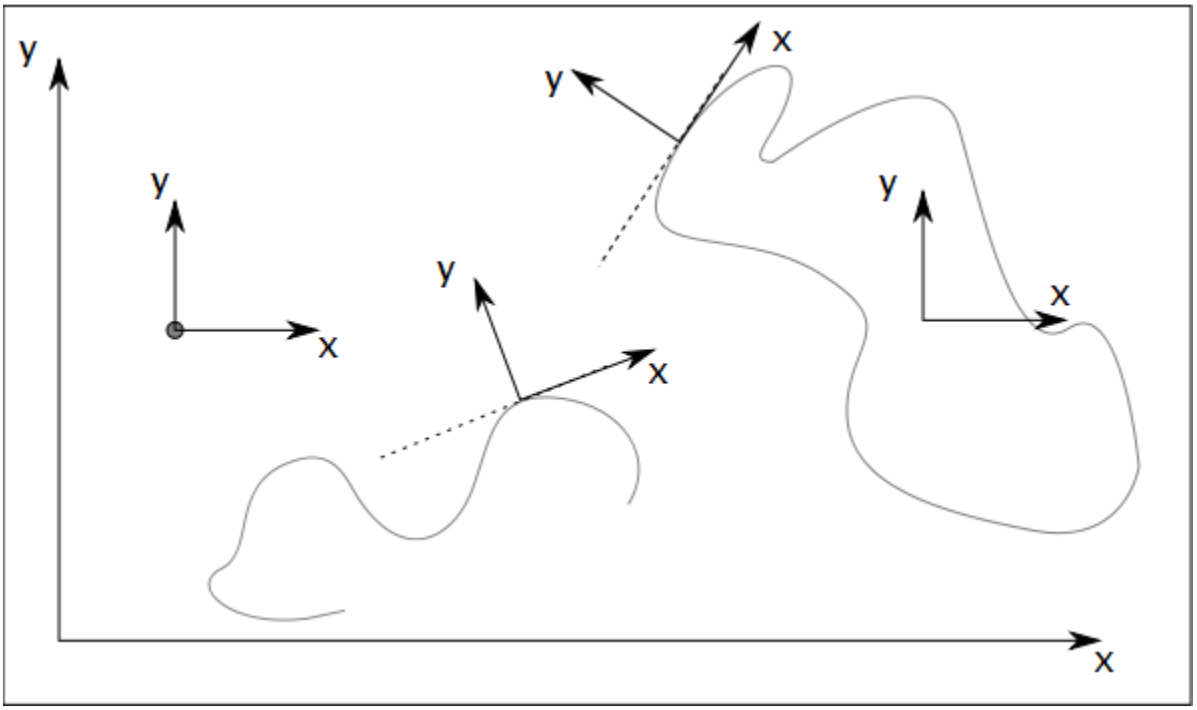
10.2.2.7 CRSType

Role Name	Name	Description
Type	CRSType	The value describes the type of a CRS. This includes the axes definitions, base line for angle measurement and units for distances.
Enumeration	GeographicCRS	A geographic CRS with axis latitude and longitude measured in degrees. Angles are defined clockwise from the true north direction. Distances will be measured in metres.
Enumeration	PortrayalCRS	A Cartesian coordinate system with the y-axis pointing upwards. Units on the axes and for distances are millimetres. Angles are measured in degrees anticlockwise from the x-axis. Note that the actual output device may have a different orientation of the y-axis.
Enumeration	LocalCRS	A Cartesian coordinate system originated at a local geometry. See explanations for details.

The following figure shows how the local CRS are defined for the different types of geometry.

From left to right:

- Local CRS for point geometry
  Note: for multi points the local CRS is repeated at each point.
- Local CRS for curve geometry. The origin of the coordinate system can be any point of the line. This point can be defined by the absolute or relative distance from the start of the line. The x-axis is directed in the direction of the tangent at the tangency point and the y-axis is oriented perpendicular to this direction.
- Local CRS for surface geometry. For the boundary the same rules apply as for curve geometry. For the interior of the surface a coordinate system is used that has axes parallel to the Portrayal CRS. The origin can be an arbitrary point that is constant relative to the surface. This point can be outside the surface.



10.2.2.8 Sector

Role Name	Name	Description	Mult.	Type
Class	Sector	Region of the Cartesian plane enclosed by two radii.	-	-
Attribute	startAngle	The direction of the radius that defines the beginning of the sector.	1	double
Attribute	endAngle	The direction of the radius that defines the end of the sector.	1	double

10.2.2.9 Path

Role Name	Name	Description	Mult.	Type

Class	Path	The definition of linear geometry by a composition of segments.	-	-
Role	segment	The segments that build up the path.	1..*	PathSegment

Paths can be closed or not closed. A closed path has coinciding start and end points. Segments are connected until a path is closed. In that case the next segment is not connected and the path contains multiple sub-paths.

10.2.2.10 PathSegment

Role Name	Name	Description	Mult.	Type
Class	PathSegment	Abstract base class for all segments that can be used within a path.	-	-

10.2.2.11 Polyline

Role Name	Name	Description	Mult.	Type
Class	Polyline	A segment defining its geometry by a series of points.	-	-
Role	point	The segments the build up the path.	2..*	Point

10.2.2.12 Arc

Role Name	Name	Description	Mult.	Type
Class	Arc	Abstract base class for segments describing arcs of a circle.	-	-

10.2.2.13 Arc3Points

Role Name	Name	Description	Mult.	Type
Class	Arc3Points	A segment describing an arc of a circle that is defined by 3 points. The points must not be colinear.	-	-
Attribute	startPoint	The point where the arc starts.	1	Point
Attribute	medianPoint	An arbitrary point on the arc.	1	Point
Attribute	endPoint	The point where the arc ends.	1	Point

10.2.2.14 ArcByRadius

Role Name	Name	Description	Mult.	Type
Class	ArcByRadius	A segment describing an arc of a circle that is defined by the centre of the arc and a radius. Optional the arc can be restricted by a sector.	-	-
Attribute	center	The centre of the arc.	1	Point
Attribute	sector	The sector defining where the arc starts and end. If not present the arc is a full circle.	0..1	Sector
Attribute	radius	The radius of the circle.	1	double

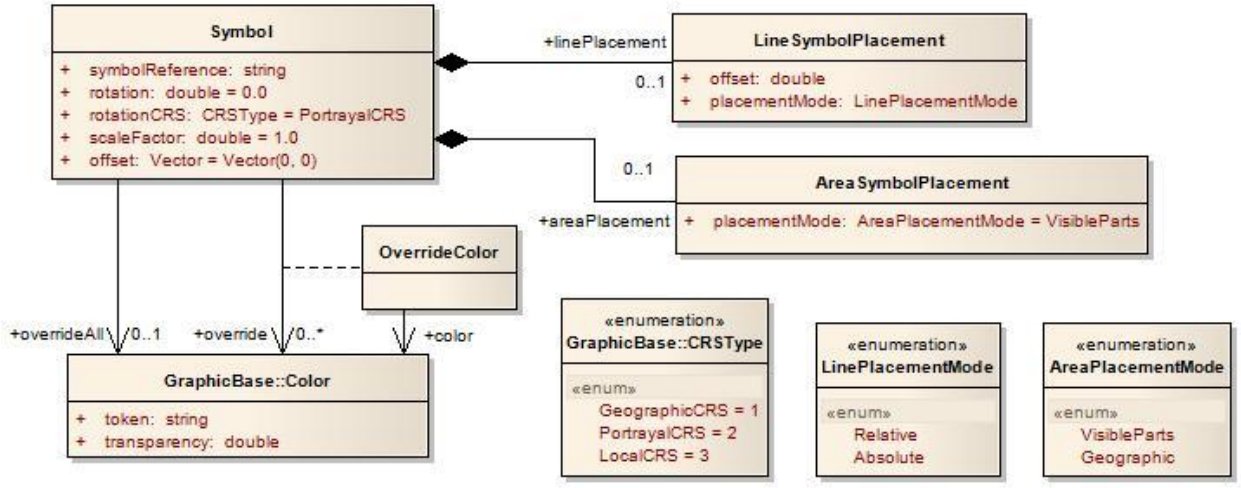
10.2.2.15 Annulus

Role Name	Name	Description	Mult.	Type
Class	Annulus	A ring-shaped region bounded by two concentric circles. Optional it can be enclosed by two radii of the circle	-	-
Attribute	center	The centre of the arc.	1	Point
Attribute	innerRadius	The radius of the smaller circle. If not present the segment describes a sector of a circle.	0..1	double
Attribute	outerRadius	The radius of the larger circle.	1	double
Attribute	sector	The sector of an annulus segment	0..1	Sector

10.3 The Symbol package

10.3.1 Model

This package contains the model of a symbol. Note that the definition of the symbol graphic itself is not the subject of this model. This will be defined in external files according to the SVG 1.1 recommendation.



10.3.1.1 Symbol

Role Name	Name	Description	Mult.	Type
Class	Symbol	A two dimensional graphical element.	-	-
Attribute	symbolReference	A reference to an external definition of the symbol graphic.	1	string
Attribute	rotation	The rotation angle of the symbol. The default value is 0.	1	double
Attribute	rotationCRS	Specifies the coordinate reference system for the rotation.	1	GraphicsBase::CRSType
Attribute	scaleFactor	The factor by which the original symbol graphic is scaled. The default value is 1.	1	double
Attribute	offset	The shift of the symbols position from the position of the geometry. The default value is the vector with length equals to 0.	1	GraphicsBase::Vector
Role	overrideAll	A colour that override all none fully transparent colours used within the symbol.	0..1	GraphicsBase::Color

Association	override	A colour to be replaced by another colour.	0..*	OverrideColor
Role	linePlacement	Information where on a line the symbol should be placed.	0..1	LineSymbolPlacement
Role	areaPlacement	Defines the placement of a symbol within an area.	0..1	AreaSymbolPlacement

10.3.1.2 OverrideColor

Role Name	Name	Description	Mult.	Type
Class	OverrideColor	Association class for the replacement of an existing colour in the symbol.	-	-
Role	color	The colour that is used to replace an existing colour in the symbol.	1	Color

10.3.1.3 LineSymbolPlacement

Role Name	Name	Description	Mult.	Type
Class	LineSymbolPlacement	Defines the placement of a symbol along a line	-	-
Attribute	offset	The offset from the start of the curve.	1	double
Attribute	placementMode	The mode that defines how the offset is to be interpreted.	1	LinePlacementMode

10.3.1.4 AreaSymbolPlacement

Role Name	Name	Description	Mult.	Type
Class	AreaSymbolPlacement	Defines the placement of a symbol within an area.	-	-
Attribute	placementMode	The mode that defines how the symbol has to placed.	1	AreaPlacementMode

10.3.1.5 LinePlacementMode

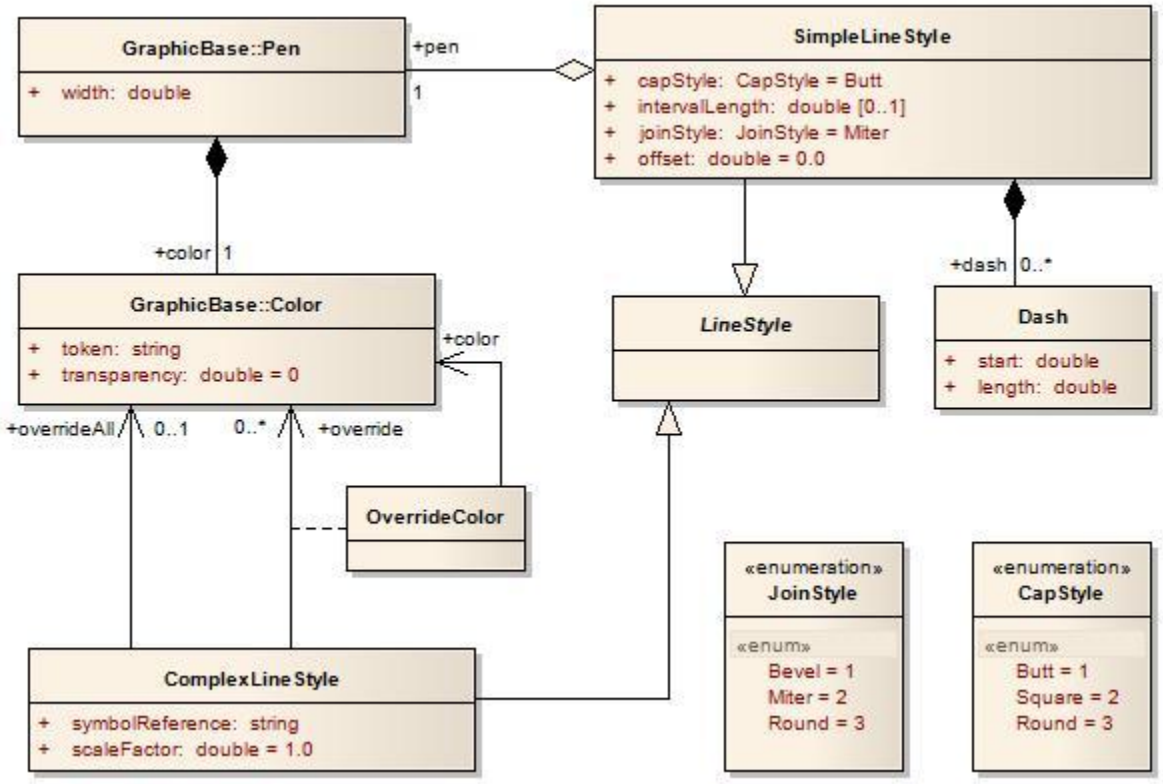
Role Name	Name	Description
Type	LinePlacementMode	Defines the type of placement of a symbol along a line.
Enumeration	Relative	The offset has to be interpreted as homogenous coordinates, 0 for the start and 1 for the end of the curve.
Enumeration	Absolute	The offset is the distance from the start of the curve.

10.3.1.6 AreaPlacementMode

Role Name	Name	Description
Type	AreaPlacementMode	Defines the type of placement of a symbol within an area.
Enumeration	VisibleParts	The symbol has to be placed at a representative position in each visible part of the surface.
Enumeration	Geographic	The symbol has to be placed at a representative position of the geographic object.

10.4 The Line Styles package

10.4.1 Model



10.4.1.1 LineStyle

Role Name	Name	Description	Mult.	Type
Class	LineStyle	Abstract base class for graphics to depict line geometry.	-	-

10.4.1.2 SimpleLineStyle

Role Name	Name	Description	Mult.	Type
Class	SimpleLineStyle	A style for line geometry either solid or dashed.	-	-
Attribute	offset	An offset perpendicular to the direction of the line. Positive offsets are at the right side of the line. The unit is mm.	1	double
Attribute	capStyle	The decoration that is applied where a line segment ends.	1	CapStyle
Attribute	joinStyle	The decoration that is applied where two line segments meet.	1	JoinStyle
Attribute	intervalLength	The length of a repeating interval of the line style in mm. If not defined the line style describes a solid line	0..1	double
Role	dash	The dashes of a dashed line style.	0..*	Dash
Role	pen	The pen used for drawing the line.	1	Pen

10.4.1.3 Dash

Role Name	Name	Description	Mult.	Type
Class	Dash	A single dash in a repeating line pattern.	-	-
Attribute	start	The start of the dash measured from the start of the repeating interval. The unit is mm.	1	double
Attribute	length	The length of the dash in mm.	1	double

10.4.1.4 ComplexLineStyle

Role Name	Name	Description	Mult.	Type
Class	ComplexLineStyle	Defining a line style with a repeating pattern. It may contain symbols.		
Attribute	symbolReference	A reference to an external definition of the symbol graphic.	1	string
Attribute	scaleFactor	The factor by which the original symbol graphic is scaled. Default = 1.0.	1	double
Role	overrideAll	A colour that override all none fully transparent colours used within the complex line style.	0..1	Color
Association	override	A colour to be replaced.	0..*	OverrideColor

10.4.1.5 OverrideColor

Role Name	Name	Description	Mult.	Type
Class	OverrideColor	Association class for the replacement of an existing colour in the complex line style.	-	-
Role	color	The colour that is used to replace an existing colour in the complex line style.	1	Color

10.4.1.6 JoinStyle

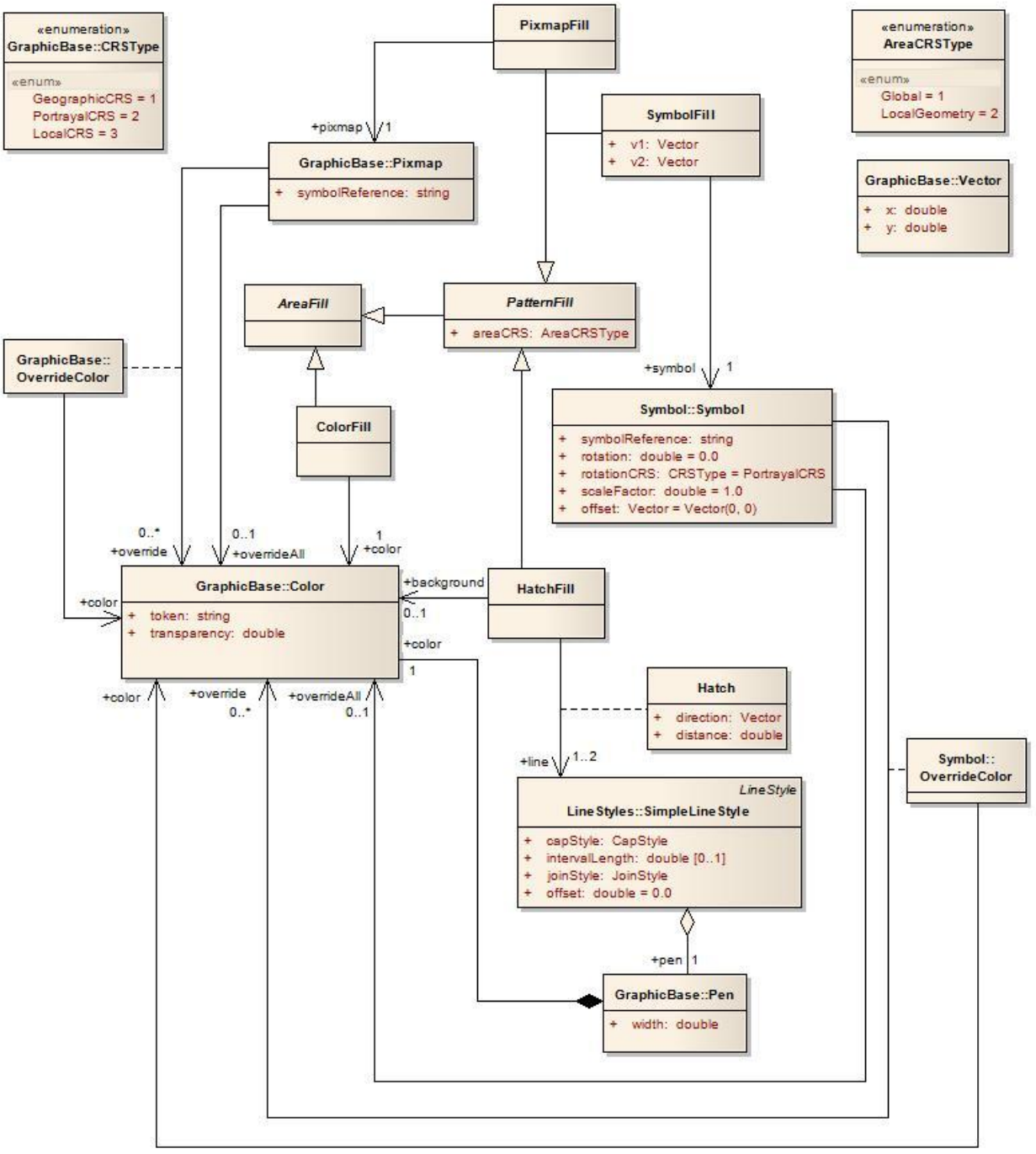
Role Name	Name	Description
Type	JoinStyle	The decoration that is applied where two line segments meet.
Enumeration	Bevel	
Enumeration	Miter	
Enumeration	Round	

10.4.1.7 CapStyle

Role Name	Name	Description
Type	CapStyle	The decoration that is applied where a line segment ends.
Enumeration	Butt	
Enumeration	Square	
Enumeration	Round	

10.5 The AreaFills package

10.5.1 Model



10.5.1.1 AreaFill

Role Name	Name	Description	Mult.	Type
Class	AreaFill	Abstract base class for graphics that are designed to fill an area.	-	-

10.5.1.2 PatternFill

Role Name	Name	Description	Mult.	Type
Class	PatternFill	Abstract base class for pattern area fills.	-	-
Attribute	areaCRS	Coordinate reference system which defines the origin of the pattern.	1	AreaCRSType

10.5.1.3 ColorFill

Role Name	Name	Description	Mult.	Type
Class	ColorFill	Class defining a solid colour fill for an area.	-	-
Role	color	References the colour and transparency for the colour fill.	1	Color

10.5.1.4 PixmapFill

Role Name	Name	Description	Mult.	Type
Class	PixmapFill	Pattern fill where the pattern is defined by a pixmap.	-	-
Role	pixmap	The pixmap defining the pattern.	1	Pixmap

10.5.1.5 SymbolFill

Role Name	Name	Description	Mult.	Type
Class	SymbolFill	Pattern fill where the pattern is defined by repeated symbols.	-	-
Role	symbol	The symbol used for the pattern.	1	Symbol
Attribute	v1	Defines the offset of the next symbol in the first dimension of the pattern.	1	Vector
Attribute	v2	Defines the offset of the next symbol in the second dimension of the pattern.	1	Vector

10.5.1.6 HatchFill

Role Name	Name	Description	Mult.	Type
Class	HatchFill	Defining a pattern made of one or two sets of parallel lines.	-	-
Role	background	The line style used for the lines.	0..1	GraphicBase::Color
Association	Hatch	A set of parallel lines.	1	Hatch

10.5.1.7 Hatch

Role Name	Name	Description	Mult.	Type
Class	Hatch	A set of parallel lines used for an area fill pattern.		
Attribute	direction	The vector defining the direction of the set of lines.	1	Vector
Attribute	distance	The distance between the lines measured perpendicular to the direction.	1	double
Role	line	The line style used for each hatch line	1..2	LineStyles::SimpleLine Style

10.5.1.8 AreaCRSType

Role Name	Name	Description
Type	PatternCRS	Describes how a fill patter is referenced.
Enumeration	Device	The anchor point of the fill pattern is in the coordinate system of the drawing device.
Enumeration	LocalGeometry	The anchor point of the fill pattern is in the coordinate system of the spatial object to be depicted.

10.6 The Text package

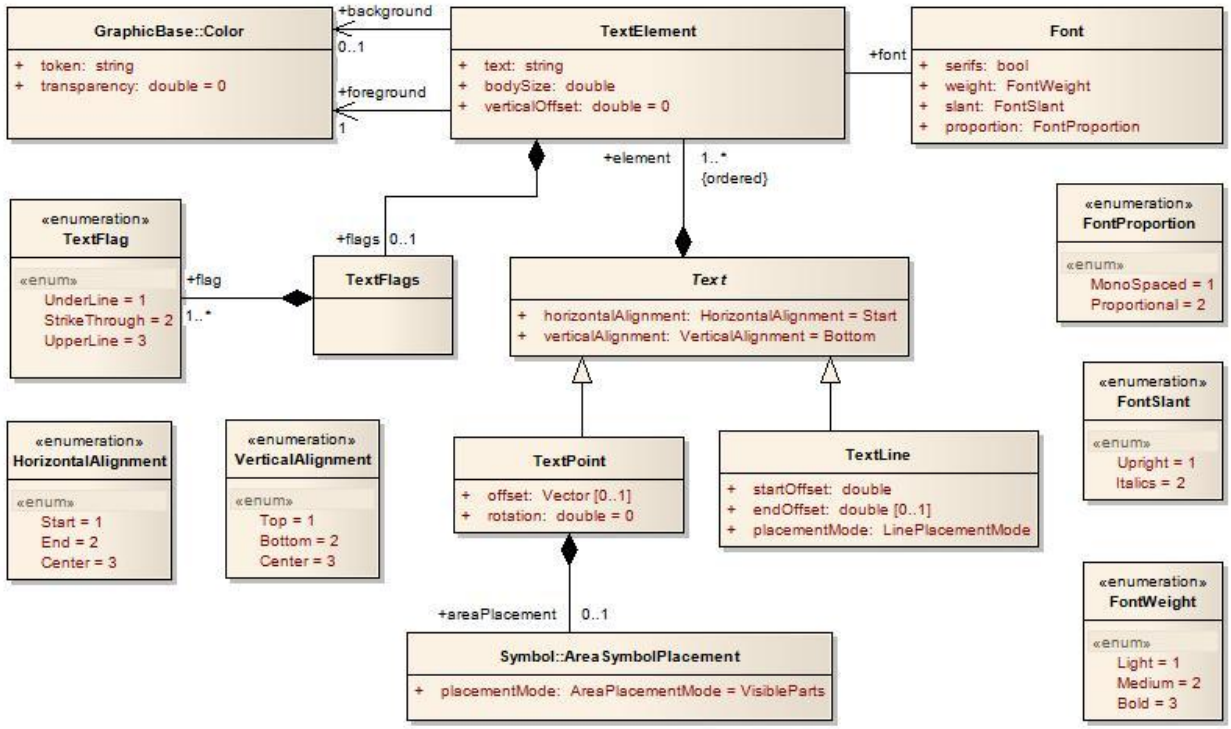
10.6.1 Overview

The text package contains the types necessary for the depiction of text. This includes fonts. In this model fonts are described by properties. Which system font will be used for rendering the text is not defined here. The selection of a system font is in the responsibility of the rendering software.

Two types of text instructions are supported:

- Text relative to a point
- Text that will be drawn along a linear geometry

10.6.2 Model



10.6.2.1 Font

Role Name	Name	Description	Mult.	Type
Class	Font	A font is a set of typefaces. A typeface is the artistic representation or interpretation of characters; it is the way the type looks. In this model the font is described by four attributes. The match to an actual font of the graphic system is part of the implementation.	-	-
Attribute	serifs	Describes whether the typefaces contain serifs or not.	1	bool
Attribute	weight	Describes the thickness of the typefaces.	1	FontWeight
Attribute	slant	Describes the slant of the typefaces.	1	FontSlant
Attribute	proportion	Describes whether all typefaces in the font have an individual width or a fixed width.	1	FontProportion

10.6.2.2 Text

Role Name	Name	Description	Mult.	Type
Class	Text	The abstract base class of graphic elements for depicting text. The text is composed of elements.	-	-
Attribute	horizontalAlignment	Specifies how the text is horizontally aligned relative to the anchor point. Default = Start.	1	HorizontalAlignment
Attribute	verticalAlignment	Specifies how the text is vertically aligned relative to the anchor point. Default = Bottom	1	VerticalAlignment
Role	element	The ordered list of text elements.	1..*	TextElement

10.6.2.3 TextPoint

Role Name	Name	Description	Mult.	Type
Class	TextPoint	A graphic element for depicting text relative to a point.	-	-
Attribute	offset	Specifies the offset from the anchor point with respect to the portrayal CRS.	0..1	GraphicsBase::Vector
Attribute	rotation	Specifies the rotation angle relative to the portrayal CRS. Default = 0.	1	double
Role	areaPlacement	Describes the placement of the text when the geometry is a surface.	0..1	Symbol:: AreaSymbolPlacement

10.6.2.4 TextLine

Role Name	Name	Description	Mult.	Type
Class	TextLine	A graphic element for depicting text along linear geometry.	-	-
Attribute	startOffset	This offset specifies the anchor point on the line.	1	double
Attribute	endOffset	This offset specifies the stop point of the text at the line. If present the startOffset does not specifies an anchor point but the start point of the text. The text will evenly be spaced between the two positions. Horizontal alignment has no effect in this case.	0..1	double
Attribute	placementMode	Specifies how the offsets have to be interpreted.	1	Symbol:: LinePlacementMode

10.6.2.5 TextFlags

Role Name	Name	Description	Mult.	Type
Class	TextFlags	A container for text flags	-	-
Role	flag	A text flag.	1..*	TextFlag

10.6.2.6 TextElement

Role Name	Name	Description	Mult.	Type
Class	TextElement	A sub element of a graphic text.	-	-
Attribute	text	The text to be depicted	1	String
Attribute	bodySize	This property describes the size with which the text will be depicted.	1	Double
Attribute	verticalOffset	The vertical offset in mm between the base line of the text element and the base line of the text. This can be used to generate sub- or superscripts. Default = 0.	1	Double
Role	flags	Flags describe special properties of the text element like underline etc.	0..1	TextFlags
Role	font	The font used for the depiction of the text element.	1	Font
Role	foreground	The colour used to depict the glyphs.	1	Color
Role	background	The colour to fill the rectangle surrounding the text element before the text is depicted. If not given there is no fill (transparent)	0..1	Color

10.6.2.7 FontSlant

Role Name	Name	Description
Type	FontSlant	The slant used within a font
Enumeration	Upright	Typefaces are upright.
Enumeration	Italics	Typefaces are cursive.

10.6.2.8 FontWeight

Role Name	Name	Description
Type	FontWeigth	The thickness used for the typefaces in a font.
Enumeration	Light	Typefaces are depicted as thin (standard)
Enumeration	Medium	Typefaces are depicted thicker as 'Light' but not as thin as 'Bold'
Enumeration	Bold	Typefaces are depicted more prominent (**Bold**)

10.6.2.9 FontProportion

Role Name	Name	Description
Type	FontProportion	The values describe how the width of the typefaces in a font is defined.
Enumeration	MonoSpaces	All typefaces in a font have the same width, also known as 'typewriter' fonts.
Enumeration	Proportional	Any typeface in the font as its individual width.

10.6.2.10 TextFlag

Role Name	Name	Description
Type	TextFlag	The values describe some effects used when the text will be depicted. The values can be combined.
Enumeration	UnderLine	Text is depicted with a line under the text.
Enumeration	StrikeThrough	Text is depicted struck through, a line goes through the text.
Enumeration	UpperLine	Text is depicted with a line above the text

10.6.2.11 VerticalAlignment

Role Name	Name	Description
Type	VerticalAlignment	Describes the text placement relative to the anchor point in vertical direction.
Enumeration	Top	The anchor point is at the top of the text.
Enumeration	Bottom	The anchor point is at the bottom of the text.
Enumeration	Center	The anchor point is at the (vertical) centre of the text.

10.6.2.12 HorizontalAlignment

Role Name	Name	Description
Type	HorizontalAlignment	Describes the text placement relative to the anchor point in horizontal direction.
Enumeration	Start	The anchor point is at the start of the text.
Enumeration	End	The anchor point is at the end of the text.
Enumeration	Center	The anchor point is at the (horizontal) centre of the text.

For schema definition see A.2 Symbol Definition Schema

11 The portrayal library

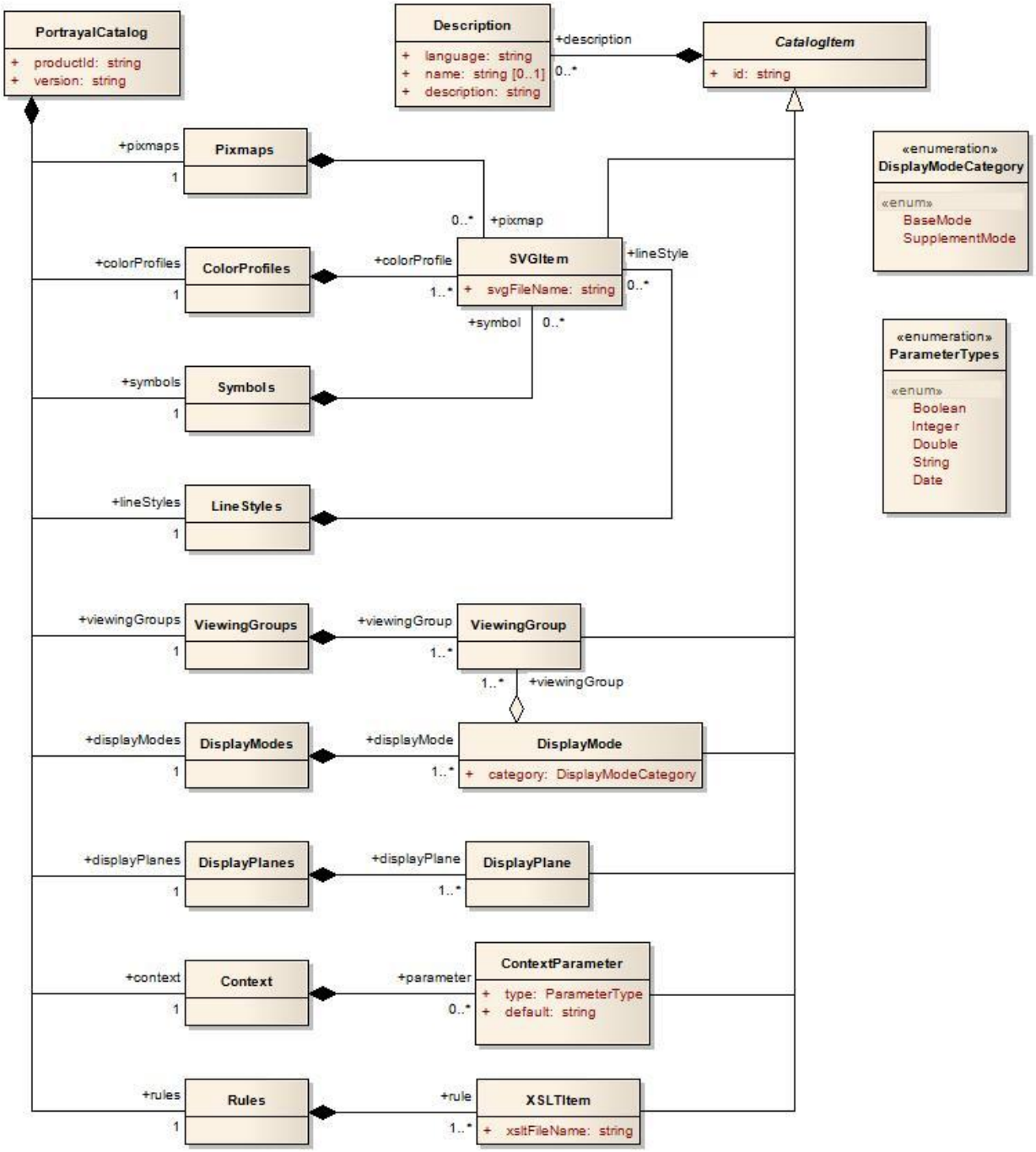
11.1 Overview

- Machine readable
- A file/directory structure with an catalogue file
- SVG files for pixmaps, symbols, complex line styles and colour profiles
- XSLT templates in separate files
- Model and schema for the catalogue included

11.2 Structure

```
Root ----  (contains the catalogue named "portrayal_catalogue.xml")
   |
   |-- Pixmaps  (contains SVG files describing pixmaps)
   |
   |-- ColorProfiles  (contains SVG files with colour profiles and CSS2 style sheets)
   |
   |-- Symbols  (contains SVG files with symbols)
   |
   |-- LineStyles(contains SVG files with line styles)
   |
   |-- Rules(contains XSLT files with templates, main rule named "main.xsl")
```

11.3 Model of the Catalogue



Role Name	Name	Description	Mult.	Type
Class	PortrayalCatalog	A container of all the Catalogue items	-	-
Attribute	productId	The ID of the product for which the Catalogue is	1	string

		intended.		
Attribute	version	The version of the product the Catalogue is defined for.	1	string
Role	pixmaps	Container of SVG Pixmap file references	1	Pixmap
Role	colorProfiles	Container of SVG Colour Profile file references	1	ColorProfiles
Role	symbols	Container of SVG Symbol file references	1	Symbols
Role	lineStyles	Container of SVG Line Style file references	1	LineStyles
Role	viewingGroups	Container of Viewing group definitions	1	ViewingGroups
Role	displayModes	Container of Display Mode definitions	1	DisplayModes
Role	displayPlanes	Container of Display Plane definitions	1	DisplayPlanes
Role	context	Container of Context parameter definitions	1	Context
Role	rules	Container of XSLT rule file references (The entry point must be called “main.xsl”)	1	Rules

Role Name	Name	Description	Mult.	Type
Class	CatalogItem	An abstract base class for components of the Catalogue	-	-
Attribute	id	A unique identifier of the catalogue item	1	string
Role	description	Meta Data common to each Catalogue Item. There can be descriptions in different languages.	0..*	Description

Role Name	Name	Description	Mult.	Type
Class	Description	Language specific information about an item	-	-
Attribute	language	A language identifier code. ISO 639-2/T alpha-3 code (eng – English, fra – French, deu - German)	1	string
Attribute	name	An optional name of an item in the identified language	0..1	string
Attribute	description	The language specific description of the item	1	string

Role Name	Name	Description	Mult.	Type
Class	SVGItem	A reference to an SVG file included in the Catalogue	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	svgFileName	The name of the SVG file being referenced	1	string

Role Name	Name	Description	Mult.	Type
Class	Pixmaps	A container of Pixmap file references	-	-
Role	pixmap	Reference to an SVG file containing a pixmap definition.	0..*	SVGItem

Role Name	Name	Description	Mult.	Type
Class	ColorProfiles	A container of Colour profile file references	-	-
Role	colorProfile	Reference to an SVG file containing a colour profile.	1..*	SVGItem

Role Name	Name	Description	Mult.	Type
Class	Symbols	A container of Symbol file references	-	-
Role	symbol	Reference to an SVG file containing a symbol definition.	0..*	SVGItem

Role Name	Name	Description	Mult.	Type
Class	LineStyles	A container of Line Style file references	-	-
Role	lineStyle	Reference to an SVG file containing a Line Style definition.	0..*	SVGItem

Role Name	Name	Description	Mult.	Type
Class	ViewingGroups	A container of Viewing Group definitions	-	-
Role	viewingGroup	Definition of a specific Viewing Group	1..*	ViewingGroup

Role Name	Name	Description	Mult.	Type
Class	ViewingGroup	A Viewing Group name and definition	-	-
Subtype of	CatalogItem	See CatalogItem	-	-

Role Name	Name	Description	Mult.	Type
Class	DisplayModes	A container of Display Mode definitions	-	-
Role	displayMode	Definition of a Display Mode	1..*	DisplayMode

Role Name	Name	Description	Mult.	Type
Class	DisplayMode	A Display Mode name and definition	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	category	The category of the Display Mode	1	DisplayModeCategory
Role	viewingGroup	Viewing Group included in this Category	1..*	ViewingGroup

Role Name	Name	Description
Type	DisplayModeCategory	Choice of Display Mode Categories
Enumeration	BaseMode	Part of Display Base
Enumeration	SupplementalMode	A supplemental Category

Role Name	Name	Description	Mult.	Type
Class	DisplayPlanes	A container of Display Plane definitions	-	-
Role	displayPlane	Definition of a Display Plane.	1..*	DisplayPlane

Role Name	Name	Description	Mult.	Type
Class	DisplayPlane	A Display Plane name and definition	-	-

Subtype of	CatalogItem	See CatalogItem	-	-

Role Name	Name	Description	Mult.	Type
Class	Context	A container of Context Parameters	-	-
Role	parameter	Context Parameter	0..*	ContextParameter

Role Name	Name	Description	Mult.	Type
Class	ContextParameter	A Context Parameter name and definition	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	type	The data type of the Parameter	1	ParameterType
Attribute	default	A default value for the Parameter	1	string

Role Name	Name	Description
Type	ParameterTypes	Choice of Parameter Types
Enumeration	Boolean	
Enumeration	Integer	
Enumeration	Double	
Enumeration	String	
Enumeration	Date	

Role Name	Name	Description	Mult.	Type
Class	Rules	A container of XSLT rule file references	-	-
Role	rule	Reference to an XSLT file containing template rules. (first file called “main.xsl”)	1..*	XSLTItem

Role Name	Name	Description	Mult.	Type
Class	XSLTItem	XSLT rule file reference	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	xsltFileName	Reference to an XSLT file containing template rules.	1	string

For schema definition see A.4 Portrayal Catalogue Schema

Annex A – XML Schemas(normative)

A.1 Input Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.iho.int/S100BaseModel"
   xmlns="http://www.iho.int/S100BaseModel" xmlns:mstns="http://www.iho.int/S100BaseModel"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="Orientation">
<xs:restriction base="xs:string">
<xs:enumeration value="Forward"/>
<xs:enumeration value="Reverse"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="RingType">
<xs:restriction base="xs:string">
<xs:enumeration value="Outer"/>
<xs:enumeration value="Inner"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="BoundaryType">
<xs:restriction base="xs:string">
<xs:enumeration value="Begin"/>
<xs:enumeration value="End"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="InterpolationType">
<xs:restriction base="xs:string">
<xs:enumeration value="None"/>
<xs:enumeration value="Linear"/>
<xs:enumeration value="Loxodromic"/>
<xs:enumeration value="CircularArcs3Points"/>
<xs:enumeration value="Geodesic"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="GeometricPrimitive">
<xs:restriction base="xs:string">
<xs:enumeration value="None"/>
<xs:enumeration value="Point"/>
<xs:enumeration value="MultiPoint"/>
<xs:enumeration value="Curve"/>
<xs:enumeration value="Surface"/>
<xs:enumeration value="Coverage"/>
<xs:enumeration value="Complex"/>
</xs:restriction>
```

```
</xs:simpleType>

<xs:complexType name="Coordinate2D">
<xs:sequence>
<xs:element name="x" type="xs:double"/>
<xs:element name="y" type="xs:double"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="Coordinate3D">
<xs:complexContent>
<xs:extension base="Coordinate2D">
<xs:sequence>
<xs:element name="z" type="xs:double"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Coordinates2D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate2D" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="Coordinates3D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate3D" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:group name="Coordinate">
<xs:choice>
<xs:element name="Coordinate2D" type="Coordinate2D"/>
<xs:element name="Coordinate3D" type="Coordinate3D"/>
</xs:choice>
</xs:group>

<xs:group name="Coordinates">
<xs:choice>
<xs:element name="Coordinates2D" type="Coordinates2D"/>
<xs:element name="Coordinates3D" type="Coordinates3D"/>
</xs:choice>
</xs:group>

<xs:complexType name="InformationAssociation">
<xs:attribute name="informationRef" type="xs:positiveInteger" use="required"/>
<xs:attribute name="role" type="xs:string" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="FeatureAssociation">
<xs:attribute name="featureRef" type="xs:positiveInteger" use="required"/>
<xs:attribute name="role" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="SpatialRelation">
<xs:attribute name="ref" type="xs:positiveInteger" use="required"/>
<xs:attribute name="scaleMinimum" type="xs:positiveInteger"></xs:attribute>
<xs:attribute name="scaleMaximum" type="xs:positiveInteger"></xs:attribute>
</xs:complexType>

<xs:complexType name="MaskedRelation">
<xs:complexContent>
<xs:extension base="SpatialRelation">
<xs:attribute name="mask" type="xs:boolean" default="false"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="BoundaryRelation">
<xs:complexContent>
<xs:extension base="SpatialRelation">
<xs:attribute name="boundaryType" type="BoundaryType" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="CurveRelation">
<xs:complexContent>
<xs:extension base="MaskedRelation">
<xs:attribute name="orientation" type="Orientation" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:group name="CurveRelations">
<xs:choice>
<xs:element name="Curve" type="CurveRelation"/>
<xs:element name="CompositeCurve" type="CurveRelation"/>
</xs:choice>
</xs:group>

<xs:group name="SpatialRelations">
<xs:choice>
<xs:element name="Point" type="MaskedRelation"/>
<xs:element name="PointSet" type="MaskedRelation"/>
<xs:element name="Surface" type="MaskedRelation"/>
<xs:group ref="CurveRelations"/>
```

```
</xs:choice>
</xs:group>
<xs:complexType name="Object" abstract="true">
<xs:attribute name="id" type="xs:positiveInteger" use="required"/>
</xs:complexType>

<xs:complexType name="Point" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="Coordinate"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="MultiPoint" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="Coordinates"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="ControlPoints">
<xs:complexContent>
<xs:restriction base="Coordinates2D">
<xs:sequence>
<xs:element name="Coordinate" type="Coordinate2D" minOccurs="2"
                maxOccurs="unbounded"/>
</xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Segment">
<xs:sequence>
<xs:element name="ControlPoints" type="ControlPoints"/>
</xs:sequence>
<xs:attribute name="interpolation" type="InterpolationType" use="required"/>
</xs:complexType>

<xs:complexType name="Curve" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:element name="Boundary" type="BoundaryRelation" minOccurs="0" maxOccurs="2"/>
```

```
<xs:element name="Segment" type="Segment" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="CompositeCurve" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="CurveRelations" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Ring">
<xs:group ref="CurveRelations" minOccurs="1" maxOccurs="unbounded"/>
<xs:attribute name="type" type="RingType" use="required"/>
</xs:complexType>

<xs:complexType name="Surface" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:element name="Ring" type="Ring" minOccurs="1" maxOccurs="unbounded"
></xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Information" abstract="true">
<xs:complexContent>
<xs:extension base="Object"></xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Feature" abstract="true">
<xs:complexContent>
<xs:extension base="Object">
<xs:sequence>
<xs:group ref="SpatialRelations" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="primitive" type="GeometricPrimitive" use="required"
></xs:attribute>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

```
</xs:schema>
```

A.2 Symbol Definition Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.iho.int/S100SymbolDefinition"
   xmlns="http://www.iho.int/S100SymbolDefinition"
   xmlns:mstns="http://www.iho.int/S100SymbolDefinition"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- THE GRAPHICS BASE PACKAGE -->
<!-- Enumeration CRSType  -->
<xs:simpleType name="Interval01">
<xs:restriction base="xs:double">
<xs:minInclusive value="0.0"></xs:minInclusive>
<xs:maxInclusive value="1.0"></xs:maxInclusive>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="CRSType">
<xs:restriction base="xs:string">
<xs:enumeration value="GeographicCRS"/>
<xs:enumeration value="PortrayalCRS"/>
<xs:enumeration value="LocalCRS"/>
</xs:restriction>
</xs:simpleType>
<!--  Class Point -->
<xs:complexType name="Point">
<xs:sequence>
<xs:element name="x" type="xs:double"/>
<xs:element name="y" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!--  Class Vector -->
<xs:complexType name="Vector">
<xs:sequence>
<xs:element name="x" type="xs:double"/>
<xs:element name="y" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!--  Class Sector -->
<xs:complexType name="Sector">
<xs:sequence>
<xs:element name="startAngle" type="xs:double"/>
<xs:element name="endAngle" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!-- Class Color -->
<xs:complexType name="Color">
<xs:sequence>
<xs:element name="token" type="xs:string"/>
<xs:element name="transparency" type="Interval01" default="0.0"/>
</xs:sequence>
```

```
</xs:complexType>
<!-- Class OverrideColor -->
<xs:complexType name="OverrideColor">
<xs:sequence>
<xs:element name="override" type="Color"/>
<xs:element name="color" type="Color"/>
</xs:sequence>
</xs:complexType>
<!-- Class Pen -->
<xs:complexType name="Pen">
<xs:sequence>
<xs:element name="color" type="Color"/>
</xs:sequence>
<xs:attribute name="width" type="xs:double" use="required"/>
</xs:complexType>
<!-- Class Pixmap -->
<xs:complexType name="Pixmap">
<xs:sequence>
<xs:element name="symbolReference" type="xs:string"/>
<xs:element name="overrideAll" type="Color" maxOccurs="1" minOccurs="0"/>
<xs:element name="override" type="OverrideColor" maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<!-- Class Polyline -->
<xs:complexType name="Polyline">
<xs:sequence>
<xs:element name="point" type="Point" minOccurs="2" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!-- Class Arc3Points -->
<xs:complexType name="Arc3Points">
<xs:sequence>
<xs:element name="startPoint" type="Point"/>
<xs:element name="medianPoint" type="Point"/>
<xs:element name="endPoint" type="Point"/>
</xs:sequence>
</xs:complexType>
<!-- Class ArcByRadius -->
<xs:complexType name="ArcByRadius">
<xs:sequence>
<xs:element name="center" type="Point"/>
<xs:element name="sector" type="Sector" minOccurs="0" maxOccurs="1"/>
<xs:element name="radius" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!-- Class Annulus -->
<xs:complexType name="Annulus">
<xs:sequence>
<xs:element name="center" type="Point"/>
```

```
<xs:element name="innerRadius" type="xs:double" minOccurs="0" maxOccurs="1"/>
<xs:element name="outerRadius" type="xs:double"/>
<xs:element name="sector" type="Sector" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
<!-- group for segments -->
<xs:group name="Segment">
<xs:choice>
<xs:element name="polyline" type="Polyline"/>
<xs:element name="arc3Points" type="Arc3Points"/>
<xs:element name="arcByRadius" type="ArcByRadius"/>
<xs:element name="annulus" type="Annulus"/>
</xs:choice>
</xs:group>
<!-- Class Path -->
<xs:complexType name="Path">
<xs:sequence>
<xs:group ref="Segment" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!-- THE SYMBOL PACKAGE -->
<!-- Enumeration LinePlacementMode -->
<xs:simpleType name="LinePlacementMode">
<xs:restriction base="xs:string">
<xs:enumeration value="Relative"/>
<xs:enumeration value="Absolute"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration AreaPlacementMode -->
<xs:simpleType name="AreaPlacementMode">
<xs:restriction base="xs:string">
<xs:enumeration value="VisibleParts"/>
<xs:enumeration value="Geographic"/>
</xs:restriction>
</xs:simpleType>
<!-- Class LineSymbolPlacement -->
<xs:complexType name="LineSymbolPlacement">
<xs:sequence>
<xs:element name="offset" type="xs:double"/>
<xs:element name="placementMode" type="LinePlacementMode"/>
</xs:sequence>
</xs:complexType>
<!-- Class AreaSymbolPlacement -->
<xs:complexType name="AreaSymbolPlacement">
<xs:sequence>
<xs:element name="placementMode" type="AreaPlacementMode" default="VisibleParts"/>
</xs:sequence>
</xs:complexType>
```

```
<!-- Class Symbol -->
<xs:complexType name="Symbol">
<xs:sequence>
<xs:element name="symbolReference" type="xs:string"/>
<xs:element name="rotation" type="xs:double" default="0.0"/>
<xs:element name="rotationCRS" type="CRSType" default="PortrayalCRS"/>
<xs:element name="scaleFactor" type="xs:double" default="1.0"/>
<xs:element name="offset" type="Vector" minOccurs="0" maxOccurs="1"/>
<xs:element name="overrideAll" type="Color" minOccurs="0" maxOccurs="1"/>
<xs:element name="override" type="OverrideColor" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="linePlacement" type="LineSymbolPlacement" minOccurs="0" maxOccurs="1"/>
<xs:element name="areaPlacement" type="AreaSymbolPlacement" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>

<!-- THE LINE STYLES PACKAGE -->
<!-- Enumeration JoinStyle -->
<xs:simpleType name="JoinStyle">
<xs:restriction base="xs:string">
<xs:enumeration value="Bevel"/>
<xs:enumeration value="Miter"/>
<xs:enumeration value="Round"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration JoinStyle -->
<xs:simpleType name="CapStyle">
<xs:restriction base="xs:string">
<xs:enumeration value="Butt"/>
<xs:enumeration value="Square"/>
<xs:enumeration value="Round"/>
</xs:restriction>
</xs:simpleType>
<!-- Class Dash -->
<xs:complexType name="Dash">
<xs:sequence>
<xs:element name="start" type="xs:double"/>
<xs:element name="length" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!-- Class SimpleLineStyle -->
<xs:complexType name="SimpleLineStyle">
<xs:sequence>
<xs:element name="capStyle" type="CapStyle" default="Butt"/>
<xs:element name="joinStyle" type="JoinStyle" default="Miter"/>
<xs:element name="intervalLength" type="xs:double" minOccurs="0" maxOccurs="1"/>
<xs:element name="offset" type="xs:double" default="0.0"/>
<xs:element name="pen" type="Pen"/>
<xs:element name="dash" type="Dash" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
```

```
</xs:complexType>
<!-- Class ComplexLineStyle -->
<xs:complexType name="ComplexLineStyle">
<xs:sequence>
<xs:element name="symbolReference" type="xs:string"/>
<xs:element name="scaleFactor" type="xs:double" default="1.0"/>
<xs:element name="overrideAll" type="Color" minOccurs="0" maxOccurs="1"/>
<xs:element name="override" type="OverrideColor" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!-- Group LineStyle -->
<xs:group name="LineStyle">
<xs:choice>
<xs:element name="simpleLineStyle" type="SimpleLineStyle"/>
<xs:element name="complexLineStyle" type="ComplexLineStyle"/>
</xs:choice>
</xs:group>

<!-- THE AREA FILLS PACKAGE -->
<!-- Enumeration AreaCRSType -->
<xs:simpleType name="AreaCRSType">
<xs:restriction base="xs:string">
<xs:enumeration value="Global"/>
<xs:enumeration value="LocalGeometry"/>
</xs:restriction>
</xs:simpleType>
<!-- Class ColorFill -->
<xs:complexType name="ColorFill">
<xs:sequence>
<xs:element name="color" type="Color"/>
</xs:sequence>
</xs:complexType>
<!-- Class PatternFill -->
<xs:complexType name="PatternFill" abstract="true">
<xs:sequence>
<xs:element name="areaCRS" type="AreaCRSType"/>
</xs:sequence>
</xs:complexType>
<!-- Class PixmapFill -->
<xs:complexType name="PixmapFill">
<xs:complexContent>
<xs:extension base="PatternFill">
<xs:sequence>
<xs:element name="pixmap" type="Pixmap"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<!-- Class SymbolFill -->
```

```
<xs:complexType name="SymbolFill">
<xs:complexContent>
<xs:extension base="PatternFill">
<xs:sequence>
<xs:element name="symbol" type="Symbol"/>
<xs:element name="v1" type="Vector"/>
<xs:element name="v2" type="Vector"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<!-- Class Hatch -->
<xs:complexType name="Hatch">
<xs:sequence>
<xs:element name="line" type="SimpleLineStyle"/>
<xs:element name="direction" type="Vector"/>
<xs:element name="distance" type="xs:double"/>
</xs:sequence>
</xs:complexType>
<!-- Class HatchFill -->
<xs:complexType name="HatchFill">
<xs:complexContent>
<xs:extension base="PatternFill">
<xs:sequence>
<xs:element name="hatch" type="Hatch" minOccurs="1" maxOccurs="2"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<!-- Group AreaFill -->
<xs:group name="AreaFill">
<xs:choice>
<xs:element name="colorFill" type="ColorFill"/>
<xs:element name="pixmapFill" type="PixmapFill"/>
<xs:element name="symbolFill" type="SymbolFill"/>
<xs:element name="hatchFill" type="HatchFill"/>
</xs:choice>
</xs:group>

<!-- THE TEXT PACKAGE -->
<!-- Enumeration FontProportion -->
<xs:simpleType name="FontProportion">
<xs:restriction base="xs:string">
<xs:enumeration value="MonoSpaced"/>
<xs:enumeration value="Proportional"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration FontSlant -->
<xs:simpleType name="FontSlant">
```

```
<xs:restriction base="xs:string">
<xs:enumeration value="Upright"/>
<xs:enumeration value="Italics"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration FontWeight -->
<xs:simpleType name="FontWeight">
<xs:restriction base="xs:string">
<xs:enumeration value="Light"/>
<xs:enumeration value="Medium"/>
<xs:enumeration value="Bold"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration TextFlag -->
<xs:simpleType name="TextFlag">
<xs:restriction base="xs:string">
<xs:enumeration value="UnderLine"/>
<xs:enumeration value="StrikeThrough"/>
<xs:enumeration value="UpperLine"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration HorizontalAlignment -->
<xs:simpleType name="HorizontalAlignment">
<xs:restriction base="xs:string">
<xs:enumeration value="Start"/>
<xs:enumeration value="End"/>
<xs:enumeration value="Center"/>
</xs:restriction>
</xs:simpleType>
<!-- Enumeration VerticalAlignment -->
<xs:simpleType name="VerticalAlignment">
<xs:restriction base="xs:string">
<xs:enumeration value="Top"/>
<xs:enumeration value="Bottom"/>
<xs:enumeration value="Center"/>
</xs:restriction>
</xs:simpleType>
<!-- Class Font -->
<xs:complexType name="Font">
<xs:sequence>
<xs:element name="serifs" type="xs:boolean"/>
<xs:element name="weight" type="FontWeight"/>
<xs:element name="slant" type="FontSlant"/>
<xs:element name="proportion" type="FontProportion"/>
</xs:sequence>
</xs:complexType>
<!-- Class TextFlags -->
<xs:complexType name="TextFlags">
<xs:sequence>
```

```
<xs:element name="flag" type="TextFlag" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!-- Class TextElement -->
<xs:complexType name="TextElement">
<xs:sequence>
<xs:element name="text" type="xs:string"/>
<xs:element name="bodySize" type="xs:double"/>
<xs:element name="verticalOffset" type="xs:double" default="0.0"/>
<xs:element name="flags" type="TextFlags" minOccurs="0" maxOccurs="1"/>
<xs:element name="foreground" type="Color"/>
<xs:element name="background" type="Color" minOccurs="0" maxOccurs="1"/>
<xs:element name="font" type="Font"/>
</xs:sequence>
</xs:complexType>
<!-- Class Text -->
<xs:complexType name="Text" abstract="true">
<xs:sequence>
<xs:element name="element" type="TextElement" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="horizontalAlignment" type="HorizontalAlignment" default="Start"/>
<xs:attribute name="verticalAlignment" type="VerticalAlignment" default="Bottom"/>
</xs:complexType>
<xs:complexType name="TextPoint">
<xs:complexContent>
<xs:extension base="Text">
<xs:sequence>
<xs:element name="offset" type="Vector" minOccurs="0" maxOccurs="1"/>
<xs:element name="rotation" type="xs:double" default="0"/>
<xs:element name="areaPlacement" type="AreaSymbolPlacement" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="TextLine">
<xs:complexContent>
<xs:extension base="Text">
<xs:sequence>
<xs:element name="startOffset" type="xs:double"/>
<xs:element name="endOffset" type="xs:double" minOccurs="0" maxOccurs="1"/>
<xs:element name="placementMode" type="LinePlacementMode"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:group name="Text">
<xs:choice>
<xs:element name="textPoint" type="TextPoint"/>
<xs:element name="textLine" type="TextLine"/>
```

```
</xs:choice>
</xs:group>
</xs:schema>
```

A.3 Presentation Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:s100Symbol="http://www.iho.int/S100SymbolDefinition" attributeFormDefault="qualified">
<xs:import namespace="http://www.iho.int/S100SymbolDefinition"
schemaLocation="S100SymbolDefinition.xsd"/>

<xs:complexType name="FeatureReference">
<xs:attribute name="reference" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="SpatialReference">
<xs:attribute name="reference" type="xs:string" use="required"/>
<xs:attribute name="forward" type="xs:boolean" default="true"/>
</xs:complexType>

<xs:complexType name="DrawingInstruction" abstract="true">
<xs:sequence>
<xs:element name="featureReference" type="FeatureReference">

</xs:element>
<xs:element name="spatialReference" type="SpatialReference" minOccurs="0" maxOccurs="1">

</xs:element>
<xs:element name="viewingGroup" type="xs:string"/>
<xs:element name="displayPlane" type="xs:string"/>
<xs:element name="drawingPriority" type="xs:int"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="NullInstruction">
<xs:complexContent>
<xs:extension base="DrawingInstruction"/>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="PointInstruction">
<xs:complexContent>
<xs:extension base="DrawingInstruction">
<xs:sequence>
<xs:element name="symbol" type="s100Symbol:Symbol"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AreaInstruction">
<xs:complexContent>
```

```
<xs:extension base="DrawingInstruction">
<xs:sequence>
<xs:group ref="s100Symbol:AreaFill"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="LineInstruction">
<xs:complexContent>
<xs:extension base="DrawingInstruction">
<xs:sequence>
<xs:group ref="s100Symbol:LineStyle"/>
</xs:sequence>
<xs:attribute name="suppression" type="xs:boolean" default="true"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="TextInstruction">
<xs:complexContent>
<xs:extension base="DrawingInstruction">
<xs:sequence>
<xs:group ref="s100Symbol:Text"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="CoverageInstruction">
<xs:complexContent>
<xs:extension base="DrawingInstruction">
<!-- NEEDS TO BE DEFINED -->
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedGeometry" abstract="true">
<xs:complexContent>
<xs:extension base="DrawingInstruction">
<xs:sequence>
<xs:element name="crs" type="s100Symbol:CRSType"/>
<xs:group ref="s100Symbol:Text" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedPoint">
```

```
<xs:complexContent>
<xs:extension base="AugmentedGeometry">
<xs:sequence>
<xs:element name="position" type="s100Symbol:Point"/>
<xs:element name="symbol" type="s100Symbol:Symbol"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedLineOrArea" abstract="true">
<xs:complexContent>
<xs:extension base="AugmentedGeometry">
<xs:sequence>
<xs:group ref="s100Symbol:LineStyle" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedRay">
<xs:complexContent>
<xs:extension base="AugmentedLineOrArea">
<xs:sequence>
<xs:element name="direction" type="xs:double"/>
<xs:element name="length" type="xs:double"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedPath">
<xs:complexContent>
<xs:extension base="AugmentedLineOrArea">
<xs:sequence>
<xs:element name="path" type="s100Symbol:Path"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="AugmentedArea">
<xs:complexContent>
<xs:extension base="AugmentedPath">
<xs:sequence>
<xs:group ref="s100Symbol:AreaFill" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
```

```
</xs:complexType>

<xs:group name="DisplayInstruction">
<xs:choice>
<xs:element name="nullInstruction" type="NullInstruction"/>
<xs:element name="pointInstruction" type="PointInstruction"/>
<xs:element name="lineInstruction" type="LineInstruction"/>
<xs:element name="areaInstruction" type="AreaInstruction"/>
<xs:element name="textInstruction" type="TextInstruction"/>
<xs:element name="augmentedPoint" type="AugmentedPoint"/>
<xs:element name="augmentedRay" type="AugmentedRay"/>
<xs:element name="augmentedPath" type="AugmentedPath"/>
<xs:element name="augmentedArea" type="AugmentedArea"/>
</xs:choice>
</xs:group>

<xs:complexType name="DisplayList">
<xs:sequence>
<xs:group ref="DisplayInstruction" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="displayList" type="DisplayList"/>

</xs:schema>
```

A.4 Portrayal Catalogue Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="IdString">
<xs:restriction base="xs:string">
<xs:minLength value="1"></xs:minLength>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="DisplayModeCategory">
<xs:restriction base="xs:string">
<xs:enumeration value="BaseMode"/>
<xs:enumeration value="SupplementMode"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="ParameterType">
<xs:restriction base="xs:string">
<xs:enumeration value="Boolean"/>
<xs:enumeration value="Integer"/>
```

```
<xs:enumeration value="Double"/>
<xs:enumeration value="String"/>
<xs:enumeration value="Date"/>
</xs:restriction>
</xs:simpleType>

<xs:complexType name="Description">
<xs:sequence>
<xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="description" type="xs:string"/>
<xs:element name="language" type="xs:language"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="CatalogItem" abstract="true">
<xs:sequence>
<xs:element name="description" type="Description" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="id" type="IdString" use="required"/>
</xs:complexType>

<xs:complexType name="SVGItem">
<xs:complexContent>
<xs:extension base="CatalogItem">
<xs:sequence>
<xs:element name="svgFileName" type="xs:anyURI"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="ViewingGroup">
<xs:complexContent>
<xs:extension base="CatalogItem"/>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="DisplayMode">
<xs:complexContent>
<xs:extension base="CatalogItem">
<xs:sequence>
<xs:element name="category" type="DisplayModeCategory"/>
<xs:element name="viewingGroup" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="DisplayPlane">
```

```
<xs:complexContent>
<xs:extension base="CatalogItem"/>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="ContextParameter">
<xs:complexContent>
<xs:extension base="CatalogItem">
<xs:sequence>
<xs:element name="type" type="ParameterType"/>
<xs:element name="default" type="xs:anyURI"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="XSLTItem">
<xs:complexContent>
<xs:extension base="CatalogItem">
<xs:sequence>
<xs:element name="xsltFileName" type="xs:string"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="Pixmaps">
<xs:sequence>
<xs:element name="pixmap" type="SVGItem" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ColorProfiles">
<xs:sequence>
<xs:element name="colorProfile" type="SVGItem" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="Symbols">
<xs:sequence>
<xs:element name="symbol" type="SVGItem" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="LineStyles">
<xs:sequence>
<xs:element name="lineStyle" type="SVGItem" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="ViewingGroups">
<xs:sequence>
<xs:element name="viewingGroup" type="ViewingGroup" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="DisplayModes">
<xs:sequence>
<xs:element name="displayMode" type="DisplayMode" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="DisplayPlanes">
<xs:sequence>
<xs:element name="displayPlane" type="DisplayPlane" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="Context">
<xs:sequence>
<xs:element name="parameter" type="ContextParameter" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="Rules">
<xs:sequence>
<xs:element name="rule" type="XSLTItem" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="PortrayalCatalog">
<xs:sequence>
<xs:element name="pixmaps" type="Pixmaps">
<xs:key name="pixmapKey">
<xs:selector xpath="pixmap"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="colorProfiles" type="ColorProfiles">
<xs:key name="colorProfileKey">
<xs:selector xpath="colorProfile"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="symbols" type="Symbols">
<xs:key name="symbolKey">
<xs:selector xpath="symbol"/>
<xs:field xpath="@id"/>
```

```
</xs:key>
</xs:element>
<xs:element name="lineStyles" type="LineStyles">
<xs:key name="lineStyleKey">
<xs:selector xpath="lineStyle"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="viewingGroups" type="ViewingGroups">
<xs:key name="viewingGroupKey">
<xs:selector xpath="viewingGroup"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="displayModes" type="DisplayModes">
<xs:key name="displayModeKey">
<xs:selector xpath="displayMode"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="displayPlane" type="DisplayPlanes">
<xs:key name="displayPlaneKey">
<xs:selector xpath="displayPlane"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="context" type="Context">
<xs:key name="contextKey">
<xs:selector xpath="parameter"/>
<xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="rules" type="Rules">
<xs:key name="ruleKey">
<xs:selector xpath="rule"/>
<xs:field xpath="@id"/>
</xs:key>

</xs:element>
</xs:sequence>
<xs:attribute name="productId" type="xs:string" use="required"/>
<xs:attribute name="version" type="xs:string" use="required"/>
</xs:complexType>

<!-- THE ROOT ELEMENT -->
<xs:element name="portrayalCatalog" type="PortrayalCatalog">
<!-- KEYS AND KEYREFS TO BE DEFINED -->
</xs:element>
```

```
</xs:schema>
```

Annex B – How to write a schema for a specific data product

B.1 Preface

This standard describes a base schema that contains base types to be used within a schema for a data product. In such a schema the real feature types and information types will be defined with their properties. Such properties are attributes or associations. Spatial types have to be derived from the appropriate base types as well.

This section will describe how such a schema can be created. Techniques will be introduced to map the data model from a feature catalogue to an input schema for portrayal. Although the schema can cover the entire data model it is sufficient to model only the part that is relevant for portrayal.

B.2 Importing the base schema

The product schema will be based on the S100 base schema. Therefore the base schema must be made available within the product schema. This can be achieved by the xs:import instruction.

```
<xs:import
  namespace="http://www.iho.int/S100BaseModel"
  schemaLocation="S100BaseModel.xsd"/>
```

B.3 Spatial Objects

Since all spatial types in the base schema are abstract there must be derived types in the product schema. Even if no additional properties are added to the base type the advantage is that all spatial types belongs to the same namespace the rest of the types defined in the product schema. In the following example a spatial type `Point` is derived from `s100:Point` and an association is added to an information type defining the spatial quality.

```
<xs:complexType name="Point">
<xs:complexContent>
<xs:extension base="s100:Point">
<xs:sequence>
<xs:element name="spatialQuality" type="s100:InformationAssociation" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

B.4 Information types and feature types

The portrayal of many feature types depends on some attributes. As a simple example we assume a feature type ‘Beacon, cardinal’. The portrayal should only depend on the attribute ‘Category of cardinal mark’. The definition of that feature type would look like.

```
<xs:complexType name="BeaconCardinal">
<xs:complexContent>
<xs:extension base="s100:Feature">
```

```
<xs:sequence>
<xs:element name="categoryOfCardinalMark" nillable="true" type="xs:int"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

An instance looks like:

```
<BeaconCardinal id="2">
<s100:Point ref="3"/>
<categoryOfCardinalMark>3</categoryOfCardinalMark>
</BeaconCardinal>
```

A feature hierarchy can be introduced to avoid redundant definitions. Assume that all feature types of a data product can have an association to a note (an information type). We can introduce an abstract type with this property and derive other feature types from this type instead from `s100:Feature`.

```
<xs:complexType name="Feature" abstract="true">
<xs:complexContent>
<xs:extension base="s100:Feature">
<xs:sequence>
<xs:element name="noteAssociation" type="s100:InformationAssociation"
 minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The beacon, cardinal type is defined now:

```
<xs:complexType name="BeaconCardinal">
<xs:complexContent>
<xs:extension base="Feature">
<xs:sequence>
<xs:element name="categoryOfCardinalMark" nillable="true" type="xs:int"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The instance may looks like:

```
<BeaconCardinal id="2">
<s100:Point ref="3"/>
<noteAssociation role="aNote" informationRef="1"/>
<categoryOfCardinalMark>3</categoryOfCardinalMark>
</BeaconCardinal>
```

Information types are very similar to feature types. Here are two examples.

The first defines an information type for carrying quality information for spatial types.

```
<xs:complexType name="SpatialQuality">
<xs:complexContent>
<xs:extension base="s100:Information">
<xs:sequence>
<xs:element name="qualityOfPosition" type="xs:int"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The second example shows a note type for general notes that can be associated to any feature type.

```
<xs:complexType name="ChartNote">
<xs:complexContent>
<xs:extension base="s100:Information">
<xs:sequence>
<xs:element name="note" type="Note" maxOccurs="unbounded"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The element `<note>` in the example corresponds here to a complex attribute. See the section on complex attributes for more details.

B.5 Associations

Associations are named relationships between objects. We have two types of associations: information associations for relationships between any object and an information type and feature associations for relationships between two feature types. The associations will be realized in the schema by elements that have the name from the camelCase code of the association. They are of type `s100:InformationAssociation` or `s100:FeatureAssociation`. As an example we extend the beacon cardinal type with a feature association to the underlying area.

```
<xs:complexType name="BeaconCardinal">
<xs:complexContent>
<xs:extension base="Feature">
<xs:sequence>
<xs:element name="underlyingArea" type="s100:FeatureAssociation" minOccurs="0"/>
<xs:element name="categoryOfCardinalMark" nillable="true" type="xs:int/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

The instance now is:

```
<BeaconCardinal id="2">
<s100:Point ref="3"/>
<noteAssociation role="aNote" informationRef="1"/>
<underlyingArea role="area" featureRef="3"/>
<categoryOfCardinalMark>3</categoryOfCardinalMark>
</BeaconCardinal>
```

B.6 Complex attributes

Complex attributes can be easily defined as complex types in XML. As an example we have a look at the complex attribute `note` of our `ChartNote` information type. This attribute comprises two simple data types the text and the language of the text. The definition is:

```
<xs:complexType name="Note">
<xs:sequence>
<xs:element name="noteText" type="xs:string"/>
<xs:element name="language" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

An instance of the information type ChartNote could be:

```
<ChartNote id="1">
<note>
<noteText>Hello world!</noteText>
<language>en</language>
</note>
<note>
<noteText>Hallo Welt!</noteText>
<language>de</language>
</note>
</ChartNote>
```

B.7 Data set definition

Tbd.

B.8 <key> and <keyref>

Tbd.