S-100 – Part 9

Portrayal

Page intentionally left blank

Contents

Page intentionally left blank

9-1 Scope

This part of the standard defines the models, structures and formats for a machine readable Portrayal Catalogue. The intent is for a Portrayal Catalogue to be delivered separately from product datasets such that it can be imported and interpreted to map Feature objects defined according to the Part 3 General Feature Model (GFM) into Drawing Instructions and symbolization.

The actual contents of a Portrayal Catalogue need to be defined as part of a Product Specification using the mechanism and structures defined in this part. For example a product specification would include ~~an input Schema derived from the abstract schema provided herein,~~ a set of mapping rules, a set of symbols, linestyles, colors etc and make it available for use with product datasets.

This part includes mechanisms for portrayal of 2D vector data according to the GFM as well as Coverage data. It does not include drawing instructions and symbol structures intended for 3D portrayal. ~~It does not include the generation of alarms and indications however this might be implemented using a very similar mechanism.~~ It does not include the generation of pick reports or textual reports however the approach of exposing the content to mapping rules could be implemented to generate textual or html formatted output.

9-2 Conformance

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

9-3 Normative references

The following referenced documents are required for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including amendments) applies.

ICC Specification Version 4 – International Color Consortium

ISO 19117: 2012 (E), *Geographic Information – Portrayal*

W3C.REC-XSLT-1.0-19991116, *XSL Transformations (XSLT) Version 1.0*, *W3C Recommendation* 16 November 1999, <http://www.w3.org/TR/xslt>

W3C.REC-SVGTiny12-20081222, *Scalable Vector Graphics (SVG) Tiny 1.2 Specification*, *W3C Recommendation* 22 December 2008***,*** <http://www.w3.org/TR/2008/REC-SVGTiny12-20081222>

W3C.REC-CSS2-20110607, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, W3C Recommendation* 07 June 2011, <http://www.w3.org/TR/2011/REC-CSS2-20110607>

TrueType-1.66-1995, *True Type Font Revision 1.66* 1995, <http://www.microsoft.com/typography/SpecificationsOverview.mspx>

9-4 Portrayal Catalogue

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 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.

9-5 General portrayal model

The general portrayal model is illustrated in figure 9.1.

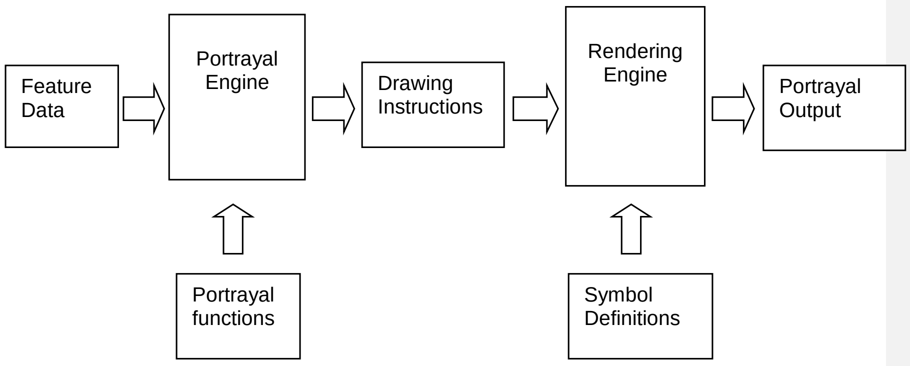


Figure 9-1 — General portrayal model

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.

9-5.1 The portrayal process

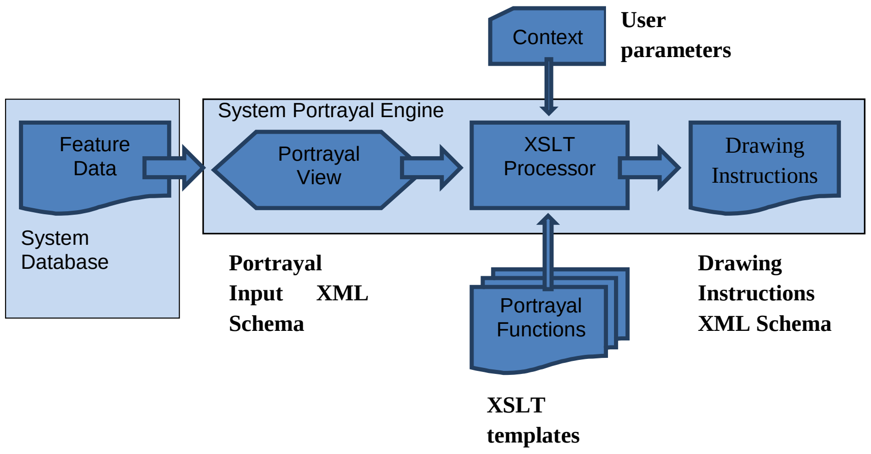


Figure 9-2 — Portrayal process

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 XML output. 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 tree however 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 described~~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.

9-6 Package overview

The following diagram shows the packages for implementing this standard.

The InputSchema describes how ~~the~~ data is presented to the portrayal engine (XSLT processor). The Presentation package includes two subpackages one describing the portrayal catalogue structure the other describing the drawing instructions. Drawing instructions are the output of the portrayal engine (XSLT processor)

The SymbolDefinitions package describes the graphic primitives used for portrayal.

The portrayal engine is using standard XSLT. There is no package describing this part of the portrayal.

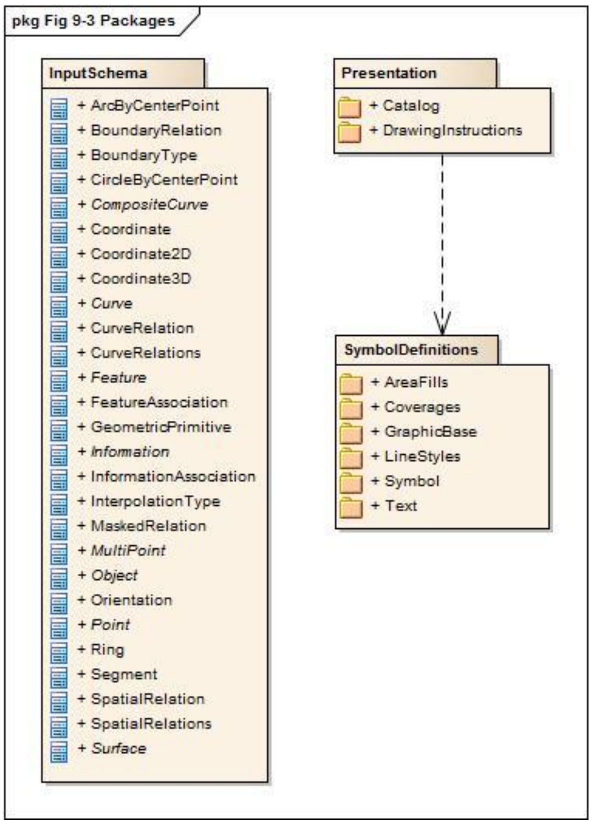


Figure 9-3 — Packages

9-7 Data input schema

9-7.1 Introduction

The data input schema describes how ~~the~~ data ~~is~~should be presented to the XSLT processor. ~~The~~ Encoded data can be transformed to an XML document or a presentation of such a document, for example 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~~ Whichever method is used, this schema describes how the data must be organized. In this standard only the base types are described. These types should be used along with a products feature catalogue to generate the input XML for portrayal processing, as described in Annex A.~~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.~~

Formatted: Not Highlight

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

9-7.2 Enumerations

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

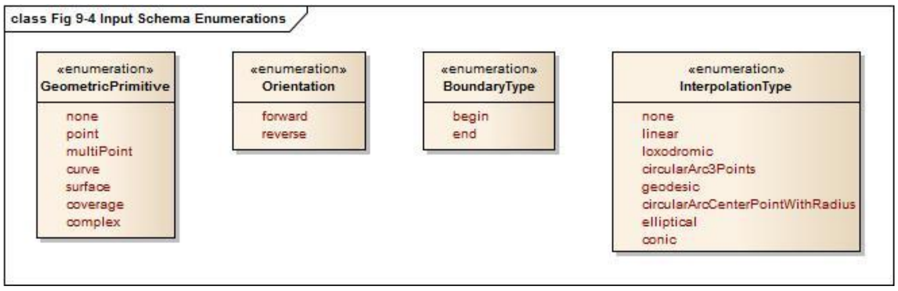

class Fig 9-4 Input Schema Enumerations
«enumeration» GeometricPrimitive
none
point
multiPoint
curve
surface
coverage
complex
«enumeration» Orientation
forward
reverse
«enumeration» BoundaryType
begin
end
«enumeration» InterpolationType
none
linear
loxodromic
circularArc3Points
geodesic
circularArcCenterPointWithRadius
elliptical
conic

Figure 9-4 — Input Schema Enumerations

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="None"/>
      <xs:enumeration value="Linear"/>
      <xs:enumeration value="Loxodromic"/>
      <xs:enumeration value="CircularArc3Points"/>
      <xs:enumeration value="Geodesic"/>
      <xs:enumeration value="CircularArcCenterPointWithRadius"/>
      <xs:enumeration value="Elliptical"/>
      <xs:enumeration value="Conic"/>
      <xs:enumeration value="PolynomialSpline"/>
      <xs:enumeration value="BezierSpline"/>
      <xs:enumeration value="BSpline"/>
      <xs:enumeration value="BlendedParabolic"/>
   </xs:restriction>
</xs:simpleType>
```

Commented [DMG1]: Added by S100WG5-4.10 / TSM7-5.3f

9-7.3 Coordinates

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

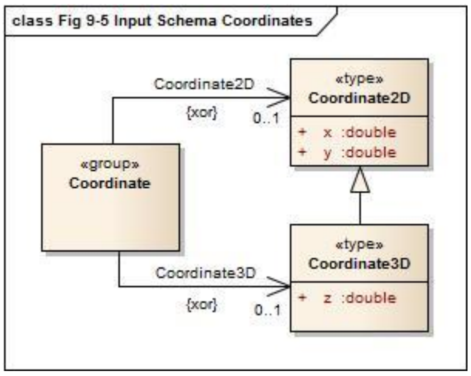


Figure 9-5 — Input Schema Coordinates

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 tuples can be used. The use of 2D or 3D tuples is mutually exclusive.

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

9-7.4 Associations

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

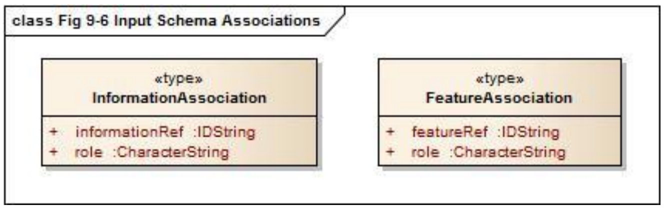


Figure 9-6 — Input Schema Associations

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

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

<xs:complexType name="FeatureAssociation">
   <xs:attribute name="featureRef" type=" IDString " 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.

Each S100_FC_InformationAssociation and S100_FC_FeatureAssociation defined in a feature catalogue describes a subtype of InformationAssociation and FeatureAssociation respectively. These subtypes may define additional attributes of the association, these attributes should be included in the input XML as described in Annex A.~~If a Product Specification requires attributes for an association a specific type must be sub-classed in the schema of the Product Specification.~~

Formatted: Not Highlight

9-7.5 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.

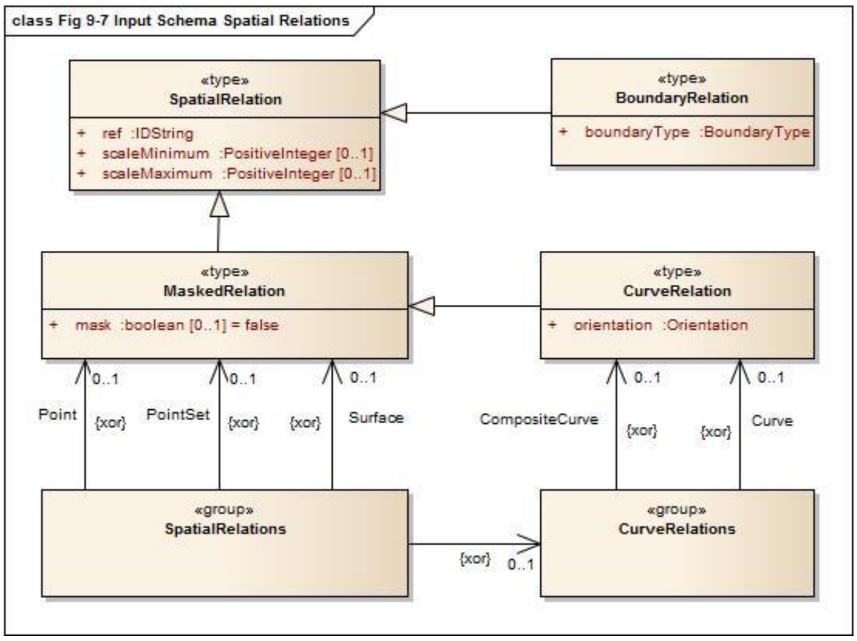

class Fig 9-7 Input Schema Spatial Relations
«type» SpatialRelation
+ ref :IDString
+ scaleMinimum :PositiveInteger [0..1]
+ scaleMaximum :PositiveInteger [0..1]
«type» BoundaryRelation
+ boundaryType :BoundaryType
«type» MaskedRelation
+ mask :boolean [0..1] = false
«type» CurveRelation
+ orientation :Orientation
0..1
Point
{xor}
PointSet
{xor}
Surface
CompositeCurve
Curve
«group» SpatialRelations
«group» CurveRelations

Figure 9-7 — Input Schema Spatial Relations

The type SpatialRelation is the base type for all relations to spatial objects. The ref attribute corresponds to the attribute id of the spatial object.

```
<xs:complexType name="SpatialRelation">
   <xs:attribute name="ref" type=" IDString " use="required"/>
   <xs:attribute name="scaleMinimum" type="xs:positiveInteger" use="required"/>
   <xs:attribute name=" scaleMaximum " type="xs:positiveInteger" 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, for example 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 to 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 Annex A~~B – How to write a schema for a specific data product~~.

9-7.6 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 object needs to be uniquely 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="IDString" use="required"/>
</xs:complexType>
```

Note that the type of the identifier is IDString to be as general as possible with respect to different methods used for identification. The characters allowed in this string are 0-9a-zA-Z.

```
<xs:simpleType name="IDString">
<xs:restriction base="xs:string">
                    <xs:minLength value="1"/>
                    <xs:pattern value="[0-9a-zA-Z_]*"/>
          </xs:restriction>
</xs:simpleType>
```

The model of all objects is given in following diagram.

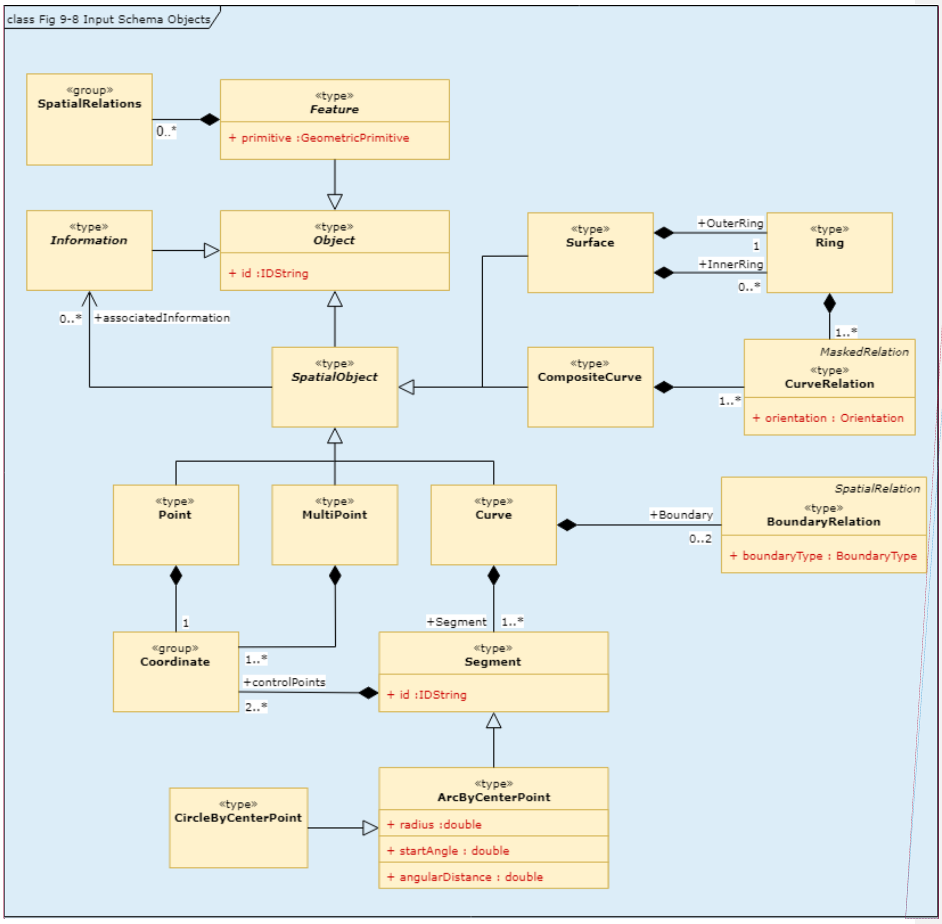


Figure 9-8 — Input Schema Objects

Commented [DMG2]: * Add association from *Feature* to *Information* (*InformationAssociation*)
* Add association from *Feature* to itself (*FeatureAssociation*)
* Add notes for substitution of associationInformation / associatedFeature using a code from the feature catalogue, see 9-7.79-7.7.2 and 9-9.

Commented [JW3]: Refer to paper S-100WG5-04.12. May need to get a better version of this diagram from Raphael.

9-7.7 Spatial objects

9-7.7.1 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

Attributes are not permitted for spatial objects by the GFM. All types are derived from the type Object, meaning they have an identifier.

9-7.7.2 SpatialObject

SpatialObject is an abstract class which serves as the base class for all spatial objects. It provides a one-way association to `Information`, which can be used to provide spatial quality or other information for any spatial object.

When an association is present, ~~T~~the code of an ~~InformationAssociation~~ information association defined ~~within~~ in the ~~products~~ feature catalogue should be used as the element name in place of *associatedInformation~~for the associated information~~* - ~~.~~ ~~S~~see Annex A for more information.

```
<xs:complexType name="SpatialObject" abstract="true">
   <xs:complexContent>
      <xs:extension base="Object">
         <xs:sequence>
            <xs:element name="associatedInformation" type="InformationAssociation" minOccurs="0"
                                          maxOccurs="unbounded"/>
          <!—When generating input XML the base schema should be extended to provide a substitution group fo
         information association defined in the feature catalogue. For example,  if the feature catalogue defin
         information association having a code of SpatialAssociation, the base schema should be extended with:
             <xs:element name="SpatialAssociation" substitutionGroup="associatedInformation"/>
          -->
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Formatted: Font: Italic

Commented [DMG4]: This should be InformationAssociation, not Information so that we reference the information type rather than duplicating it in each spatial object to which it applies.

The code of an InformationAssociation defined within the feature catalogue should be used as the element name rather than „associatedInformation“.

Formatted: Justified, Indent: Left: 0.5", Right: -0.76", Line spacing: Multiple 1.15 li

Formatted: Font color: Green

Formatted: Font color: Green

Commented [JW5]: Refer to paper S-100WG5-04.12.

9-7.7.3 Point

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

```
<xs:complexType name="Point">
   <xs:complexContent>
      <xs:extension base="SpatialObject">
         <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

9-7.7.4 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 maxOccurs is set to unbounded.

```
<xs:complexType name="MultiPoint">
   <xs:complexContent>
      <xs:extension base="SpatialObject">
         <xs:sequence>
            <xs:group ref="Coordinate" minOccurs="1" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

9-7.7.5 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. There are two special types of segments:

1. ArcByCenterPoint
   A circular arc defined by a center point and a radius. The beginning of the arc is defined by the start angle and the length of the arc is defined by the angular length. This length is a signed quantity defining the direction of the arc: positive means clockwise.
2. CircleByCenterPoint
   A circle defined by a center point and a radius.

The abstract type SegmentBase defines a sequence of control points and the attribute for the interpolation type:

```
<xs:complexType name="SegmentBase" abstract="true">
   <xs:sequence>
      <xs:element name="ControlPoint" type="Coordinate2D" minOccurs="1" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="interpolation" type="InterpolationType" use="required"/>
</xs:complexType>
```

From this type the type Segment is derived by restricting the number of control points to at least two:

```
<xs:complexType name="Segment">
   <xs:complexContent>
      <xs:restriction base="SegmentBase">
         <xs:sequence>
            <xs:element name="ControlPoint" type="Coordinate2D" minOccurs="2" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:restriction>
   </xs:complexContent>
</xs:complexType>
```

For the “by center point” segments the abstract base type is also derived from SegmentBase restricting the number of control points to exact 1 and fixes the value of the attribute interpolation.

```
<xs:complexType name="ArcByCenterPointBase" abstract="true">
   <xs:complexContent>
      <xs:restriction base="SegmentBase">
         <xs:sequence>
            <xs:element name="ControlPoint" type="Coordinate2D" minOccurs="1" maxOccurs="1"/>
         </xs:sequence>
         <xs:attribute name="interpolation" type="InterpolationType" use="required"
```

```
                              fixed="CircularArcCenterPointWithRadius"/>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
```

The ArcByCenterPoint is then an extension of this type adding attributes for radius, start angle and angular length.

```
<xs:complexType name="ArcByCenterPoint">
    <xs:complexContent>
        <xs:extension base="ArcByCenterPointBase">
            <xs:attribute name="radius" type="xs:double" use="required"/>
            <xs:attribute name="startAngle" type="xs:double" use="required"/>
            <xs:attribute name="angularDistance" type="xs:double" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The CircleByCenterPoint type is very similar. The attribute start angle is not defined since it is meaningless. The direction is here defined by the attribute direction which has values '+' or '-'.

```
<xs:simpleType name="Direction">
    <xs:restriction base="xs:string">
        <xs:enumeration value="+"/>
        <xs:enumeration value="-"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="CircleByCenterPoint">
    <xs:complexContent>
        <xs:extension base="ArcByCenterPointBase">
            <xs:attribute name="radius" type="xs:double" use="required"/>
            <xs:attribute name="direction" type="Direction" default="+"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

A group is defined that allows the use of the different type of segments.

```
    <xs:group name="Segments">
        <xs:choice>
            <xs:element name="Segment" type="Segment"/>
            <xs:element name="ArcByCenterPoint" type="ArcByCenterPoint"/>
            <xs:element name="CircleByCenterPoint" type="CircleByCenterPoint"/>
        </xs:choice>
    </xs:group>
```

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">
    <xs:complexContent>
        <xs:extension base="SpatialObject">
            <xs:sequence>
                <xs:element name="Boundary" type="BoundaryRelation" minOccurs="0" maxOccurs="2"/>
                <xs:group ref="Segments" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

9-7.7.6 CompositeCurve

A composite curve describes the line geometry of a feature object just like a 'simple' curve. But instead of 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">
  <xs:complexContent>
    <xs:extension base="SpatialObject">
      <xs:sequence>
        <xs:group ref="CurveRelations" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

9-7.7.7 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 `SpatialObject` 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">
  <xs:complexContent>
    <xs:extension base="SpatialObject">
      <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>
```

9-8 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.~~

9-9 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, as well as optional associations to information and / or other features.

When an association is present, the code of an information association or feature association defined in the feature catalogue should be used as the element name in place of *associatedInformation* or *associatedFeature* respectively - see Annex A for more information.

~~. 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.~~

Formatted: Font: Italic

Formatted: Indent: Left: 0"

```
<xs:complexType name="Feature" abstract="true">
    <xs:complexContent>
        <xs:extension base="Object">
            <xs:sequence>
                <xs:group ref="SpatialRelations" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="associatedInformation" type="InformationAssociation" minOccurs="0"
maxOccurs="unbounded"/>
              <!—When generating input XML the base schema should be extended to provide a substitution group for each
              information association defined in the feature catalogue. For example,  if the feature catalogue defines an
              information association having a code of AdditionalInformation, the base schema should be extended with:
                <xs:element name="AdditionalInformation" substitutionGroup="associatedInformation"/>
              -->
                <xs:element name="associatedFeature" type="FeatureAssociation" minOccurs="0"
                                          maxOccurs="unbounded"/>
              <!—When generating input XML the base schema should be extended to provide a substitution group for
              feature association defined in the feature catalogue. For example,  if the feature catalogue defines a feature
              association having a code of StructureEquipment, the base schema should be extended with:
                <xs:element name="StructureEquipment" substitutionGroup="associatedFeature"/>
              -->
            </xs:sequence>
            <xs:attribute name="primitive" type="GeometricPrimitive" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Formatted: Not Highlight

Formatted: Highlight

Commented [DMG6]: Add standard mechanism to describe associations.

~~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~~

Formatted: Left, Space After: 10 pt, Line spacing: Multiple 1.15 li

9-10 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.

XSLT 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>
```

9-11 Drawing Instructions

9-11.1 The concepts of drawing instructions

9-11.1.1 General concept

The output of the portrayal engine is a set of drawing instructions, which link the feature type to a symbol and/or alert 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.

9-11.1.2 Portrayal Coordinate Reference Systems (CRS)

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, for example a screen or pixmap.

Line symbols have two kinds of coordinate reference systems. The line coordinate reference system is a non-Cartesian 2-axis coordinate system. The x-axis is following the line geometry and the y-axis is perpendicular to the geometry of the curve. This CRS allows for the specification of line widths,offsets and symbols following the geometry. The second kind of coordinate reference system is a local Cartesian 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.

9-11.1.3 Viewing Groups, Viewing Group Layers and Display Mode

The viewing group is a concept to control the content of the display. It works 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 Viewing Group Layers and Viewing Group Layer can be aggregated into Display Modes. Both aggregations are part of the portrayal catalogue.

9-11.1.4 Transparency

Additive effects of applying transparencies shall be accomplished by multiplying the component alpha values. For example, a color token which has transparency of 10% which is drawn with a transparency of 20% shall result in (1 – 10%) * (1 – 20%) = 72% alpha = 28% transparency.

9-11.1.5 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.

9-11.1.6 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. Instructions which have equal display priority must be ordered so that area instructions are rendered first, followed by line instructions, then point instructions, and lastly text instructions. If the display priority is equal among the same type of instruction (area, line, point, or text) some other neutral criterion must be used to order the instructions.

9-11.1.7 Null Instruction

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

9-11.1.8 Point Instruction

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.

Point Geometry

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

MultiPoint Geometry

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

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.

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.

9-11.1.9 Line Instruction

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.

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.

9-11.1.10 Area Instruction

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.

9-11.1.11 Text Instruction

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.

Point Geometry

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

MultiPoint Geometry

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

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.

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.

9-11.1.12 Coverage Instruction

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

"A coverage is a feature that has multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type." [ISO 19123:2005, Introduction]

In this document coverage attributes used for portrayal are expected to have numeric values.

The assignment of Portrayal for a Coverage starts with a Coverage Feature. Like other Feature types a rule is used to match the Feature to Drawing instructions.

A first match lookup table is used to assign portrayal based on a specified coverage attribute. There are three options for coverage portrayal, filling with colour, annotating with numeric text or annotating with symbols.

Colour assignment

Colours are applied to a coverage by using a lookup table that matches a selected attribute value and specifies a colour. For a continuous coverage such as grid cells, pixels or tiles then each element is processed and colour filled with the appropriate colour. For a discrete coverage with distinct points colour is applied as a Pen Down or dott operation using the assigned pen width.

A lookup table entry can match a range of values and assign a single colour to that range or specify a start and end colour that is used to create a gradient or ramp effect as a linear interpolation of the value range across the colour range.

Numeric and Symbol Annotations

For a continuous coverage the centre of each cell (for example rectangle, tile, triangle) is used as the anchor point of the text or symbol.

For numeric annotations, overplot removal or collision avoidance is expected. A buffer can be used to provide some space between the annotations. A buffer of 0 means that direct overplot is used when digits interact. An enumeration called 'champion' is used to specify which annotation to keep (largest or smallest value) when an interaction occurs. For numeric annotations the text shall be placed such that the optical/geometric centre of the text represents the location.

For symbol annotations separate attributes from the coverage can be used to apply a scaling and rotation to the symbol. This can be useful for example when portraying a coverage that carries wave height and direction.

9-11.1.13 Augmented Geometry

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.

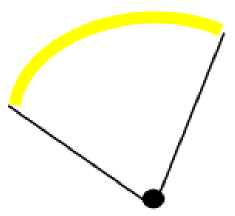


Figure 9-9 — Point Feature with Augmented geometries

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

9-11.2 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.

Commented [JW7]: Amendment required – refer to papers S-100WG5-04.05 and S-100WG5-04.05

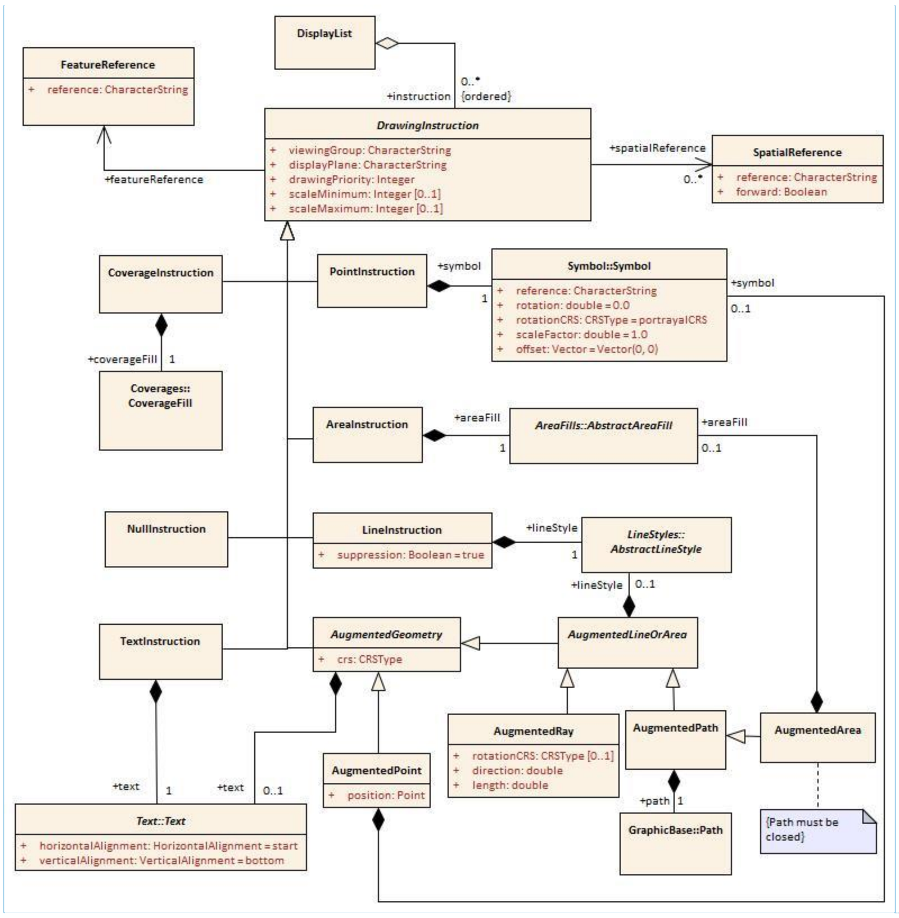


Figure 9-10 — Drawing Instructions

9-11.2.1 DisplayList

Role Name	Name	Description	Mult.	Type
Class	DisplayList	A ordered set of Drawing Instructions	-	-
Role	instruction	An instruction of this list	0..*	DrawingInstruction

9-11.2.2 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 Name	Name	Description	Mult.	Type
Attribute	scaleMinimum	Scale denominator to define the minimum scale for which the instruction will be shown. If not given there is no minimum scale	0..1	integer
Attribute	scaleMaximum	Scale denominator to define the maximum scale for which the instruction will be shown. If not given there is no maximum scale	0..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
Role	alertReference	The reference to the alert in the alert catalogue that is triggered by the geometry of the instruction	0..1	AlertReference

9-11.2.3 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

9-11.2.4 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

9-11.2.5 AlertReference

Role Name	Name	Description	Mult.	Type
Class	AlertReference	A reference to an alert in the alert catalogue	-	-
Attribute	reference	The identifier of the alert catalogue entry	1	string
Attribute	plan	The viewing group the alert highlight is assigned to when active in route planning	0..1	string
Attribute	monitor	The viewing group the alert highlight is assigned to when active in route monitoring	0..1	string

9-11.2.6 NullInstruction

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

9-11.2.7 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

9-11.2.8 LineInstruction

Role Name	Name	Description	Mult.	Type
Class	LineInstruction	A drawing instruction for line geometry	-	-
Attribute	suppression	Whether another line instruction of higher priority can suppress the drawing of this line instruction	1	boolean = *True*
Role	lineStyle	The line style used for the depiction	1	LineStyles::AbstractLineStyle

9-11.2.9 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::AbstractAreaFill

9-11.2.10 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

9-11.2.11 CoverageInstruction

Role Name	Name	Description	Mult.	Type
Class	CoverageInstruction	A drawing instruction for depicting coverages of data	-	-
Role	coverageFill	The coverage fill used for depiction	1	Coverages::CoverageFill

9-11.2.12 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

9-11.2.13 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

9-11.2.14 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

9-11.2.15 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	rotationCRS	If present, specifies the CRS for *direction*	0..1	GraphicsBase::CRSType
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

9-11.2.16 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

9-11.2.17 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

9-12 Symbol Definitions

9-12.1 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.

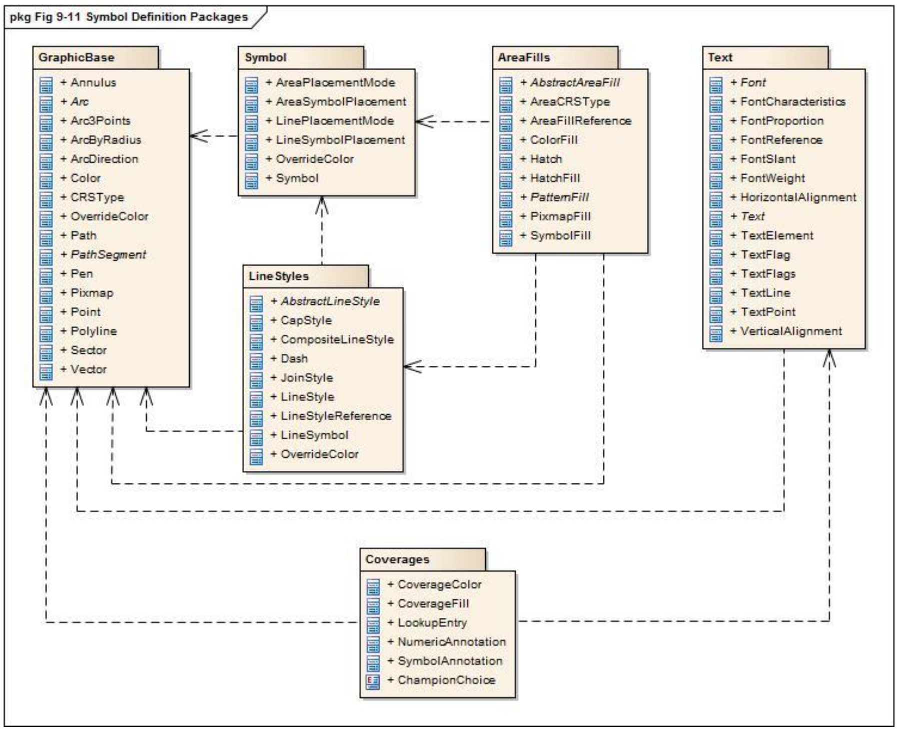


Figure 9-11 — Symbol Definition Packages

9-12.2 The GraphicBase package

9-12.2.1 Overview

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

9-12.2.2 Model

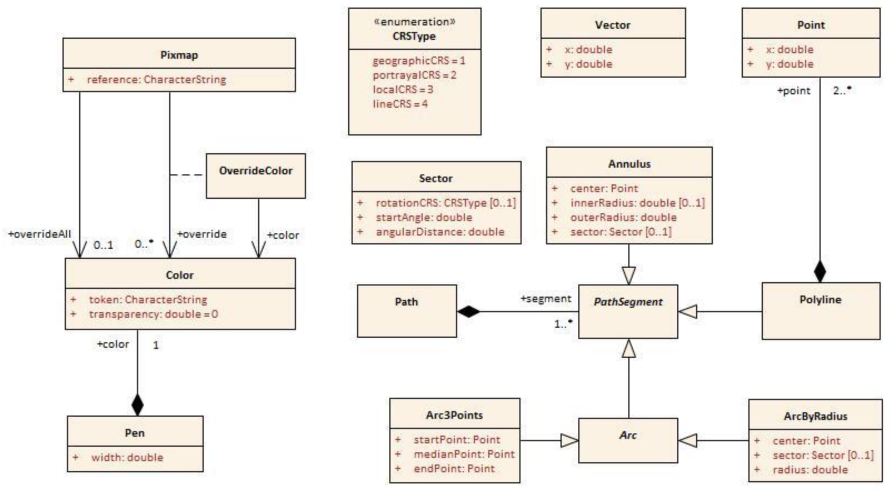


Figure 9-12 — Graphics Base

9-12.2.2.1 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

9-12.2.2.2 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

9-12.2.2.3 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

9-12.2.2.4 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

9-12.2.2.5 Pixmap

Role Name	Name	Description	Mult.	Type
Class	Pixmap	A two dimensional array of pixels defining an image	-	-
Attribute	reference	A reference to an external definition of the pixmap. This string is a unique identifier within the pixmap section of the portrayal catalogue	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

9-12.2.2.6 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	1	Color

		colour in the pixmap		

9-12.2.2.7 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 clockwise from the positive y-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. Units on the axes and for distances are millimetres. Angles are measured in degrees clockwise from the positive y-axis. See explanations for details
Enumeration	lineCRS	A none-Cartesian coordinate system where the x-axis is following the geometry of a curve and the y-axis is perpendicular to the x-axis (positive to the left of the x-axis). Units on the axes and for distances are millimetres. Angles are measured in degrees clockwise from the positive y-axis

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.

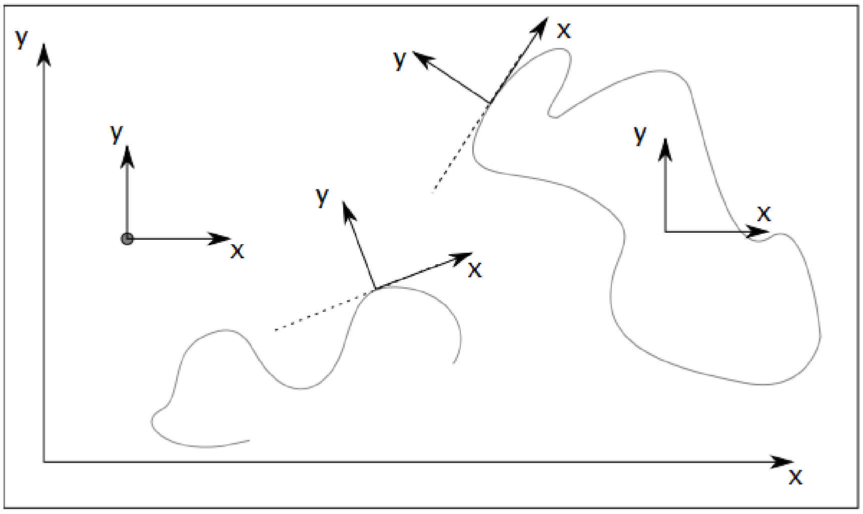


Figure 9-13 — Local CRS

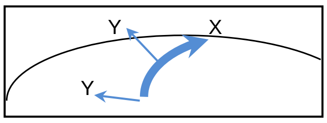


Figure 9-14 — Line CRS

9-12.2.2.8 Sector

Role Name	Name	Description	Mult.	Type
Class	Sector	Region of the Cartesian plane enclosed by two radii	-	-
Attribute	rotationCRS	If present, specifies the CRS for startAngle	0..1	CRSType
Attribute	startAngle	The direction of the radius that defines the beginning of the sector	1	double
Attribute	angularDistance	The angular distance of the sector measured in degrees. Positive values means clockwise, negative values means anti-clockwise	1	double

9-12.2.2.9 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.

9-12.2.2.10 PathSegment

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

9-12.2.2.11 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

9-12.2.2.12 Arc

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

9-12.2.2.13 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	-	-

Role Name	Name	Description	Mult.	Type
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

9-12.2.2.14 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

9-12.2.2.15 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

9-12.3 The Symbol package

9-12.3.1 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.

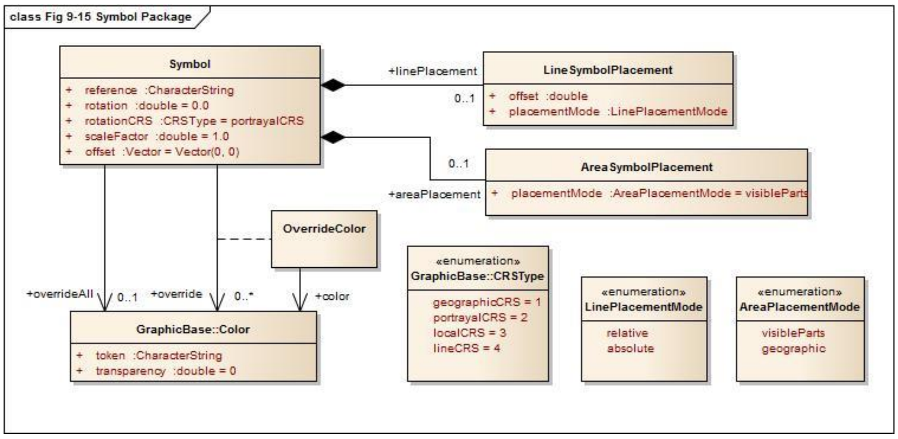


Figure 9-15 — Symbol Package

9-12.3.1.1 Symbol

Role Name	Name	Description	Mult.	Type
Class	Symbol	A two dimensional graphical element	-	-
Attribute	reference	A reference to an external definition of the symbol graphic. This is an unique identifier in the symbol section of the portrayal catalogue	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

9-12.3.1.2 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

9-12.3.1.3 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

9-12.3.1.4 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

9-12.3.1.5 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

9-12.3.1.6 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

9-12.4 The LineStyles package

9-12.4.1 Model

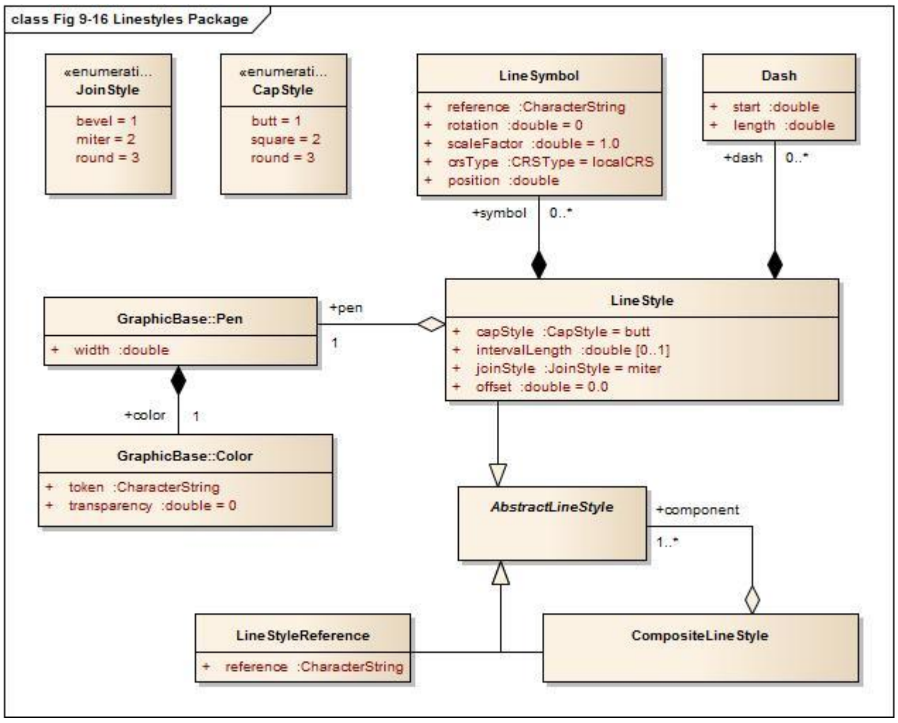


Figure 9-16 — Line Styles Package

9-12.4.1.1 AbstractLineStyle

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

9-12.4.1.2 LineStyle

Role Name	Name	Description	Mult.	Type
Class	LineStyle	A style for line geometry either solid or dashed	-	-
Attribute	offset	An offset perpendicular to the direction of the line. The value refers to the y-axis of the line CRS (positive to the left, mm)	1	double
Attribute	capStyle	The decoration that is applied where a line segment ends	1	CapStyle

Role Name	Name	Description	Mult.	Type
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 along the x-axis of the line CRS (units 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
Role	symbol	Symbols placed along the line	0..*	LineSymbol

9-12.4.1.3 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,along the x-axis of the line CRS (units in mm)	1	double
Attribute	length	The length of the dash along the x-axis of the line CRS (units in mm)	1	double

9-12.4.1.4 LineSymbol

Role Name	Name	Description	Mult.	Type
Class	LineSymbol	A symbol placed along a line in a repeating pattern	-	-
Attribute	reference	A reference to an external definition of the symbol graphic. This refers to an identifier of a catalogue item	1	string
Attribute	rotation	The rotation angle of the symbol. The default value is 0	1	double
Attribute	scaleFactor	The scale factor of the symbol. The default is 1.0	1	double
Attribute	crsType	The type of the CRS where the symbol has to be transformed to. Possible values are localCRS and lineCRS	1	CRSType
Attribute	position	The position of the symbol measured from the start of the repeating interval,along the x-axis of the line CRS (units in mm)	1	double

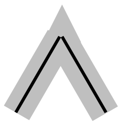
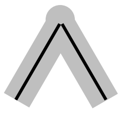
9-12.4.1.5 CompositeLineStyle

Role Name	Name	Description	Mult.	Type
Class	CompositeLineStyle	A line style made with an aggregation of other line styles		
Role	component	The components of the composite line style	1..*	AbstractLineStyle

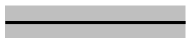
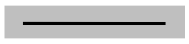
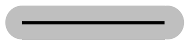
9-12.4.1.6 LineStyleReference

Role Name	Name	Description	Mult.	Type
Class	LineStyleReference	A line style defined in an external file		
Attribute	reference	The reference to the external definition of the line style. This is an unique identifier in the line style section of the Portrayal Catalogue	1	string

9-12.4.1.7 JoinStyle

Role Name	Name	Description
Type	JoinStyle	The decoration that is applied where two line segments meet
Enumeration	bevel	
Enumeration	miter	
Enumeration	round	

9-12.4.1.8 CapStyle

Role Name	Name	Description
Type	CapStyle	The decoration that is applied where a line segment ends.
Enumeration	butt	
Enumeration	square	
Enumeration	round	

9-12.5 The AreaFills package

9-12.5.1 Model

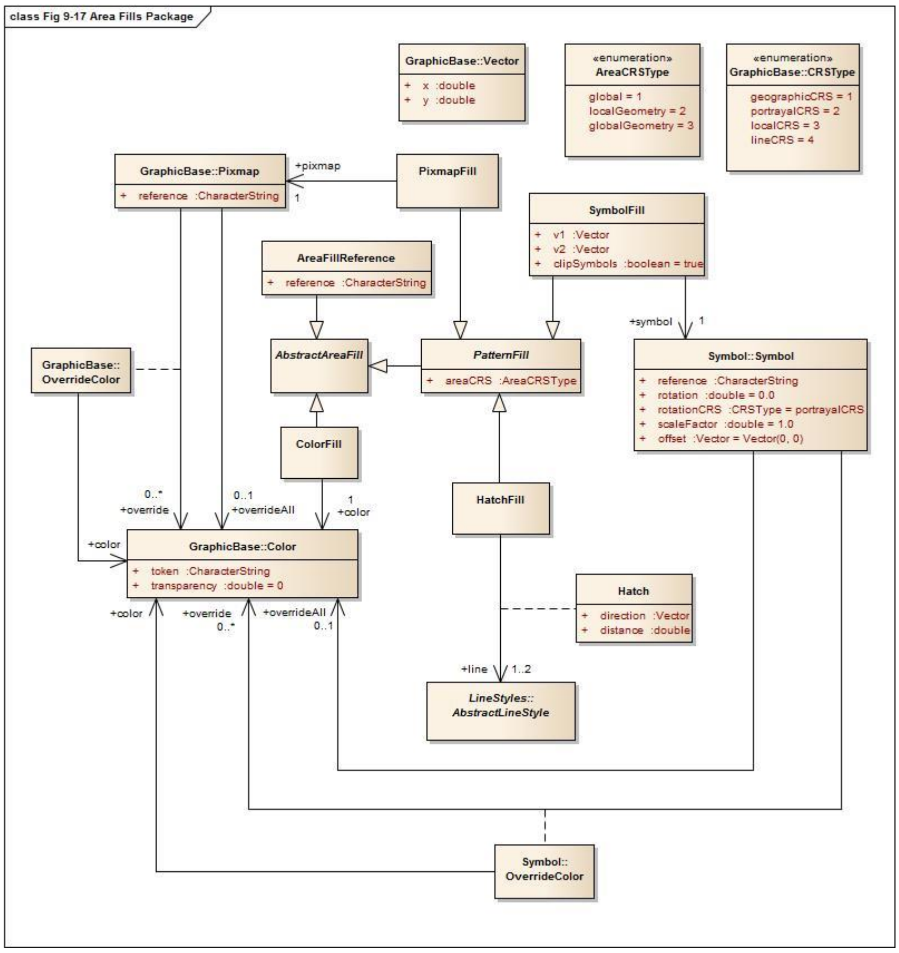


Figure 9-17 — Area Fills Package

9-12.5.1.1 AbstractAreaFill

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

9-12.5.1.2 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

9-12.5.1.3 AreaFillReference

Role Name	Name	Description	Mult.	Type
Class	AreaFillReference	An area fill defined in an external file	-	-
Attribute	reference	The reference to the external definition. This is an unique identifier in the area fill section of the Portrayal Catalogue	1	string

9-12.5.1.4 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

9-12.5.1.5 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

9-12.5.1.6 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 according to the local CRS	1	Vector
Attribute	v2	Defines the offset of the next symbol in the second dimension of the pattern according to the local CRS	1	Vector
Attribute	clipSymbols	Indicates whether the symbols in the pattern are to be clipped by the area (when they are part in/out of the area) or whether the symbol is not drawn at all unless it is completely contained in the area	1	boolean *True*: Fill symbols are clipped at area boundaries *False*: Fill symbols extending over the area boundaries are not drawn at all

9-12.5.1.7 HatchFill

Role Name	Name	Description	Mult.	Type
Class	HatchFill	Defining a pattern made of one or two sets of parallel lines	-	-
Association	Hatch	A set of parallel lines	1	Hatch

9-12.5.1.8 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	1	double

		perpendicular to the direction		
Role	line	The line style used for each hatch line	1..2	LineStyles:: AbstractLineStyle

9-12.5.1.9 AreaCRSType

Role Name	**Name**	**Description**
Type	PatternCRS	Describes how a fill patter is referenced
Enumeration	global	Anchor point is consistent with a location on the drawing device for, example starting with the corner of the screen. As screen pans the pattern will appear to shift/move through the object on screen
Enumeration	localGeometry	Anchor point is consistent with the local geometry of the object being depicted, for example the upper left corner of the object. Patterns of adjacent objects may not match
Enumeration	globalGeometry	The anchor point of the fill pattern is defined at a common location such that patterns remain consistent relative to all area objects

9-12.6 The Text package

9-12.6.1 Overview

The text package contains the types necessary for the depiction of text. This includes fonts. In this model fonts may be described by characteristics or referenced by name. Two types of text instructions are supported:

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

9-12.6.2 Fonts

A font is a set of typefaces. A typeface is the artistic representation or interpretation of characters; it is the way the type looks.

This standard supports two methods of defining fonts, the first describes a font by four attributes and let the system find a best match to an actual font available on the graphic system. The second method is referencing an external font file. The format of this file must conform to the 'True Type Font' standard and must be included in the the Portrayal Catalogue.

9-12.6.3 Model

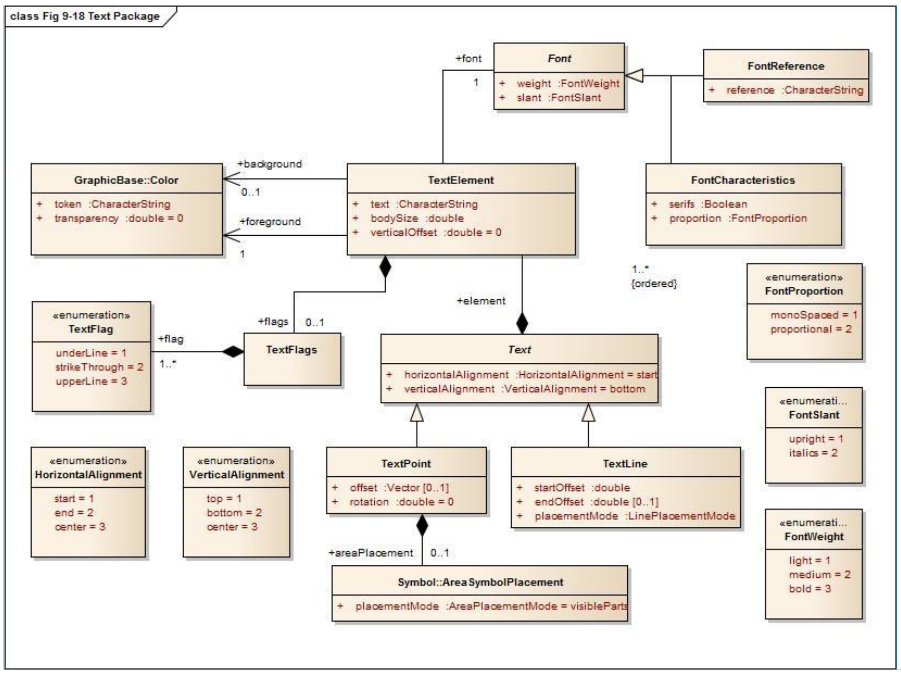


Figure 9-18 — Text Package

Commented [JW8]: Table rquires updating. Refer to paper S-100WG5-04.13B. (Raphael)

9-12.6.3.1 Font

Role Name	Name	Description	Mult.	Type
Class	Font	Abstract base class for fonts	-	-

9-12.6.3.2 FontCharacteristics

Role Name	Name	Description	Mult.	Type
Class	FontCharacteristics	Class describing the main characteristics of a font	-	-
Attribute	serifs	Describes whether the typefaces contain serifs or not	1	boolean
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

9-12.6.3.3 FontReference

Role Name	Name	Description	Mult.	Type
Class	FontReference	Class referencing a font from an external source	-	-
Attribute	reference	The identifier for the external file within the portrayal catalogue	1	string

9-12.6.3.4 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

9-12.6.3.5 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

9-12.6.3.6 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

9-12.6.3.7 TextFlags

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

9-12.6.3.8 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

9-12.6.3.9 FontSlant

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

9-12.6.3.10 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**)

9-12.6.3.11 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

9-12.6.3.12 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

9-12.6.3.13 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

9-12.6.3.14 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

9-12.7 The Coverage package

9-12.7.1 Overview

The coverage package contains the types for the depiction of a Coverage. This portrayal is applicable to the portrayal of numeric Coverage values. Three types of coverage portrayals are supported:

- Colour;
- Numeric Annotation; and
- Symbol Annotation.

9-12.7.2 Ranges

Ranges are used to control how portrayal is assigned to the values in a Coverage. These make use of the S-100_NumericRange complex type which is defined in S-100 Part 1 Conceptual Schema Language. The Numeric Range type allows for various range definitions with different closure options.

9-12.7.3 Lookup Table

The CoverageFill class carries an ordered list of lookup entries. Each of these entries carries a range used to evaluate a match by testing if the coverage value matches the range. The first lookup entry with a matching range is used to apply up to one of each type of portrayal (colour, numeric annotation or a symbol) to the coverage element. This allows for example to fill a cell in a grid with a colour and assign a numeric or symbol annotation to the cell as well.

The lookup table can also associate an alert specified in the drawing instruction with a range of coverage values. When associating alerts with coverage values there may or may not be portrayal elements (colour, numeric annotation or a symbol) present.

Commented [JW9]: Amendment required. Refer to paper S-100WG5-04.05.

9-12.7.4 Model

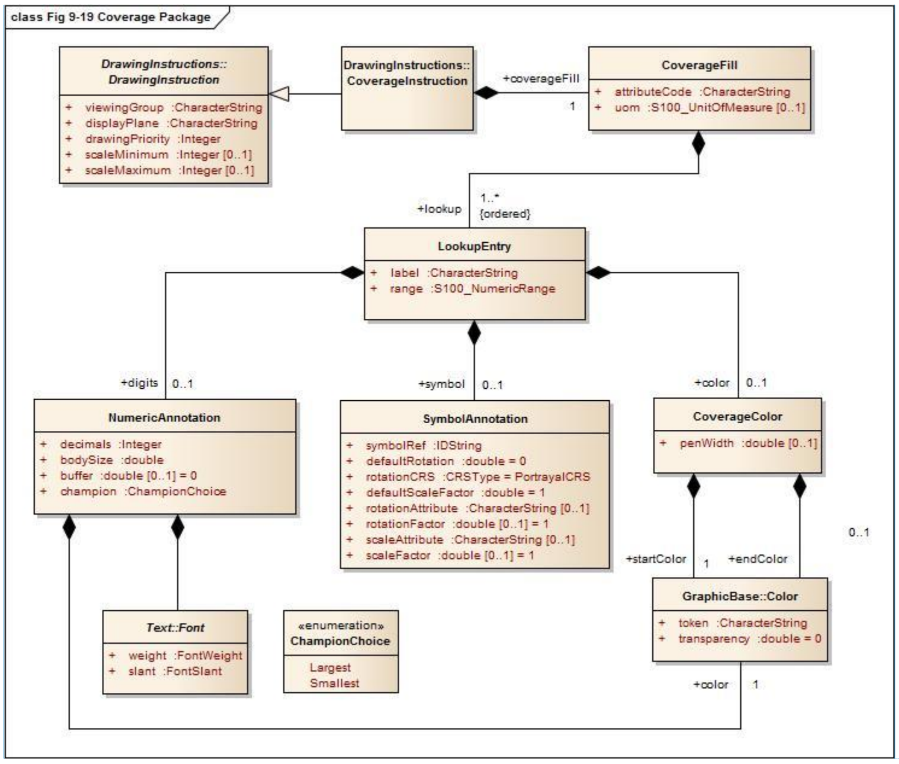

class Fig 9-19 Coverage Package
DrawingInstructions::DrawingInstruction
+ viewingGroup :CharacterString
+ displayPlane :CharacterString
+ drawingPriority :Integer
+ scaleMinimum :Integer [0..1]
+ scaleMaximum :Integer [0..1]
DrawingInstructions::CoverageInstruction
+coverageFill
1
CoverageFill
+ attributeCode :CharacterString
+ uom :S100_UnitOfMeasure [0..1]
+lookup
1..* {ordered}
LookupEntry
+ label :CharacterString
+ range :S100_NumericRange
+digits
0..1
+symbol
0..1
+color
0..1
NumericAnnotation
+ decimals :Integer
+ bodySize :double
+ buffer :double [0..1] = 0
+ champion :ChampionChoice
SymbolAnnotation
+ symbolRef :IDString
+ defaultRotation :double = 0
+ rotationCRS :CRSType = PortrayalCRS
+ defaultScaleFactor :double = 1
+ rotationAttribute :CharacterString [0..1]
+ rotationFactor :double [0..1] = 1
+ scaleAttribute :CharacterString [0..1]
+ scaleFactor :double [0..1] = 1
CoverageColor
+ penWidth :double [0..1]
0..1
+startColor
1
+endColor
GraphicBase::Color
+ token :CharacterString
+ transparency :double = 0
Text::Font
+ weight :FontWeight
+ slant :FontSlant
«enumeration» ChampionChoice
Largest
Smallest
+color
1

Figure 9-19 — Coverage Package

9-12.7.4.1 CoverageFill

Role Name	Name	Description	Mult.	Type
Class	CoverageFill	A class to fill a Coverage with using a lookup table to match a value or range of values and assign colour, numeric or symbol annotations	-	-
Attribute	attributeCode	Code of coverage attribute value to match	1	characterString
Attribute	uom	Unit of measure. If not given the values in the range are assumed to be same units as the coverage attribute values	0..1	S100_UnitOfMeasure
Role	lookup	Lookup table. The entries are ordered and processed on a first match basis	1..*	LookupEntry

9-12.7.4.2 LookupEntry

Role Name	Name	Description	Mult.	Type
Class	LookupEntry	An entry in a lookup table used to assign portrayal to coverage elements.	-	-
Attribute	label	String used as a display label or legend field.	1	characterString
Attribute	range	Value range definition. Can be a single value, open or closed range etc. See S-100 Part 1 Conceptual Schema Language for details.	1	S100_NumericRange
Role	color	The color to assign to the matching range. Can be a single color or a color ramp.	0..1	CoverageColor

Role Name	Name	Description	Mult.	Type
Role	digits	Display the value as numeric digits.	0..1	NumericAnnotation
Role	symbol	Display a symbol.	0..1	SymbolAnnotation

9-12.7.4.3 CoverageColor

Role Name	Name	Description	Mult.	Type
Class	CoverageColor	A class to fill a Coverage with color	-	-
Attribute	penWidth	Optional pen width to apply for dot color used for discrete points	0..1	double
Role	startColor	The color to assign to the matching range or to use as start point in a color ramp when 'endColor' is defined	1	GraphicBase::Color
Role	endColor	The color to use as stop point in a color ramp. The range of values is spread linearly across the range of colors from 'startColor' to 'endColor' to produce a gradient effect	0..1	GraphicBase::Color

9-12.7.4.4 NumericAnnotation

Role Name	Name	Description	Mult.	Type
Class	NumericAnnotation	A class for numeric textual annotations of values in a Coverage	-	-
Attribute	decimals	Number of decimal digits to show in subscript	1	integer
Attribute	bodySize	This property describes the size with which the text will be depicted	1	double
Attribute	buffer	Buffer to apply for collision detection in presentation units. Default=0	1	double
Attribute	champion	Enumeration to indicate which value to display in the event of a collision	1	ChampionChoice
Role	font	Font information to use for display of numeric values across a coverage. Text::Font is a choice of either FontCharacteristics or FontReference	1	Text::Font
Role	color	Color to draw the numeric annotation	1	GraphicBase::Color

9-12.7.4.5 SymbolAnnotation

Role Name	Name	Description	Mult.	Type
Class	SymbolAnnotation	A class for symbol annotations of values in a coverage.	-	-
Attribute	symbolRef	Reference to the symbol to apply. Catalogue id.	1	IDString
Attribute	defaultRotation	A default symbol rotation. Applies when rotation attribute not defined. Default=0	0..1	double
Attribute	rotationCRS	Specifies the coordinate reference system for the rotation. Default=PortrayalCRS	1	GraphicsBase::CRSType
Attribute	defaultScale	A default symbol scale factor. Applies when scale attribute not defined. Default=1	1	double
Attribute	rotationAttribute	The attribute code of the Coverage Attribute to use for the symbol rotation value.	0..1	characterString
Attribute	rotationFactor	Used to adjust the 'rotationAttribute' value by multiplication before applying. Default 1.0	0..1	double
Attribute	scaleAttribute	The attribute code of the Coverage attribute to use for scaling the symbol size.	0..1	characterString
Attribute	scaleFactor	Used to adjust the ' scaleAttribute' value by multiplication before applying. Default 1.0	0..1	double

For schema definition see 9-A-2 Symbol Definition Schema

Commented [TS10]: For S-100WG: Suggest this paragraph can be removed, as the Annex A as included in Edition 4.0.0 has been removed (refer paper S-100WG5-04.10 – Part 9 Redlines).

9-13 The portrayal library

9-13.1 Overview

- Machine readable.
- A file/directory structure with a catalogue file.
- Files for pixmaps, symbols, complex line styles, areafills, fonts and colour profiles.
- Alert information in a separate file.
- Portrayal rules in separate files.
- Model and schema for the catalogue included.

9-13.2 Structure

```
Root ---- (contains the catalogue named "portrayal_catalogue.xml", and optionally an alert catalogue file)
    |
    |-- Pixmaps (contains XML files describing pixmaps)
    |
    |-- ColorProfiles (contains XML files with colour profiles)
    |
    |-- Symbols (contains SVG files with symbols and CSS2 style sheets)
    |
    |-- LineStyles (contains XML files with line styles)
    |
    |-- AreaFills (contains XML files area fills)
    |
    |-- Fonts (contains TrueType font files)
    |
    |-- Rules (contains files with rules which map features to drawing instructions)
```

9-13.3 Model of the Catalogue

Commented [TS11]: This Figure needs to be updated for amendments approved at S-100WG4 (refer paper 04.06) and S-100WG5 (refer papers 04.04 and 04.05). Raphael to supply?

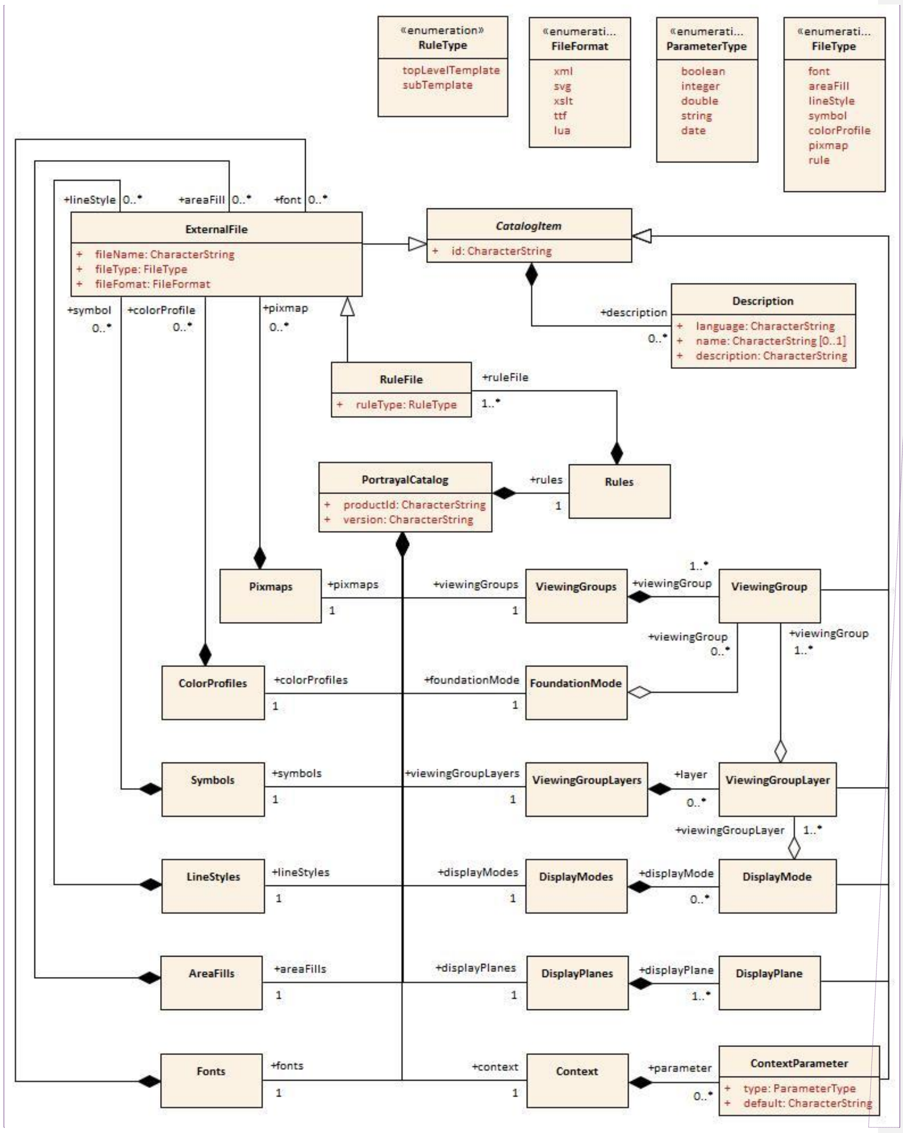


Figure 9-20 — Catalogue

9-13.3.1 PortrayalCatalog

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 intended	1	string
Attribute	version	The version of the product the Catalogue is defined for	1	string
Role	alertCatalog	A file reference to an alert catalogue	0..1	ExternalFile

Role Name	Name	Description	Mult.	Type
Role	pixmaps	Container of XML Pixmap file references	1	Pixmaps
Role	colorProfiles	Container of XML Colour Profile file references	1	ColorProfiles
Role	symbols	Container of SVG Symbol file references	1	Symbols
Role	styleSheets	Container of CSS file references	1	StyleSheets
Role	lineStyles	Container of XML Line Style file references	1	LineStyles
Role	areaFills	Container of XML Area Fill file references	1	AreaFills
Role	fonts	Container of True Type font references	1	Fonts
Role	viewingGroups	Container of viewing group definitions	1	ViewingGroups
Role	foundationMode	The definition of the foundation of the portrayal	1	FoundationMode
Role	viewingGroupLayers	Container of viewing group layers.	1	ViewingGroupLayers
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 rule file references	1	Rules

9-13.3.2 CatalogItem

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

9-13.3.3 ExternalFile

Role Name	Name	Description	Mult.	Type
Class	ExternalFile	A catalogue item that defines the reference to an external file	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	fileName	The name of the file	1	string
Attribute	fileType	The type of the file	1	FileType
Attribute	fileFormat	The format of the file	1	FileFormat

9-13.3.4 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

9-13.3.5 Pixmaps

Role Name	Name	Description	Mult.	Type
Class	Pixmaps	A container of pixmap file references	-	-
Role	pixmap	The file reference. The type is XML	0..*	ExternalFile

9-13.3.6 ColorProfiles

Role Name	Name	Description	Mult.	Type
Class	ColorProfiles	A container of colour profile file references	-	-
Role	colorProfile	The file reference. The type is XML	0..*	ExternalFile

9-13.3.7 Symbols

Role Name	Name	Description	Mult.	Type
Class	Symbols	A container of Symbol file references	-	-
Role	symbol	The file reference. The type is SVG	0..*	ExternalFile

9-13.3.8 StyleSheets

Role Name	Name	Description	Mult.	Type
Class	StyleSheets	A container of CSS file references	-	-
Role	styleSheet	The file reference	0..*	ExternalFile

9-13.3.9 LineStyles

Role Name	Name	Description	Mult.	Type
Class	LineStyles	A container of Line Style file references	-	-
Role	lineStyle	The file reference. The type is XML	0..*	ExternalFile

9-13.3.10 AreaFills

Role Name	Name	Description	Mult.	Type
Class	AreaFills	A container of Area Fill file references	-	-
Role	lineStyle	The file reference. The type is XML.	0..*	ExternalFile

9-13.3.11 Fonts

Role Name	Name	Description	Mult.	Type
Class	Fonts	A container for fonts	-	-
Role	font	The file reference. For true type fonts the type is ttf.	0..*	ExternalFile

9-13.3.12 ViewingGroups

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

9-13.3.13 ViewingGroup

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

9-13.3.14 FoundationMode

Role Name	Name	Description	Mult.	Type
Class	FoundationMode	A set of viewing groups that forms the foundation of the portrayal and cannot be removed from the display	-	-
Role	viewingGroup	Viewing group of the foundation mode	0..*	ViewingGroup

9-13.3.15 ViewingGroupLayers

Role Name	Name	Description	Mult.	Type
Class	ViewingGroupLayers	A container of Viewing Group Layers	-	-
Role	layer	Definition of a specific Viewing Group layer	0..*	ViewingGroupLayer

9-13.3.16 ViewingGroupLayer

Role Name	Name	Description	Mult.	Type
Class	ViewingGroupLayer	A set of Viewing groups which are intended to switch on or off in an application	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Role	viewingGroup	Viewing Group of the layer	1..*	ViewingGroup

9-13.3.17 DisplayModes

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

9-13.3.18 DisplayMode

Role Name	Name	Description	Mult.	Type
Class	DisplayMode	A set of Viewing Layers to switch on or off in an application	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Role	viewingGroupLayer	Viewing Group Layer included in this Display Mode	1..*	ViewingGroupLayer

9-13.3.19 DisplayPlanes

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

9-13.3.20 DisplayPlane

Role Name	Name	Description	Mult.	Type
Class	DisplayPlane	A Display Plane name and definition	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	order	Used to sort the drawing order of display planes. Display planes with larger values are drawn above those with lower values Positive: Above RADAR Zero: Reserved for RADAR Negative: Below RADAR	1	integer

Commented [JW12]: Refer S-100WG5-04.04

9-13.3.21 Context

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

9-13.3.22 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

9-13.3.23 Rules

Role Name	Name	Description	Mult.	Type
Class	Rules	A container of XSLT rule file references	-	-
Role	ruleFile	Reference to a file containing rules	1..*	RuleFile

9-13.3.24 RuleFile

Role Name	Name	Description	Mult.	Type
Class	RuleFile	Rule file reference	-	-
Subtype of	ExternalFile	See ExternalFile	-	-
Attribute	ruleType	The type of the templates within the rule file. There can be more than one top level rule which can be selected in an application to allow different portrayal of the data	1	RuleType

9-13.3.25 ParameterType

Role Name	Name	Description
Type	ParameterType	Choice of Parameter Types
Enumeration	boolean	A Boolean value
Enumeration	integer	An integer number
Enumeration	double	A floating point number
Enumeration	string	A character string
Enumeration	date	A date according to the gregorian calendar

9-13.3.26 FileFormat

Role Name	Name	Description
Type	FileFormat	The format of an external file
Enumeration	xml	
Enumeration	svg	
Enumeration	xslt	
Enumeration	ttf	
Enumeration	lua	
Enumeration	css	

Commented [JW13]: Refer to paper S-100WG5-04.13B.

9-13.3.27 FileType

Role Name	Name	Description
Type	FileType	The type of an external file
Enumeration	font	A font file
Enumeration	areaFill	A file describing an area fill
Enumeration	lineStyle	A file describing a line style
Enumeration	symbol	A file describing a symbol
Enumeration	colorProfile	A file describing a colour profile
Enumeration	pixmap	A file describing a pixmap
Enumeration	rules	A file containing portrayal rules
Enumeration	styleSheet	A file containing styles for symbols
Enumeration	alertCatalog	A file containing an alert catalogue

9-13.3.28 RuleType

Role Name	Name	Description
Type	RuleType	The type of templates within a rule file
Enumeration	topLevelTemplate	The rule file contains a top level template
Enumeration	subTemplate	The rule file contains templates that are used or called by other templates

For schema definition see 9-A-5 Portrayal Catalogue Schema

Commented [TS14]: For S-100WG: Suggest this paragraph can be removed, as the Annex A as included in Edition 4.0.0 has been removed (refer paper S-100WG5-04.10 – Part 9 Redlines).

Commented [DMG15R14]: This part should identify where the schemas can be retrieved from.

9-13.4 Model of the alert catalogue

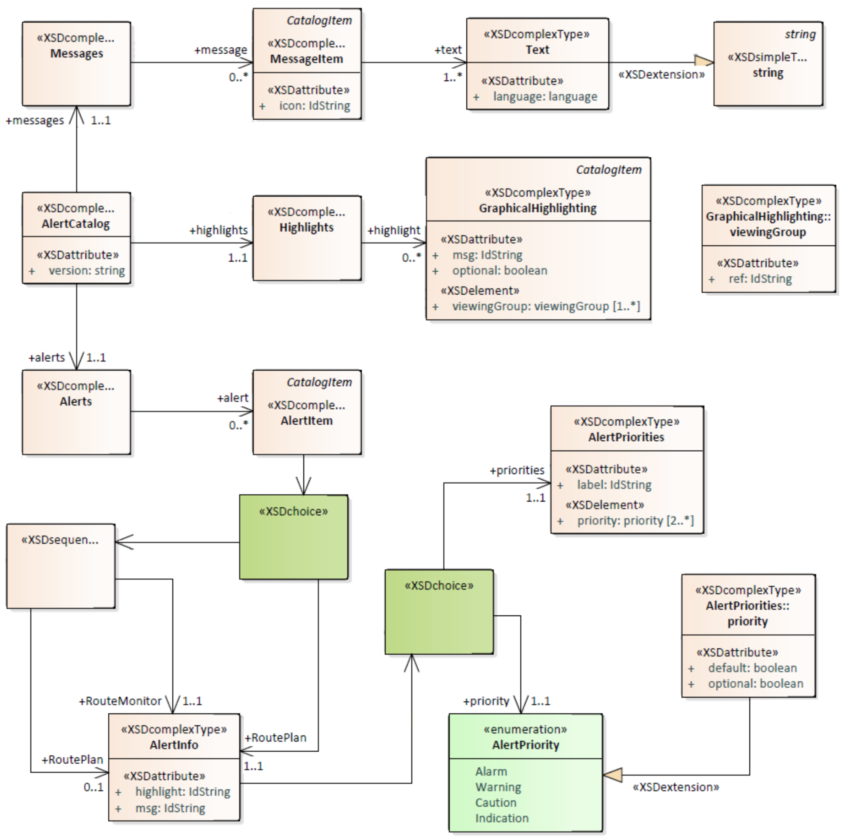


Figure 9-21 — Alert Catalogue

9-13.4.1 AlertCatalog

Role Name	Name	Description	Mult.	Type
Class	AlertCatalog	A container of all the Catalogue items	-	-
Attribute	version	The version of the catalogue	1	string
Role	messages	Container of messages	1	Messages
Role	highlights	Container of highlights	1	Highlights
Role	alerts	Container of alerts	1	Alerts

9-13.4.2 Messages

Role Name	Name	Description	Mult.	Type
Class	Messages	A container of MessageItems	-	-
Role	message	Definition of a message	0..*	MessageItem

9-13.4.3 MessageItem

Role Name	Name	Description	Mult.	Type
Class	MessageItem	Defines a language independent message	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	icon	Reference to a symbol in the portrayal catalogue	0..1	string
Role	text	Language specific text	1..*	Text

9-13.4.4 Text

Role Name	Name	Description	Mult.	Type
Class	Text	Language specific string	-	-
Subtype of	string		-	-
Attribute	language	Identifies a language, default is eng. ISO 639-2/T alpha-3 code (eng – English, deu – German)	0..1	string

9-13.4.5 Highlights

Role Name	Name	Description	Mult.	Type
Class	Highlights	A container of GraphicalHighlighting items	-	-
Role	highlight	Definition of a highlight	0..*	GraphicalHighlighting

9-13.4.6 GraphicalHighlighting

Role Name	Name	Description	Mult.	Type
Class	GraphicalHighlighting	Associates viewing groups with alert information	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Attribute	optional	Allowing the highlight to be turned off is not required. Default is false	0..1	boolean
Attribute	msg	A reference to a message to be displayed while any of the viewing groups are disabled	0..1	string
Role	viewingGroup	References viewing groups used to control graphical highlighting	1..*	ViewingGroupReference

9-13.4.7 ViewingGroupReference

Role Name	Name	Description	Mult.	Type
Class	ViewingGroupReference	A reference to a viewing group	-	-
Role	ref	The identifier of the viewing group	1	string

9-13.4.8 Alerts

Role Name	Name	Description	Mult.	Type
Class	Alerts	A container of AlertItems	-	-
Role	alert	Definition of an alert	0..*	AlertItem

9-13.4.9 AlertItem

Role Name	Name	Description	Mult.	Type
Class	AlertItem	Describes a single alert	-	-
Subtype of	CatalogItem	See CatalogItem	-	-
Role	routeMonitor	The alert behavior in route monitoring	0..1	AlertInfo
Role	routePlan	The alert behavior in route planning	0..1	AlertInfo

9-13.4.10 AlertInfo

Role Name	Name	Description	Mult.	Type
Class	AlertInfo	The behavior of an alert in a single mode	-	-
Attribute	routeMonitor	The alert behavior in route monitoring	0..1	AlertInfo
Attribute	routePlan	The alert behavior in route planning	0..1	AlertInfo
Role	priority	A single alert priority. If present, precludes use of *priorities*	0..1	AlertPriority
Role	priorities	A set of alert priorities. If present, precludes use of *priority*	0..1	AlertPriorities

9-13.4.11 AlertPriorities

Role Name	Name	Description	Mult.	Type
Class	AlertPriorities	A set of alert priorities	-	-
Attribute	label	Reference to a message used to label the UI component which allows selection of the desired alert priority	1	string
Role	priority	An alert priority	2..*	AlerPrioritySelection

9-13.4.12 AlertPrioritySelection

Role Name	Name	Description	Mult.	Type
Class	AlertPrioritySelection	Adds information to an alert priority	-	-
Subtype of	AlertPriority	See AlertPriority	-	-
Attribute	default	Identifies the default priority selection. Default is false	0..1	boolean
Attribute	optional	Indicates allowing the user to choose this priority is optional. Default is false	0..1	boolean

9-13.4.13 AlertPriority

Role Name	Name	Description
Type	AlertPriority	The priority of an alert
Enumeration	Alarm	Indicates conditions requiring immediate attention and action by the bridge team (refer to MSC.252(83) 19.1.2)
Enumeration	Warning	Indicates changed conditions and should be presented for precautionary reasons which are not immediately hazardous but which may become so, if no action is taken (refer to MSC.252(83) 19.1.3)
Enumeration	Caution	Indicates a condition which does not warrant an alarm or warning condition, but still requires attention and out of the ordinary consideration of the situation or of given information (refer to MSC.252(83) 19.1.4)
Enumeration	Indication	Display of regular information and conditions (refer to MSC.252(83) appendix 1)

9-13.5 Schema for pixmap files

A pixmap is a two dimensional array of pixels defining an image. This schema allows to encode pixmaps that can be then be referenced, for example from pixmap area fills. The coordinate system for the pixmap is different than other coordinate systems in this standard. The y-axis is directed downwards and the origin is in the upper left corner of the pixmap.

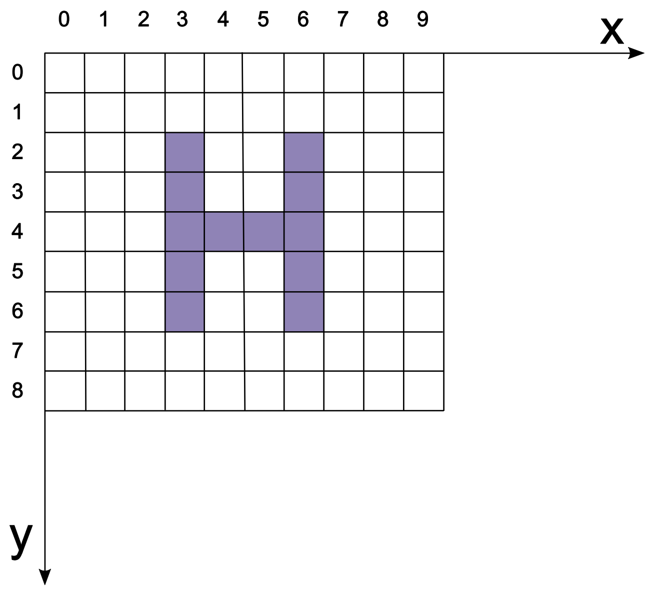


Figure 9-22 — Coordinate system for pixmap

The graphic above shows a simple pixmap with width 10 pixel and height 9 pixel. Most of the pixels are transparent (here white) some pixels are coloured.

The style defines a simple type for the colour identifier:

```
<xs:simpleType name="ColorId">
   <xs:restriction base="xs:string">
      <xs:minLength value="1"></xs:minLength>
      <xs:maxLength value="3"></xs:maxLength>
      <xs:pattern value="[a-zA-Z0-9_]+"></xs:pattern>
   </xs:restriction>
</xs:simpleType>
```

This describes a token 1 to 3 characters long that can contain digits, alpha characters or the underscore. It is used to identifiy a colour in the colour map.

The next type is a complex type for a pixel:

```
<xs:complexType name="Pixel">
   <xs:simpleContent>
      <xs:extension base="ColorId">
         <xs:attribute name="x" type="xs:nonNegativeInteger" use="required"/>
         <xs:attribute name="y" type="xs:nonNegativeInteger" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

It extends the colour identifier and adds two attributes for the coordinate of the pixel according to the pixmap coordinate system.

Each pixmap contains a colour map; a list of colour definitions bundled with a colour identifier. Two types are defined in the schema one for the colour map item and one for the colour map.

```
<xs:complexType name="ColorMapItem">
   <xs:complexContent>
      <xs:extension base="s100Symbol:Color">
         <xs:attribute name="id" type="ColorId" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="ColorMap">
   <xs:sequence>
      <xs:element name="color" type="ColorMapItem" minOccurs="1" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

Note: The color definition is taken from the S-100 symbol definition schema. That allows using colour token from a colour profile or direct sRGB colour definitions. Transparency can be defined here as well.

The last type defined is the complex type for the pixmap itself.

```
<xs:complexType name="Pixmap">
   <xs:sequence>
      <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="width" type="xs:positiveInteger"/>
      <xs:element name="height" type="xs:positiveInteger"/>
      <xs:element name="colorMap" type="ColorMap">
         <xs:key name="colorKey">
            <xs:selector xpath="color"/>
            <xs:field xpath="@id"/>
         </xs:key>
      </xs:element>
      <xs:element name="background" type="ColorId"/>
      <xs:element name="pixel" type="Pixel" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

It defines an optional description element and mandatory elements for width and height. Furthermore it defines an element for the colour map, an element for the background colour and the any number of pixel elements. The background colour is implicitly used for all pixels that are not defined by a pixel element. Note that there is a key element to ensure that colour identifiers are unique.

Finally the root element is defined:

```
<xs:element name="pixmap" type="Pixmap">
   <xs:keyref refer="colorKey" name="pixelRef">
      <xs:selector xpath="pixel"/>
      <xs:field xpath="."/>
   </xs:keyref>
   <xs:keyref refer="colorKey" name="backgroundRef">
      <xs:selector xpath="background"/>
      <xs:field xpath="."/>
   </xs:keyref>
```

```
    <xs:unique name="positionUnique">
        <xs:selector xpath="pixel"/>
        <xs:field xpath="@x"/>
        <xs:field xpath="@y"/>
    </xs:unique>
</xs:element>
```

The keyref element are there for ensure the referential integrity of the colour identifier used in the pixel and background element. The unique element ensures that no pixel is defined more than ones.

A complete pixmap file for the example above looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<pixmap xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="S100Pixmap.xsd">
    <description>Test pixmap showing a capital H in faint magenta.</description>
    <width>10</width>
    <height>9</height>
    <colorMap>
        <color id="_" transparency="1.0">#000000</color>
        <color id="M">#8F83B6</color>
    </colorMap>
    <background>_</background>
    <pixel x="3" y="2">M</pixel>
    <pixel x="3" y="3">M</pixel>
    <pixel x="3" y="4">M</pixel>
    <pixel x="3" y="5">M</pixel>
    <pixel x="3" y="6">M</pixel>
    <pixel x="4" y="4">M</pixel>
    <pixel x="5" y="4">M</pixel>
    <pixel x="6" y="2">M</pixel>
    <pixel x="6" y="3">M</pixel>
    <pixel x="6" y="4">M</pixel>
    <pixel x="6" y="5">M</pixel>
    <pixel x="6" y="6">M</pixel>
</pixmap>
```

Appendix 9-A ~~XML~~ Generating Portrayal Input XML~~Schemas~~ (informative)

Preface

This standard describes a base schema that contains base types which define the expected structure of the XML input to the portrayal processing.~~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.~~ The XSLT portrayal rules expect the input XML to conform with the base schema, with additional information as defined within a products feature catalogue.

Although the encoded elements of the input XML must conform to the base schema, it is not necessary to fully describe a schema for each products input XML. It is sufficient that the portrayal rules can consistently parse the input XML using XPath 1.0 queries.

The base schema should be used along with patterns described in this Annex to model a generic pattern for describing all S-100 content; essentially providing a mapping from the S-100 general feature model to an XML encoding of an instance of the model.

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

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Normal

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Highlight

Commented [DMG16]: TODO: Update 9-13.3 to support specifying the elements referenced by portrayal rules (e.g. what needs to be available in the input XML)

Formatted: Normal

~~9-A-1 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"/>~~
```

9-A-1 ~~Spatial Objects~~Input XML structure

The input XML must conform to the following structure, using the indicated element names and order.

Formatted: Normal, Space After: 0 pt, No bullets or numbering

```
<!—The root element -->
<Dataset>
        <InformationTypes>
        </InformationTypes>

        <!—Spatial Objects -->
        <Points>
        </Points>
        <MultiPoints>
        </MultiPoints>
```

Formatted: Indent: Left: 0.5"

```
<Curves>
</Curves>
<CompositeCurves>
</CompositeCurves>
<Surfaces>
</Surfaces>

<Features>
</Features>
</Dataset>
```

~~9-A-5~~

~~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>~~
```

9-A-2 Simple attributes

When encoding simple attributes the element name should match the code of a simple attribute defined within the product feature catalogue. For enumerations, the code should be passed as the value, the label should not be used.

```
    <colour>3</colour> <!—don’t encode as red -->
```

9-A-3 Complex attributes

When encoding complex attributes the element name should match the code of a complex attribute defined within the product feature catalogue. Simple and/or complex sub attributes should be nested within the element.

An instance of complex attribute „speed“ with simple sub attributes „speedMaximum“ and „speedMinimum“:

```
  <speed>
    <speedMaximum>10</speedMaximum>
    <speedMinimum>0</ speedMinimum>
  </speed>
```

9-A-4 Information types

Instances of information types are encoded within the *<InformationTypes>* section of the input XML. Information types are referenced by other objects encoded in the input XML.

The base schema defines abstract type *Information* for encoding of information types. The following pattern describes instantiation of the abstract type:

Formatted: Normal, Space After: 0 pt, Line spacing: Multiple 1.15 li, No bullets or numbering

Formatted: Font: 10 pt

Formatted: Highlight

Formatted: Left, Indent: Left: 0.5", Line spacing: Multiple 1.15 li

Formatted: Font: (Default) Times New Roman, Font color: Custom Color(RGB(0,0,150)), English (United States)

Formatted: Highlight

Formatted: Justified, Right: 0", Space Before: 0 pt, After: 0 pt, Line spacing: single

Formatted: Normal

Formatted: Normal, Space After: 0 pt, No bullets or numbering

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Space After: 6 pt

Formatted: Font: Italic

Formatted: No bullets or numbering

Formatted: Font: Italic

- The element name of each information type object must match the code of an InformationType defined in the product feature catalogue. The code defines the instantiation of the abstract type; in the example below, ChartNote is the instantiation of *Information*.
- The value(s) of the attributes of the information type should be included in the encoding, as described in 9-A-2 and 9-A-3. These values should be nested within the element described by the preceding bullet (they are part of the instantiation).

Notional information type ChartNote, described in the feature catalogue as having complex attribute *note*, which has simple sub attributes *noteText* and *language*:

```
    <ChartNote id="I1">
        <note>
            <noteText>Hello world!</noteText>
            <language>en</language>
        </note>
        <note>
            <noteText>Hallo Welt!</noteText>
            <language>de</language>
        </note>
    </ChartNote>
```

9-A-5 Spatial Objects

The base schema defines spatial types which describe XML encodings of the Part 7 Spatial Schema. These types provide a consistent way to represent spatial objects to the portrayal processing. The input XML encoding of spatial objects should conform with the base schema; the spatial model cannot be modified by a product feature catalogue.

In most cases the portrayal rules do not attempt to reference the geometry coordinates of spatial objects; in these cases the geometry coordinates may be omitted from the input XML, potentially providing significant advantages to the speed of generating and parsing the input XML. A notable exception when the geometry coordinates cannot be omitted: symbolization of MultiPoint spatial objects. Geometry coordinates should always be provided for MultiPoint spatial objects.

~~9-A-9~~9-A-6 ~~and f~~Features ~~types~~

Instances of feature types are encoded within the *<Features>* section of the input XML.

The base schema defines abstract type *Feature* for encoding of feature objects. The following pattern describes instantiation of the abstract type:

- The element name of each feature object must match the code of a feature type defined in the product feature catalogue. The code defines the instantiation of the abstract type; in the example below, BeaconLateral is the instantiation of *Feature*.
- The value(s) of the attributes of the feature object should normally be included in the encoding, as described in 9-A-2 and 9-A-3. These values should be nested within the element described by the preceding bullet (they are part of the instantiation). Attribute values provided via Coverages should be omitted from the encoding.
- All references to spatial objects should be included as described in the base schema.
- All associations to information type instances and feature instances should be included, as described in 9-A-7.

An example encoding of an instance of a BeaconLateral feature. The feature references a spatial object of type Point (P23), and has an association to a feature instance (F8) which should also be described in the input XML, but is not shown in this example. The value of each of the features attributes is also encoded.

~~[illegible]~~

Formatted: Font: Italic

Formatted: No bullets or numbering

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Normal, Space After: 0 pt, No bullets or numbering

Formatted: Font: Times New Roman, 10 pt, Font color: Custom Color(RGB(0,0,150)), English (United States)

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Bulleted + Level: 1 + Aligned at: 0" + Tab after: 0.2" + Indent at: 0.2"

Formatted: Font: Italic

Formatted: Bulleted + Level: 1 + Aligned at: 0" + Tab after: 0.2" + Indent at: 0.2"

~~<xs:complexType name="BeaconCardinal">~~

~~An instance looks like:~~

```
<BeaconLateral id="F4" primitive="Point" >
    <Point ref="P23" scaleMinimum="4294967295" scaleMaximum="0" />
    <beaconShape>1</beaconShape>
    <categoryOfLateralMark>2</categoryOfLateralMark>
    <colour>3</colour>
    <reportedDate>20050124</reportedDate>
    <status>1</status>
    <scaleMinimum>89999</scaleMinimum>
    <StructureEquipment role="supports" featureRef="F8" />
</BeaconLateral>
```

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

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

~~<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.~~

Formatted: Left, Space After: 0 pt, Line spacing: Multiple 1.15 li

~~<xs:complexType name="SpatialQuality">~~

~~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.~~

~~9-A-32~~9-A-7 Associations

Associations are named relationships between objects. ~~We have~~There are 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 base schema provides element *associatedInformationCode* as part of both feature objects and spatial objects, and provides element *associatedFeatureCode* as part of feature objects. The pattern for encoding associations is:

- The element name of each association must match the code of an association defined in the product feature catalogue. The code describes the subtype of the association; in the example below, StructureEquipment replaces *associatedFeatureCode* from the base schema and describes the relationship between the two feature objects.
- The value of the *role* attribute should match a role described in the feature catalogue.
- The value of *featureRef* (for feature associations) or *informationRef* (for information associations) should match the unique id of a feature instance or information type instance encoded within the input XML.
- Include the values of any simple or complex attributes which are defined for the association as described in the feature catalogue. These values should be nested within the association element.

An example feature association: *StructureEquipment*. The association should be nested within the encoding of a feature instance.

```
<Features>
  <FeatureTypeX id="F1" primitive=”Point” >
    <!-- feature attributes and spatial reference omitted -->
    <StructureEquipment role=”supports” featureRef=”F2”/>
  </FeatureTypeX>
  <FeatureTypeY id="F2" primitive=”Point” >
    <!-- feature attributes and spatial reference omitted -->
    <StructureEquipment role=”supportedBy” featureRef=”F1” />
  </FeatureTypeY>
</Features>
```

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Bulleted + Level: 1 + Aligned at: 0" + Tab after: 0.2" + Indent at: 0.2"

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Indent: Left: 0"

Formatted: Font color: Custom Color(RGB(56,86,35))

Formatted: Font color: Custom Color(RGB(56,86,35))

An example information association: An instance of a *SpatialQuality* information type referenced from a spatial object using a *SpatialAssociation* InformationAssociation type:

```
<InformationTypes>
    <SpatialQuality id="I1">
        <qualityOfHorizontalMeasurement>4</qualityOfHorizontalMeasurement>
    </SpatialQuality>
</InformationTypes>
<Points>
    <Point id="P1">
        <Coordinate2D>
            <x>0.0</x>
            <y>1.0</y>
        </Coordinate2D>
        <SpatialAssociation role="defines" informationRef="I1" />
    </Point>
</Points>
```

~~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>~~
```

~~9-A-33 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>~~
```

Formatted: Font: Italic

Formatted: Indent: Left: 0"

Formatted: Indent: Left: 0"

Formatted: Font: (Default) Times New Roman

Formatted: Left, Space After: 0 pt, Line spacing: Multiple 1.15 li

Formatted: Font: (Default) Times New Roman, Font color: Custom Color(RGB(0,0,150))

Formatted: Left, Space After: 0 pt, Line spacing: Multiple 1.15 li

~~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>~~
```

~~9-A-34 Sample S-101 Product Input Schema~~

```
~~<?xml version="1.0" encoding="UTF-8"?>~~
~~<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"~~
~~xmlns:s100="http://www.iho.int/S100BaseModel">~~
~~    <xs:import namespace="http://www.iho.int/S100BaseModel"~~
~~schemaLocation="S100BaseModel.xsd"/>~~

~~    <!-- INFORMATION ASSOCIATIONS -->~~
~~    <xs:complexType name="SpatialQualityAssociation">~~
~~        <xs:complexContent>~~
~~            <xs:extension base="s100:InformationAssociation"/>~~
~~        </xs:complexContent>~~
~~    </xs:complexType>~~
~~    <!-- FEATURE ASSOCIATIONS -->~~
~~    <xs:complexType name="UnderlyingAreaAssociation">~~
~~        <xs:complexContent>~~
~~            <xs:extension base="s100:FeatureAssociation"/>~~
~~        </xs:complexContent>~~
~~    </xs:complexType>~~

~~    <!-- INFORMATION TYPES -->~~
~~    <xs:complexType name="SpatialQuality">~~
~~        <xs:complexContent>~~
~~            <xs:extension base="s100:Information">~~
~~                <xs:sequence>~~
~~                    <xs:element name="qualityOfPosition"/>~~
~~                </xs:sequence>~~
~~            </xs:extension>~~
~~        </xs:complexContent>~~
~~    </xs:complexType>~~

~~    <xs:complexType name="Note">~~
~~        <xs:sequence minOccurs="1" maxOccurs="unbounded">~~
~~            <xs:element name="noteText" type="xs:string"/>~~
~~            <xs:element name="language" type="xs:language"/>~~
~~        </xs:sequence>~~
~~    </xs:complexType>~~

~~    <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>~~

  ~~<!-- GROUP OF ALL INFORMATION TYPES -->~~
  ~~<xs:group name="InformationType">~~
    ~~<xs:choice>~~
      ~~<xs:element name="SpatialQuality" type="SpatialQuality"/>~~
      ~~<xs:element name="ChartNote" type="ChartNote"/>~~
    ~~</xs:choice>~~
  ~~</xs:group>~~

  ~~<!-- SPATIAL TYPES (WITH SPATIAL QUALITY RELATIONS FOR POINTS,~~
~~POINTSETS, AND CURVES-->~~
  ~~<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>~~

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

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

  ~~<xs:complexType name="CompositeCurve">~~
    ~~<xs:complexContent>~~
      ~~<xs:extension base="s100:CompositeCurve"/>~~
    ~~</xs:complexContent>~~
  ~~</xs:complexType>~~

  ~~<xs:complexType name="Surface">~~
    ~~<xs:complexContent>~~
      ~~<xs:extension base="s100:Surface"/>~~
    ~~</xs:complexContent>~~
  ~~</xs:complexType>~~

  ~~<!-- FEATURE TYPES -->~~
  ~~<!-- BASE CLASS FOR ALL FEATURES WITH NOTE ASSOCIATION -->~~
  ~~<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>~~

~~<xs:complexType name="DepthArea">~~
~~<xs:complexContent>~~
~~<xs:extension base="Feature">~~
~~<xs:sequence>~~
~~<xs:element name="depthValue1" type="xs:double" nillable="true" minOccurs="0"~~
    ~~maxOccurs="1"/>~~
~~<xs:element name="depthValue2" type="xs:double" nillable="true" minOccurs="0"~~
    ~~maxOccurs="1"/>~~
~~</xs:sequence>~~
~~</xs:extension>~~
~~</xs:complexContent>~~
~~</xs:complexType>~~

~~<xs:complexType name="DepthContour">~~
~~<xs:complexContent>~~
~~<xs:extension base="Feature">~~
~~<xs:sequence>~~
~~<xs:element name="valueOfDepthContour" type="xs:double" nillable="true"~~
    ~~minOccurs="0" maxOccurs="1"/>~~
~~</xs:sequence>~~
~~</xs:extension>~~
~~</xs:complexContent>~~
~~</xs:complexType>~~

~~<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" type="xs:int" nillable="true" minOccurs="0"~~
    ~~maxOccurs="1"/>~~
~~</xs:sequence>~~
~~</xs:extension>~~
~~</xs:complexContent>~~
~~</xs:complexType>~~

~~<xs:complexType name="Landmark">~~
~~<xs:complexContent>~~
~~<xs:extension base="Feature">~~
~~<xs:sequence>~~
~~<xs:element name="categoryOfLandmark" type="xs:int" nillable="true" minOccurs="0"/>~~
~~<xs:element name="function" type="xs:string" minOccurs="0"/>~~
~~<xs:element name="visuallyConspicuous" type="xs:int" nillable="true" minOccurs="0"/>~~
~~<xs:element name="objectName" type="xs:string" nillable="true" minOccurs="0"/>~~
~~</xs:sequence>~~
~~</xs:extension>~~
~~</xs:complexContent>~~
~~</xs:complexType>~~

~~<!-- GROUP OF ALL FEATURES -->~~
~~<xs:group name="Feature">~~
~~<xs:choice>~~
~~<xs:element name="DepthArea" type="DepthArea"/>~~
~~<xs:element name="DepthContour" type="DepthContour"/>~~
```

```
~~<xs:element name="BeaconCardinal" type="BeaconCardinal"/>~~
~~<xs:element name="Landmark" type="Landmark"/>~~
~~</xs:choice>~~
~~</xs:group>~~

~~<!-- THE ELEMENTS OF THE DATA SET -->~~
~~<xs:complexType name="InformationTypes">~~
~~<xs:sequence minOccurs="0" maxOccurs="unbounded">~~
~~<xs:group ref="InformationType"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~
~~<xs:complexType name="Points">~~
~~<xs:sequence>~~
~~<xs:element name="Point" type="Point" minOccurs="0" maxOccurs="unbounded"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="MultiPoints">~~
~~<xs:sequence>~~
~~<xs:element name="MultiPoint" type="MultiPoint" minOccurs="0" maxOccurs="unbounded"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="Curves">~~
~~<xs:sequence>~~
~~<xs:element name="Curve" type="Curve" minOccurs="0" maxOccurs="unbounded"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="CompositeCurves">~~
~~<xs:sequence>~~
~~<xs:element name="CompositeCurve" type="CompositeCurve" minOccurs="0"~~
~~maxOccurs="unbounded"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="Surfaces">~~
~~<xs:sequence>~~
~~<xs:element name="Surface" type="Surface" minOccurs="0" maxOccurs="unbounded"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="Features">~~
~~<xs:sequence minOccurs="0" maxOccurs="unbounded">~~
~~<xs:group ref="Feature"/>~~
~~</xs:sequence>~~
~~</xs:complexType>~~

~~<xs:complexType name="Dataset">~~
~~<xs:sequence>~~
~~<!-- THE INFORMATION TYPES -->~~
~~<xs:element name="InformationTypes" type="InformationTypes" minOccurs="0">~~
~~<xs:key name="informationKey">~~
~~<xs:selector xpath="*"/>~~
~~<xs:field xpath="@id"/>~~
~~</xs:key>~~
~~</xs:element>~~
~~<!-- THE POINTS -->~~
~~<xs:element name="Points" type="Points" minOccurs="0">~~
~~<xs:key name="pointKey">~~
```

```
~~          <xs:selector xpath="Point"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~      <!-- THE MULTI POINTS  -->~~
~~      <xs:element name="MultiPoints" type="MultiPoints" minOccurs="0">~~
~~        <xs:key name="multiPointKey">~~
~~          <xs:selector xpath="MultiPoint"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~      <!-- THE CURVES -->~~
~~      <xs:element name="Curves" type="Curves" minOccurs="0">~~
~~        <xs:key name="curveKey">~~
~~          <xs:selector xpath="Curve"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~      <!-- THE COMPOSITE CURVES -->~~
~~      <xs:element name="CompositeCurves" type="CompositeCurves" minOccurs="0">~~
~~        <xs:key name="compositeCurveKey">~~
~~          <xs:selector xpath="CompositeCurve"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~      <!-- THE SURFACES -->~~
~~      <xs:element name="Surfaces" type="Surfaces" minOccurs="0">~~
~~        <xs:key name="surfaceKey">~~
~~          <xs:selector xpath="Surface"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~      <!-- THE FEATURE TYPES -->~~
~~      <xs:element name="Features" type="Features">~~
~~        <xs:key name="featureKey">~~
~~          <xs:selector xpath="*"/>~~
~~          <xs:field xpath="@id"/>~~
~~        </xs:key>~~
~~      </xs:element>~~
~~    </xs:sequence>~~
~~  </xs:complexType>~~

~~  <!-- THE ROOT ELEMENT (OF TYPE DataSet) -->~~
~~  <xs:element name="Dataset" type="Dataset">~~
~~    <!-- KEY REFERENCES -->~~
~~    <xs:keyref name="informationRef" refer="informationKey">~~
~~        <xs:selector xpath=".//*"/>~~
~~        <xs:field xpath="@informationRef"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="pointRef" refer="pointKey">~~
~~        <xs:selector xpath="Features/*/s100:Point | Curves/Curve/s100:Boundary"/>~~
~~        <xs:field xpath="@ref"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="multiPointRef" refer="multiPointKey">~~
~~        <xs:selector xpath="Features/*/s100:MultiPoint"/>~~
~~        <xs:field xpath="@ref"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="curveRef" refer="curveKey">~~
~~        <xs:selector xpath="Features/*/s100:Curve | CompositeCurves/CompositeCurve/s100:Curve |~~
~~            Surfaces/Surface/s100:Ring/s100:Curve"/>~~
~~        <xs:field xpath="@ref"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="compositeCurveRef" refer="compositeCurveKey">~~
~~        <xs:selector xpath="Features/*/s100:CompositeCurve |~~
~~            CompositeCurves/CompositeCurve/s100:CompositeCurve |~~
~~            Surfaces/Surface/s100:Ring/s100:CompositeCurve"/>~~
~~        <xs:field xpath="@ref"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="surfaceRef" refer="surfaceKey">~~
~~        <xs:selector xpath="Features/*/s100:Surface"/>~~
~~        <xs:field xpath="@ref"/>~~
~~    </xs:keyref>~~
~~    <xs:keyref name="featureRef" refer="featureKey">~~
~~        <xs:selector xpath="Features/*/*"/>~~
~~        <xs:field xpath="@featureRef"/>~~
~~    </xs:keyref>~~
~~  </xs:element>~~

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

~~9-A-35~~9-A-8 Example Product Input Dataset

Commented [DMG17]: Consider removing this and referencing a sample input XML provided via the portrayal registry. Allows for updating outside of the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dataset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
~~xsi:noNamespaceSchemaLocation="S101DataModel.xsd"~~
xmlns:s100="http://www.iho.int/S100BaseModel">

  <!-- THE INFORMATION TYPES -->
  <InformationTypes~~ xmlns=""~~>

    <!-- A CHART NOTE -->
    <ChartNote id="1">
      <note>
        <noteText>Hello world!</noteText>
        <language>en</language>
      </note>
      <note>
        <noteText>Hallo Welt!</noteText>
        <language>de</language>
      </note>
    </ChartNote>

    <!-- AN INFORMATION OBJECT INDICATING SPATIAL QUALITY -->
    <SpatialQuality id="2">
```

```
            <qualityOfPosition>2</qualityOfPosition>
        </SpatialQuality>

    </InformationTypes>

    <!-- THE SPATIAL OBJECTS -->
    <!-- SOME POINTS -->
    <Points>
        <Point id="1">
            <Coordinate2D>
                <x>1.0</x>
                <y>2.0</y>
            </Coordinate2D>
        </Point>

        <Point id="2">
            <Coordinate2D>
                <x>1.0</x>
                <y>2.0</y>
            </Coordinate2D>
        </Point>

        <Point id="3">
            <Coordinate2D>
                <x>1.0</x>
                <y>2.0</y>
            </Coordinate2D>
        </Point>
    </Points>

    <!-- POINT SETS -->
    <MultiPoints>
        <MultiPoint id="1">
            <Coordinate3D>
                <x>2.0</x>
                <y>3.0</y>
                <z>5.3</z>
            </Coordinate3D>
            <Coordinate3D>
                <x>5</x>
                <y>6</y>
                <z>7</z>
            </Coordinate3D>
        </MultiPoint>

        <MultiPoint id="2">
            <Coordinate3D>
                <x>2.0</x>
                <y>2.5</y>
                <z>5.3</z>
            </Coordinate3D>
            <Coordinate3D>
                <x>2.0</x>
                <y>2.5</y>
                <z>5.3</z>
            </Coordinate3D>
        </MultiPoint>

        <MultiPoint id="5">
            <Coordinate3D>
                <x>2.0</x>
                <y>2.5</y>
                <z>5.3</z>
            </Coordinate3D>
        </MultiPoint>
    </MultiPoints>

    <!-- CURVES -->
    <Curves>
        <Curve id="1">
            <Boundary ref="1" boundaryType="Begin"/>
            <Boundary ref="1" boundaryType="End"/>
            <Segment interpolation="Loxodromic">
                <ControlPoint>
                    <x>1</x>
                    <y>2</y>
                </ControlPoint>
                <ControlPoint>
                    <x>1</x>
                    <y>2</y>
                </ControlPoint>
            </Segment>
        </Curve>

        <Curve id="2">
            <Boundary ref="2" boundaryType="Begin"/>
            <Boundary ref="3" boundaryType="End"/>
            <Segment interpolation="Loxodromic">
                <ControlPoint>
                    <x>3.6</x>
                    <y>4.5</y>
                </ControlPoint>
                <ControlPoint>
                    <x>3.8</x>
                    <y>4.7</y>
                </ControlPoint>
            </Segment>
        </Curve>

        <Curve id="3">
            <Segment interpolation="Loxodromic">
                <ControlPoint>
                    <x>3.6</x>
                    <y>4.5</y>
                </ControlPoint>
                <ControlPoint>
                    <x>3.8</x>
                    <y>4.7</y>
                </ControlPoint>
            </Segment>
            <S~~s~~patial~~Quality~~Association role="~~q~~defines~~ualityOfPosition~~" informationRef="2"/>
        </Curve>

        <Curve id="5">
            <Segment interpolation="Loxodromic">
                <ControlPoint>
                    <x>3.6</x>
                    <y>4.5</y>
                </ControlPoint>
                <ControlPoint>
```

```
                <x>3.8</x>
                <y>4.7</y>
            </ControlPoint>
            <ControlPoint>
                <x>7</x>
                <y>5.3</y>
            </ControlPoint>
        </Segment>
    </Curve>
    <Curve id="6">
        <CircleByCenterPoint interpolation="CircularArcCenterPointWithRadius" radius="5.0">
            <ControlPoint>
                <x>2.0</x>
                <y>3.0</y>
            </ControlPoint>
        </CircleByCenterPoint>
        <ArcByCenterPoint interpolation="CircularArcCenterPointWithRadius" radius="5.0"
                startAngle="23.0" angularDistance="-45.0">
            <ControlPoint>
                <x>12.4</x>
                <y>22</y>
            </ControlPoint>
        </ArcByCenterPoint>
    </Curve>

</Curves>

<!-- COMPOSITE CURVES -->
<CompositeCurves>
    <CompositeCurve id="1">
        <Curve ref="1" orientation="Forward"/>
        <Curve ref="5" orientation="Reverse"/>
    </CompositeCurve>
    <CompositeCurve id="2">
        <Curve ref="3" orientation="Forward"/>
        <CompositeCurve ref="1" orientation="Reverse"/>
    </CompositeCurve>
</CompositeCurves>

<!-- SURFACES -->
<Surfaces>
    <Surface id="1">
        <Ring type="Outer">
            <Curve ref="1" orientation="Forward"/>
            <CompositeCurve ref="1" orientation="Reverse"/>
        </Ring>
        <Ring type="Inner">
            <Curve ref="5" orientation="Reverse"/>
        </Ring>
    </Surface>
</Surfaces>

<!-- THE FEATURE TYPES OF THE DATA SET -->
<Features>
    <DepthArea id="1" primitive="Surface">
        <depthValue1 xsi:nil="true"/>
        <depthValue2>0</depthValue2>
    </DepthArea>

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

    <DepthArea id="3" primitive="Surface">
        <Surface ref="1"/>
        <depthValue1>0</depthValue1>
        <depthValue2>5</depthValue2>
    </DepthArea>

    <DepthContour id="4" primitive="Curve">
        <Curve ref="2" orientation="Forward"/>
        <CompositeCurve ref="2" orientation="Reverse"/>
    </DepthContour>

    <DepthContour id="5" primitive="Curve">
        <Curve ref="2" orientation="Forward"/>
        <Curve ref="3" orientation="Reverse"/>
    </DepthContour>

    <Landmark id="6" primitive="Point">
        <Point ref="2"/>
        <categoryOfLandmark>15</categoryOfLandmark>
        <function>21</function>
        <visuallyConspicuous>1</visuallyConspicuous>
    </Landmark>

    <DepthArea id="7" primitive="Surface">
        <depthValue1>5</depthValue1>
        <depthValue2>10</depthValue2>
    </DepthArea>

    <DepthArea id="8" primitive="Surface">
        <depthValue1>10</depthValue1>
        <depthValue2>20</depthValue2>
    </DepthArea>

    <DepthArea id="9" primitive="Surface">
        <depthValue1>20</depthValue1>
    </DepthArea>

    <Landmark id="10" primitive="Point">
        <categoryOfLandmark>5</categoryOfLandmark>
    </Landmark>

    <Landmark id="11" primitive="Point"/>

    <Landmark id="12" primitive="Point">
        <categoryOfLandmark>17</categoryOfLandmark>
        <function>33</function>
        <objectName>A name</objectName>
    </Landmark>

    <Landmark id="13" primitive="Point">
        <visuallyConspicuous>1</visuallyConspicuous>
        <objectName>No display name</objectName>
    </Landmark>
```

```
    </Features>
</Dataset>
```

Appendix 9-B
SVG Profile
(normative)

9-B-1 Introduction

This appendix describes the subset of SVG elements that have been used in the creation of S-100 SVG symbols and covers the set of SVG elements and associated attributes and properties that are in use by S-100.

The S-100 SVG profile is a subset of the SVG Tiny 1.2 profile
http://www.w3.org/TR/SVGTiny12/

9-B-2 Top Level SVG

The main svg element carries this indication as well as properties of each individual svg symbol as xml attributes.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
xml:space="preserve" style="shape-rendering:geometricPrecision; fill-rule:evenodd;"
width="4.34mm" height="5.35mm" viewBox="-2.22 -2.79 4.34 5.35">
```

9-B-2-1 Coordinate System

The overall width and height of the symbol are defined in mm. The viewbox covers the range of coordinates used for the symbol. The pivot point of the symbol is designed to be at the 0,0 position.

The default coordinate system used for S-100 SVG has the origin in the upper left corner with the x-axis pointing to the right and the y-axis pointing down.

9-B-2-2 Title

The title element is used to carry the name of the symbol.

```
<title>ACHARE02</title>
```

9-B-2-3 Description

The description element is used to carry a brief textual description of the symbol.

```
<desc>anchorage area as a point at small scale, or anchor points of mooring trot at large
scale</desc>
```

9-B-2-4 Metadata

SVG has a metadata element which allows for the direct inclusion of metadata document fragments from other namespaces. The following example shows how IHO could define the appropriate metadata content for a symbol. The IHO S-100 working group is encouraged to define a metadata schema for use with S-100 symbols.

```
  <metadata>
    <iho:S100SVG xmlns:iho="http://www.iho.int/SVGMetadata">
      <iho:Description iho:publisher="IHO" iho:creationDate="2014-06-09"
iho:source="S52Preslib4.0" iho:format="S100SVG" iho:version="0.1"/>
    </iho:S100SVG>
  </metadata>
```

9-B-3 Drawing Elements

The body of the SVG symbol contains the drawing elements. The drawing elements implemented so far include path, rect and circle with details to follow. These drawing elements share some common attributes such as 'class'.

9-B-3-1 Class

The 'class' attribute is used to assign one or more class names to the element. In the S-100 SVG the class attribute is used to assign style information by way of a CSS stylesheet. It can also be used to filter or control which elements should be shown. Essentially the class tokens can be used as a key to find a set of style instructions in the corresponding Cascading Style Sheet (CSS). A processing instruction at the head of the SVG symbol indicates the corresponding CSS file.

9-B-3-1.1 CSS

```
<?xml-stylesheet href="SVGStyle.css" type="text/css"?>
```

An example excerpt of such a CSS file might be as follows:

```
.layout {display:none}  /* used to control visibility of symbolBox, svgBox, pivotPoint (none or inline) */
.symbolBox {stroke:black;stroke-width:0.32;}  /* show the cover of the symbol graphics */
.svgBox {stroke:blue;stroke-width:0.32;}  /* show the entire SVG cover */
.pivotPoint {stroke:red;stroke-width:0.64;}  /* show the pivot/anchor point, 0,0 */
.sl {stroke-linecap:round;stroke-linejoin:round} /* default line style elements */
.f0 {fill:none}  /* no fill */
.sCURSR {stroke:#E38039}  /* sRGB line colour for colour token CURSR */
.fCURSR{fill:#E38039} /* sRGB fill colour for colour token CURSR*/
.sCHBLK {stroke:#000000}
.fCHBLK {fill:#000000}
.sCHGRD {stroke:#4C5B63}
.fCHGRD {fill:#4C5B63}
.sCHGRF {stroke:#768C97}
.fCHGRF {fill:#768C97}
.sCHRED {stroke:#EA5471}
.fCHRED {fill:#EA5471}
.sCHGRN {stroke:#52E93A}
.fCHGRN {fill:#52E93A}
.sCHMGD {stroke:#C045D1}
.fCHMGD {fill:#C045D1}
…
```

This mechanism allows for possibility to change the colours used in the symbols by swapping the CSS file with different contents according to the desired colour scheme. Each colour token is encoded for both a stroke style and a fill style. The stroke is used for drawing lines and the fill for filling closed shapes. In the above example the token 'sCHMGD' translates into a 'stroke' property using the sRGB colour #C045D1 and fCHMGD represents a fill operation. Different CSS files would be used for each colour palette with the sRGB values calculated using a formula to convert from the official CIE values.

NOTE: In converting CIE to sRGB, the rendering intent must follow an absolute colorimetry method. Due to the differences in color and luminance performance between individual monitors, any "formula" for conversion from CIE to sRGB must be based on measurements to characterize (calibrate) the monitor in order to meet the color accuracy and separation specified for ECDIS. For interoperability with ECDIS, portrayal of other S-1xx products would need to follow the same rendering intent.

9-B-3-2 Style Properties

The style properties used in the draft S-100 SVG symbols include:

- 'stroke' - the pen colour for lines defined with a hexadecimal sRGB value;

- ‘stroke-width’ - the pen width in the same units as the SVG width/height. For S-100 SVG we are using mm;
- ‘stroke-opacity’ - range of 0.0 (fully transparent) to 1.0 (fully opaque);
- ‘fill’ - the colour to fill closed shapes defined with a hexadecimal sRGB value;
- ‘fill-opacity’- range of 0.0 (fully transparent) to 1.0 (fully opaque);
- ‘stroke-linecap’ – style for the ends of lines, choice of (butt | round | square);
- ‘stroke-linejoin’ – style for corners within a line path, choice of (miter | round | bevel);
- ‘display’ – identifies whether the element is to be included/rendered or not. Default is ‘inline’. This is used as a way to hide or show the layout elements of the symbol such as the covering box or the pivot point. This way a different CSS file with layout display set to ‘inline’ can be used when viewing the symbol in a design/engineering view.

9-B-3-3 Path

```
<path d=" M -2.06,1.36 L -1,2.4 L 0.98,2.4 L 1.96,1.39" class="sl f0 sCHMGD" style="stroke-width: 0.32;"/>
```

```
<path d=" M -5.88,-5.88 L 5.87,-5.88 L 5.87,5.87 L -5.88,5.87 L -5.88,-5.88 Z" class="fDNGHL" style="fill-opacity:0.25;"/>
```

The ‘d’ attribute carries the path data which describes the outline of a shape. In the current set of SVG symbols the path data is made up of *moveto* ‘M’ and *lineto* ‘L’ instructions as well as the *closepath* ‘Z’ instruction. The *curve* instructions have not yet been used. ‘M’ and ‘L’ instructions are following by a pair of absolute coordinates. Relative coordinates indicated with lowercase ‘m’ and ‘l’ instructions are not used. Note that some style elements can be assigned specifically using the ‘style’ attribute and others are coming from the stylesheet via the class lookups as described above.

9-B-3-4 Rectangle

```
<rect class="symbolBox layout" fill="none" x="-2.06" y="-2.63" height="5.03" width="4.02"/>
```

The ‘rect’ command uses ‘x’ and ‘y’ to define the upper left corner of the rectangle and the attributes ‘width’ and ‘height’ in the user units, mm. Specific style parameters are defined using the ‘style’ attribute while colours and other common styles are applied via the class token CSS lookups.

9-B-3-5 Circle

```
<circle class="pivotPoint layout" fill="none" cx="0" cy="0" r="1"/>
```

The ‘circle’ command uses ‘cx’ and ‘cy’ to define the centre of the circle and the attribute ‘r’ to define the radius in the user units, mm. Specific style parameters are defined using style attributes while colours and other common styles are applied via the class token CSS lookups.

Page intentionally left blank