CR-Form-v3

CHANGE REQUEST

⌘ **SE 1.1.0 CR ?** ⌘ rev **-** ⌘ Current version: 1.1.0 ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ AS ☐ Imp Spec ☒ Best Practices Paper ☐ Other ☐

Title:	⌘	Enhancements to Symbology Encoding in Support of IHO S-52 and UKHO AML
Source:	⌘	Alessandro Triglia, OSS Nokalva
Work item code:	⌘	***Date:*** ⌘ 2009-04-14
Category:	⌘	**C**

Use one of the following categories:
- ***F*** *(Critical correction)*
- ***A*** *(corresponds to a correction in an earlier release)*
- ***B*** *(Addition of feature),*
- ***C*** *(Functional modification of feature)*
- ***D*** *(Editorial modification)*

Detailed explanations of the above categories can be found in the TC Policies and Procedures.

Reason for change:	⌘	The current SE standard does not adequately support the symbols used in electronic navigational charts as specified by the International Hydrographic Organization and the UK Hydrographic Office.
Summary of change:	⌘	Improvements in the following areas: (1) complex linestyles based on single repeated graphics and combinations of multiple graphics and simple linestyles; (2) specification of pivot points for placement and rotation; and (3) portrayal of feature types having multiple delineations. Minor corrections.
Consequences if not approved:	⌘	Lack of support or insufficient support for navigational charts.

Clauses affected:	⌘	11
Other specs Affected:	⌘	☐ Other core specifications ⌘ ☐ Abstract specifications ☐ Best Practices Document
Supporting Doc.	⌘	
Other comments:	⌘	
Status	⌘	
Disposition	⌘	

1 Enhancements to Symbology Encoding in Support of IHO S-52 and UKHO AML

1.1 Issues related to the representation of S-52 symbols in SE

TENET Technology, Inc. encountered some difficulties in trying to express a certain set of electronic navigational charts symbols in the SE language. The symbology in question is specified in a document of the UK Hydrographic Office ([AML]), which relies upon a set of symbols specified in a Special Publication of the

International Hydrographic Organization ([S-52]). A report ([TENET]) recently published by TENET Technology discusses some of the issues that they identified:

— how to describe AML and S-52 complex linestyles in SE;

— how to represent S-52 symbol pivot points in SE; and

— how to deal with AML symbols that have multiple delineations.

1.1.1 Complex linestyles

According to [TENET], S-52 includes two kinds of complex linestyles: *single-unit complex linestyles* and *composite complex linestyles*, both of which are not adequately supported by SE:

— *single-unit complex linestyles* are formed by concatenating multiple copies of a single graphic symbol (all oriented in the same way) between each pair of vertices of the line, supplemented by a simple linestyle to fill any gap between graphic symbols with different orientations at each vertex;

— *composite complex linestyles* are formed by concatenating multiple copies of a sequence of graphic symbols and horizontal lines, whose orientation can easily be changed at each vertex of the line.

The current definition of **Stroke** in [SE] is:

```
<xsd:element name="Stroke" type="se:StrokeType"/>


<xsd:complexType name="StrokeType">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element ref="se:GraphicFill"/>
      <xsd:element ref="se:GraphicStroke"/>
    </xsd:choice>

    <xsd:element ref="se:SvgParameter" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Cubewerx proposes ([CW ER SE]) to modify the definition of the **Stroke** element to an abstract element **AbstractStroke** with four substitutions: **Stroke**, **GraphicStroke**, **TextStroke**, and **CompoundStroke**. We support that proposal with some minor changes. One of the changes is the use of the element name **SingleGraphic** instead of **GraphicStroke**, to avoid confusion with the existing element name **GraphicStroke**. Another is the use of the element name **SimpleStroke** instead of **Stroke** to better qualify this kind of stroke and avoid confusion with the old element name **Stroke**. Another is the use of the elements **InitialStrokeElement** and **FinalStrokeElement** (instead of **StartGraphic** and **EndGraphic**) within the **CompoundStroke** element to achieve more generality.

An AML or S-52 complex linestyle could be supported by the following definitions, which also partly implement the Cubewerx proposal.

```
<xsd:element name="AbstractStroke" type="se:AbstractStrokeType" abstract="true"/>


<xsd:complexType name="AbstractStrokeType">
  <xsd:sequence>
    <xsd:element ref="se:UnitOfMeasure" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>


<xsd:element name="UnitOfMeasure" type="xsd:anyURI"/>


<xsd:element name="SimpleStroke" type="se:SimpleStrokeType"
```

```
    substitutionGroup="se:AbstractStroke"/>


<xsd:complexType name="SimpleStrokeType">
  <xsd:complexContent>
    <xsd:extension base="se:AbstractStrokeType">
      <xsd:sequence>
        <xsd:choice minOccurs="0">
          <xsd:element name="SolidColor" type="se:ParameterValueType"/>
          <xsd:element name="Stipple" type="se:StippleType/>
        </xsd:choice>

        <xsd:element name="Opacity" type="se:ParameterValueType" minOccurs="0"/>
        <xsd:element name="Width" type="se:ParameterValueType" minOccurs="0"/>
        <xsd:element name="LineJoin" type="se:ParameterValueType" minOccurs="0"/>
        <xsd:element name="LineCap" type="se:ParameterValueType" minOccurs="0"/>
        <xsd:element name="DashArray" type="se:ParameterValueType" minOccurs="0"/>
        <xsd:element name="DashOffset" type="se:ParameterValueType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>



<xsd:complexType name="StippleType">
  <xsd:sequence>
    <xsd:element ref="se:Graphic"/>
  </xsd:sequence>
</xsd:complexType>



<xsd:element name="SingleGraphic" type="se:SingleGraphicType"
    substitutionGroup="se:AbstractStroke"/>


<xsd:complexType name="SingleGraphicType">
  <xsd:complexContent>
    <xsd:extension base="se:AbstractStrokeType">
      <xsd:sequence>
        <xsd:element ref="se:Graphic"/>
        <xsd:element ref="se:Error! Reference source not found." minOccurs="0"/>
          </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>


<xsd:element name="RelativeOrientation" type="se:RelativeOrientationType">


<xsd:simpleType name="RelativeOrientationType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="normal"/>
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="map"/>
    <xsd:enumeration value="normal180"/>
  </xsd:restriction>
</xsd:simpleType>


<xsd:element name="TextStroke" type="se:TextStrokeType"
    substitutionGroup="se:AbstractStroke"/>


<xsd:complexType name="TextStrokeType">
  <xsd:complexContent>
    <xsd:extension base="se:AbstractStrokeType">
      <xsd:sequence>
        <xsd:element ref="se:LineLabel"/>
```

```
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>


<xsd:element name="CompoundStroke" type="se:CompoundStrokeType"
    substitutionGroup="se:AbstractStroke"/>


<xsd:complexType name="CompoundStrokeType">
  <xsd:complexContent>
    <xsd:extension base="se:AbstractStrokeType">
      <xsd:sequence>
        <xsd:element name="InitialStrokeElement" type="se:StrokeElementType"
            minOccurs="0" maxOccurs="1"/>
        <xsd:element name="StrokeElement" type="se:StrokeElementType"
            minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="FinalStrokeElement" type="se:StrokeElementType"
            minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="StrokeElementType">
  <xsd:sequence>
    <xsd:element ref="se:AbstractStroke"/>
    <xsd:element ref="se:PerpendicularOffset" minOccurs="0"/>
    <xsd:element ref="se:GapBefore" minOccurs="0"/>
    <xsd:element ref="se:Length" minOccurs="0"/>
    <xsd:element ref="se:GapAfter" minOccurs="0"/>
    <xsd:element name="DrawOnlyWhereLineIsNotStraight" type="xsd:boolean"
        minOccurs="0"/>
    <xsd:element name="DrawOnlyWhereLineIsStraight" type="xsd:boolean"
        minOccurs="0"/>
    <xsd:element name="ReturnAfterDrawing" type="xsd:boolean"
        minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>


<xsd:element name="Graphic" type="se:GraphicType"/>
```

The boolean elements **DrawOnlyWhereLineIsStraight**, **DrawOnlyWhereLineIsNotStraight**, and **ReturnAfterDrawing** are intended to support S-52 complex linestyles.

A **DrawOnlyWhereLineIsStraight** element with a value of **true** indicates that a copy of the stroke element is to be drawn only when it corresponds to a single line segment that is straight. The default value is **false**.

A **DrawOnlyWhereLineIsNotStraight** element with a value of **true** indicates that a copy of the stroke element is to be drawn only when it corresponds to a single line element that is not straight or to two or more consecutive line segments(e.g., near a vertex). The default value is **false**.

A **ReturnAfterDrawing** element with a value of **true** indicates that the drawing position is to be reset to the initial drawing position for this element once the stroke element (which may be any type of stroke) has been completely drawn. The default value is **false**.

An alternative to introducing the **ReturnAfterDrawing** element could be to add a new level in the structure of the **CompoundStroke** element: Each stroke element would be a group of strokes that are all to be drawn starting from the same position, and the drawing position would be advanced once all the strokes in the group have been drawn. Of course, in many cases the "group" would contain exactly one stroke.

The above elements would support S-52 complex linestyles in the following ways:

— An S-52 single-unit complex linestyle can be created by using a compound stroke containing two stroke elements: the first stroke element will be a graphic stroke with **`DrawOnlyWhereLineIsStraight`** = **`true`**, and the second will be a simple stroke (e.g., a dashed line) with **`DrawOnlyWhereLineIsNotStraight`** = **`true`**.

NOTE As mentioned above, an AML or S-52 single-unit complex linestyle uses a single repeated graphic for straight line segments and a simple linestyle for the parts of the line near the vertices.

— An S-52 composite complex linestyle can be created by using a compound stroke containing multiple stroke elements, one or more of which will have **`ReturnAfterDrawing`** = **`true`**.

NOTE As mentioned above, an AML or S-52 composite complex linestyle uses a combination of horizontal lines and multiple repeated graphics which can be rotated.

We also suggest moving the **`PerpendicularOffset`** element from **`LineSymbolizer`** and **`PolygonSymbolizer`** into **`StrokeElement`**. This will enable the individual stroke elements of a compound stroke to each have its own perpendicular offset.

Cubewerx ([CW ER SE]) suggests that compound strokes containing other compound strokes should be forbidden. The possibility of specifying strokes composed of other compound strokes, however, seems useful as it would enable more complex patterns of subsymbols for a composite linestyle (especially in combination with **`ReturnAfterDrawing`** and **`PerpendicularOffset`**, both of which could be specified for each stroke element at any depth). It needs to be determined whether the potential advantages justify the added complexity due to recursion in practical applications of symbology encoding.

1.1.2 Pivot points

Every S-52 symbol, including those forming complex linestyles, has a defined "pivot point". The pivot point of a symbol:

— can be anywhere inside or outside the symbol's bounding box;

— is used for positioning the symbol; and

— is used as the rotation point for the symbol.

[TENET] states that SE does not dequately support S-52 pivot points. The issues are the following.

In SE, a graphic has an optional "anchor point". However, the semantics of the anchor point is ambiguous. The anchor point is defined as follows ([SE], 11.3.2, p. 25):

> The AnchorPoint element of a PointSymbolizer gives the location inside of a Graphic to use for anchoring the graphic to the main-geometry point. The coordinates are given as two floating-point numbers ... between 0.0 and 1.0 inclusive. The bounding box of the graphic to be rendered is considered to be in a coordinate space from 0.0 (lower-left corner) to 1.0 (upper-right corner), and the anchor position is specified as a point in this space.

Yet SE has also a concept of "hot spot" for a graphic, which occurs in the following statement:

> The “hot spot” to use for positioning the rendering at a point must either be inherent in the external format or is defined to be the “central point” of the graphic, where the exact definition “central point” is system-dependent.

It is unclear whether the "anchor point" and the "hot spot" are the same thing or not. If they are, then there is an apparent contradiction between the statements quoted above. If they are not, then the relationship between the two points is unclear. In both cases the text needs to be corrected:

— if the hot spot of a graphic is intended to be the same as the anchor point, then the text needs to be corrected to eliminate the term "hot spot"; and the statement beginning with *"The hot spot to use for positioning the rendering...*" needs to be corrected to remove the contradiction;

— otherwise, the term "hot spot" needs to be defined in the standard, and its difference from the anchor point needs to be made clear.

Another problem with the SE anchor point (at least with respect to the S-52 pivot point) is that it is not used to rotate the graphic. For an explicitly specified rotation (using a **Rotation** element), the rotation point is specified in the following statement:

> The Rotation element gives the rotation of a graphic in the clockwise direction about its center point ... Also, the point within the graphic about which it is rotated is format dependent. If a format does not include an inherent rotation point, then the point of rotation should be the centroid.

Clearly, in SE the anchor point is not used as the rotation point, even when an anchor point is explicitly provided for the graphic (in an **AnchorPoint** element). Undoubtedly this also applies to the graphics that are used as stroke elements and are rotated implicitly (i.e., without a **Rotation** element) to follow the varying direction of the line.

Another problem with the SE anchor point (at least with respect to the S-52 pivot point) is that it is required to be placed within the symbol's bounding box.

In order to address these problems and provide better support for AML and S-52 symbols in SE, we suggest that a new element **PivotPoint** be added to SE as shown in the following schema fragment.

```
<xsd:complexType name="ExternalGraphicType">
  <xsd:sequence>
    ...
    <xsd:element ref="se:Opacity" minOccurs="0"/>
    <xsd:element ref="se:Size" minOccurs="0"/>
    <xsd:element ref="se:Rotation" minOccurs="0"/>
    <xsd:element ref="se:Error! Reference source not found." minOccurs="0"/>
    <xsd:element ref="se:AnchorPoint" minOccurs="0"/>
    ...
  </xsd:sequence>
</xsd:complexType>


<xsd:element name="PivotPoint" type="se:PivotPointType">


<xsd:complexType name="PivotPointType">
  <xsd:sequence>
    <xsd:element ref="se:Error! Reference source not found."/>
    <xsd:element ref="se:Error! Reference source not found."/>
  </xsd:sequence>
</xsd:complexType>


<xsd:element name="PivotPointX" type="se:ParameterValueType"/>


<xsd:element name="PivotPointY" type="se:ParameterValueType"/>
```

The pivot point would be optional, and would have the same semantics as an S-52 pivot point. Namely:

a) the pivot point can be anywhere inside or outside the symbol's bounding box;

b) pivot point X or Y coordinates below 0.0 or above 1.0 indicate positions outside the bounding box;

c) the pivot point is used for positioning the symbol; and

d) the pivot point is used as the rotation point for the symbol.

The introduction of the **PivotPoint** element interferes with the semantics of the existing **AnchorPoint** element, and therefore the following compatibility rules are suggested:

a) a graphic must not have both an **PivotPoint** element and an **AnchorPoint** element;

b) if a **PivotPoint** element is not present in a graphic, then the current rules for positioning the graphic and for rotating the graphic must be followed.

In a future version of SE, the **AnchorPoint** element could be made obsolete, and only the **PivotPoint** (which has a cleaner semantics) would then be retained.

An alternative solution could be to modify the semantics of **AnchorPoint** and **Rotation** and the rules for positioning and rotation of graphics. For example, one could remove the current restriction on the coordinates of the anchor point to allow it to be placed outside the bounding box of the graphic. However, we feel that the introduction of a new element (**PivotPoint)** is a better solution because it associates the new behavior with a conspicuous change in syntax, while the old syntax still carries the old semantics unchanged. This helps preventing conformance and compatibility headaches.

Cubewerx ([CW ER SE]) proposes to replace the elements **Size**, **Rotation**, **AnchorPoint**, and **Displacement** with a new set of elements **ViewBox**, **Center**, **UnitOfMeasure**, and **Transform**. Assuming that that proposal is accepted, we think that a distinct **PivotPoint** element would provide a way to specify a major property of a graphic that is both useful and intuitive.

1.1.3 Delineations

The discussion about delineations in [TENET], 3.3 is a little confusing. While it is clear from that discussion that certain features need the ability to behave (with respect to portrayal) either as a point or as a line or as an area depending on the situation, it is not as clear whether this ability is required for the *feature types* or for the individual *instances* of those feature types.

The main example given in the report is the feature type "Marine Farm/Culture", which has all the three possible "delineations": area, line, and point. Does this mean that each marine farm instance will be able to behave as a point as well as a line as well as an area? Or does it mean that each marine farm instance will be just one of those three things even though the feature type allows all of them? The five "fixes/workarounds" presented on p. 12-13 of [TENET] suggest the latter. The fifth, for example, implies that the feature instance will have only one non-null relevant geometry-valued property, which will be either 0-dimensional or 1-dimensional or 2-dimensional. In all of the five workarounds, the "delineation" is treated as an implicit property of a feature instance, rather than, say, as an externally provided value (like the portrayal scale), and all of the five workarounds seem to be concerned with finding how to best express the delineation condition in SE syntax, and do not show any concern with, say, how to select one geometry-value property of a feature instance that has many such properties, as would be the case if a feature *instance* were allowed to have multiple delineations. In conclusion, our interpretation of the requirements is that there is a need to support feature types that have multiple delineations, but there is no need for each individual instance of those feature types to have more than one delineation.

Restricting feature instances to using one delineation at a time does not completely solve the problem of helping the renderer determine which delineation is being used by a given feature instance. When each delineation of a feature type is carried by a different geometry-valued property of the feature, the name of the only geometry-valued property that is present will allow the renderer to determine the delineation, but when all the possible delineations are carried by the same geometry-valued property, the renderer will have to examine the nature of the geometry (the value of that property) in order to determine the delineation. Our interpretation of [TENET] is that the latter case occurs in the [AML]. We believe that an appropriate way to support such usage would be to add to the OGC Filter standard a group of functions that test (or query) the dimensionality of a geometry (whether it's a point, a line, a surface, etc.). A rule in SE could then make use of one of those functions in a filter expression to determine the delineation of a feature.

1.1.4 Other issues related to the representation of S-52 symbols in SE

[TENET] seems to indicate that there are other problematic areas in SE in addition to those discussed in the report. Perhaps it would be useful to obtain the complete report on AML symbology (of which [TENET] is a summary) from TENET Technology in order to understand what those other areas are.

1.2 Other issues

1.2.1 Graphic stroke

In the current text (11.1.3), the role of the two "hot spot points" for a graphic stroke is unclear. The following paragraph:

> The Graphic sub-element specifies the linear graphic. Proper stroking with a linear graphic requires two “hot-spot” points within the space of the graphic to indicate where the rendering line starts and stops. In the case of raster images with no special mark-up, this line will be assumed to be middle pixel row of the image, starting from the first pixel column and ending at the last pixel column.

seems to mean that the renderer must extract the pixels of the image that lie in the segment between the two hot spots and must copy those pixels onto the map.

Is this functionality still useful after the introduction of single graphic strokes and compound strokes? Should it be made obsolete in a future version of the standard?

1.2.2 Point symbolizer example

The example on page 28 (11.3.3) is totally incorrect, as it talks about a point symbolizer composed of two graphics one of which is displaced, shows the (incorrect) XML, and shows the (impossible) result.

A point symbolizer has at most one graphic, which contains zero or more external graphics and/or marks all of which "provide the equivalent graphic in multiple formats" (p. 24), and the displacement can only be specified for the graphic as a whole.

This example needs to be either deleted or replaced with a correct example.

1.2.3 SE example documents

All the SE example documents (nine files) in the SE folder under SCHEMAS_OPENGIS_NET contain an `xsi:schemaLocation` attribute with an incorrect value.

The attribute is set as follows:

```
xsi:schemaLocation="http://www.opengis.net/se/1.1.0/Symbolizer.xsd"
```

According to [W3C XML Schema], 4.3.2 (item 3), an `xsi:schemaLocation` attribute must list one or more *pairs* of URIs. In each pair, the first URI is a namespace name and the second URI indicates the location of a schema document for that namespace name. In all the SE example documents, instead, the `xsi:schemaLocation` attribute contains just one URI (a location). This needs to be corrected.

2 References

[AML] AML Symbology Guidance, Version 1.0, United Kingdom Hydrographic Office

[S-52] Specifications for Chart Content and Display Aspects of ECDIS, IHO, 5th Edition, 1996 (amended 1999)

[TENET] Report: Some Unresolved Issues with the OGC Symbology Encoding (SE), Tenet Technology Ltd., 18 July 2008

[SE] OGC 05-077r4, Symbology Encoding Implementation Specification, v.1.1.0, Open Geospatial Consortium, 2006

[CW ER SE] OGC 09-016, OWS-6 Symbology Encoding (SE) Changes

[W3C XML Schema] XML Schema Part 1: Structures Second Edition, W3C Recommendation, 28 October 2004.