Paper for Consideration by the S-100 Working Group

Recommended Changes to the S-100 Portrayal

Submitted by:	SPAWAR Atlantic
Executive Summary:	Provides recommendations to address issues discovered while implementing the current S-100 Portrayal design within a generic S-100 viewer.
Related Documents:	S-100 Specification
Related Projects:	IHO S-100/S-101 Test Bed Project

Introduction / Background

This paper provides recommendations intended to correct issues with the S-100 Portrayal design and implementation. These issues were discovered while attempting to create a generic, product-agnostic S-100 viewer application, capable of portraying S-100 based datasets without the use of product specific knowledge. A generic S-100 viewer should not require code based on information contained within S-101 or any other S-100 derived product specification.

Note that the recommendations provided in this paper apply whether XSLT or Lua is used for processing of the portrayal rules.

Analysis / Discussion

1. We recommend removal of the *productId* and *version* attributes from the portrayal catalogue schema (highlighted entries below). *S100_ProductSpecification* from the *S100_CatalogueMetadata* should be used to determine the appropriate product type and version. The duplication of this metadata is a potential source of error. If it is decided to retain the *productId* and *version* attributes, we recommend a *versionDate* attribute be added to match *S100_CatalogueMetadata:S100_ProductSpecification*.

   Note that removal of these attributes implies that the catalogue must be delivered as part of an exchange set. Whenever a new catalogue is created an exchange set containing the catalogue must also be created for delivery to catalogue consumers. The exchange set does not need to contain datasets or other catalogues.

   9-14.3.1 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	Rules	Container of XSLT rule file references	1	Rules

Part 4a - S100_CatalogueMetadata

Role Name	Name	Description	Mult	Type	Remarks
Class	S100_CatalogueMetadata		-	-	-
Attribute	filename	The name for the catalogue	1..*	CharacterString	
…	…	…	…	…	…
Attribute	productSpecification	The product specification used to create this file	1..*	S100_ProductSpecification	

…	…	…	…	…	…

Part 4a - S100_ProductSpecification

Role Name	Name	Description	Mult	Type	Remarks
Class	S100_ProductSpecification	The Product Specification contains the information needed to build the specified product	-	-	-
Attribute	name	The name of the product specification used to create the datasets	1	CharacterString	
Attribute	*number (proposed)*	*The number of the product specification*	*1*	*Integer*	*e.g. 101*
Attribute	version	The version number of the product specification	1	CharacterString	
Attribute	date	The version date of the product specification	1	Date	

2. In order to support S-100 based portrayal, rather than product specific portrayal, it is necessary to modify the portrayal input schema so that it can model all S-100 datasets without requiring product specific extensions. Currently each product type can modify the S-100 feature and spatial models however they choose, by extending the types defined in S-100 9-A-1.

   For instance, 9-B-3 recommends that *s100:Point* be extended with an element defining an association to *spatialQuality*. However, a generic viewer only knows about *s100:Point*, and therefore would not know to add the association to *spatialQuality*.

   In order to be able to generically provide associations between spatial elements and information types, we recommend modifying the 9-A-1 *MaskedRelation* type so that it contains zero or more *InformationAssociations*. The xsd for this change is shown below, with the new content highlighted:

```
<xs:complexType name="MaskedRelation">
        <xs:complexContent>
                <xs:extension base="SpatialRelation">
                <xs:sequence>
                        <xs:element name="spatialInformation" type="InformationAssociation"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="mask" type="xs:boolean" default="false"/>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

   Currently, the testbed viewer does not add *spatialQuality* to the portrayal input because the method for doing so is defined in S-101. With the recommended change, we will be able to pass *spatialQuality* to the portrayal without hard coded knowledge of the S-101 specific portrayal input schema.

   If the change is accepted, the example shown in 9-B-3 is no longer necessary and should be removed.

3. We recommend that 9-7.5 Objects clarify that a spatial identifier must be unique for all spatial elements regardless of type (Curve, Point, etc.), or the XSD should be modified to require this. It is not always possible to infer from the portrayal output drawing instructions which spatial type is being referenced.

   For example, note that 9-11.2.2 *DrawingInstruction:spatialReference:reference* is a reference to a spatial, but the type is not specified. The type cannot necessarily be inferred from *DrawingInstruction:featureReference*, because the feature may have multiple associated spatial types.

Recommended change, with new content highlighted:

9-7.5 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 identifiable within a data set. This is done by the attribute id.

In the product specific schemas constraints can be made on this identifier, in particular by the use of the <xs:key> and <xs:keyref> elements. Spatial object identifiers must be unique for all spatial instances within the dataset, not just for spatial instances of a particular spatial type.

4. We recommend adding support for constraining the values allowed for an S-100 9-14.3.20 *ContextParameter* (also defined in XSD in 9-A-5) by adding the following optional elements: *minAllowedValue*, *maxAllowedValue*, *enumeratedValue*.

   This change would allow the application to restrict user input to reasonable values, and provide for selection from portrayal context parameters with specified enumerated values.

   The 9-14.3.20 change with new content highlighted:

9-14.3.20 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 Parameter	1	ParameterType
Attribute	default	A default value for the Parameter	1	any
Attribute	minAllowedValue	The smallest allowed value	0..1	any
Attribute	maxAllowedValue	The largest allowed value	0..1	any
Attribute	enumeratedValue	Allowed values are restricted to enumerated values when present	0..*	any

The 9-A-5 XSD for this change is shown below, with the new content highlighted:

```
<!-- Class for a context parameter -->
<xs:complexType name="ContextParameter">
    <xs:complexContent>
        <xs:extension base="CatalogItem">
            <xs:sequence>
                <xs:element name="type" type="ParameterType"/>
                <xs:element name="default" type="xs:anyType"/>
                <xs:element name=”minAllowedValue” type=”xs:anyType” minOccurs=”0”/>
                <xs:element name=”maxAllowedValue” type=”xs:anyType” minOccurs=”0”/>
                <xs:element name=”enumeratedValue” type=”xs:anyType” minOccurs=”0”
                maxOccurs=”unbounded”/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

5. The 9-14.3.3 *ExternalFile* class should clarify whether the *fileName* contains the relative path to the file. The current S-101 portrayal catalogue only contains filenames, and the application must add the (implied) relative path information (“ColorProfiles/”, “Fonts/”, etc.). However, the sample catalogue in 9-A-7.1.6 has *fileName* entries such as: file://colorProfiles/Day.xml, with a defined type of xs:anyURI.

   We recommend clarifying that the filename only stores the name of a file, and the path is implied by the files use (“ColorProfiles/”, etc).

9-14.3.3 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. The relative path to the file is implied by the enclosing catalogue section (e.g. colorProfiles implies "ColorProfiles/")	1	string
Attribute	fileType	Type type of the file	1	FileType
Attribute	fileFormat	The format of the file	1	FileFormat

We further recommend changing the type of *fileName* in 9-A-5 from xs:anyURI to "xs:string" to agree with 9-14.3.3. With these changes the path to the Day colour profile should be encoded as "Day.xml", with an implied path of "ColorProfiles/".

9-A-5

```
<!-- catalogue item for an external file -->
<xs:complexType name="ExternalFile">
        <xs:complexContent>
                <xs:extension base="CatalogItem">
                        <xs:sequence>
                                ~~<xs:element name="fileName" type="xs:anyURI"/>~~
                                <xs:element name="fileName" type="xs:string"/>
                                <xs:element name="fileType" type="FileType"/>
                                <xs:element name="fileFormat" type="FileFormat"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

9-A-7.1.6 Sample Catalogue XML

Requires updates to all fileName elements, such as:

~~<fileName>file://pixmaps/bla.xml</fileName>~~
<fileName>bla.xml</fileName>

6. It is currently not possible for a generic application to know how to change the SVG symbol colours when a new S-100 9-A-6 *Palette* is selected. We have created svgStyle.css files for each colour palette in S-101, but the portrayal catalogue schema provides no mechanism to associate these files with a *Palette*. In order to provide this association, we recommend adding a stylesheet section to the portrayal_catalogue.xml such as:

```
<styleSheets>
        <styleSheet id="daySVGStyle.css" paletteName="Day">
                <description>
                        <name>Day Style Sheet</name>
                        <description>Style sheet for use with the day colour palette</description>
                        <language>en</language>
                </description>
                <fileName>Symbols/daySVGStyle.css</fileName>
                <fileType>StyleSheet</fileType>
                <fileFormat>CSS</fileFormat>
        </styleSheet>
        <!-- Add more stylesheets here... -->
< /styleSheets>
```

The stylesheet *paletteName* attribute must match the *id* of a *S-100 9-A-6 Palette* so that the application can associate the two.

9-14.3.x StyleSheet

Role Name	Name	Description	Mult	Type
Class	StyleSheet	A CSS file which defines drawing rules for a particular color palette	-	-
Subtype of	ExternalFile	See ExternalFile	-	-
Attribute	paletteName	Matches the name of a single colour palette, e.g. “Day”	1	string

9-14.3.y StyleSheets

Role Name	Name	Description	Mult	Type
Class	StyleSheets	A container of StyleSheets	-	-
Role	styleSheet	The file reference. The type is CSS.	0..*	StyleSheet

9-14.3.24 FileFormat

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

9-14.3.25 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 XSLT rules
Enumeration	styleSheet	A CSS file

Note that this change requires adding a new *StyleSheet* complexType to 9-A-5. *StyleSheet* should extend *ExternalFile,* and add attribute *paletteName*. “CSS” should be added to 9-A-5 *FileType*, and “StyleSheet” should be added to 9-A-5 *FileFormat*.

7. Currently, the S-101 SVG symbols contain a single persistent stylesheet reference, as noted in 9-C-3.1.1 CSS:

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

We recommend modifying this so that the SVG symbols contain a preferred and multiple alternate stylesheet references:

```
<?xml-stylesheet href="daySVGStyle.css" title="daySVGStyle.css" type="text/css"?>
<?xml-stylesheet href="duskSVGStyle.css" alternate="yes" title="duskSVGStyle.css" type="text/css"?>
<?xml-stylesheet href="nightSVGStyle.css" alternate="yes" title="nightSVGStyle.css" type="text/css"?>
```

This mechanism is capable of supporting any number of color palettes, and allows for non-ECDIS applications to more easily support alternate color palettes. In order to make the SVG symbols more portable the stylesheet information can be added either by the catalogue maintainer when the portrayal catalogue is created, or by the application when the portrayal catalogue is installed. We prefer the first option, as it reduces the implementation burden for application developers.

8. We recommend adding a mechanism to specify a default or preferred colour *Palette*. Currently the application must make an arbitrary selection of a default palette, or code product specific knowledge. We recommend modifying 9-14.3.6 *ColorProfiles* as follows:

9-14.3.6 ColorProfiles

Role Name	**Name**	**Description**	**Mult**	**Type**
Class	ColorProfiles	A container of colour profile file references	-	-
Attribute	default	Specifies a default colour palette (not a default colorProfile)	1	string
Role	colorProfile	The file reference. The type is XML	0..*	ExternalFile

This change requires modifying 9-A-5 *ColorProfiles* as follows:

```
<xs:complexType name="ColorProfiles">
        <xs:sequence>
                <xs:element name="colorProfile" type="ExternalFile" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <!—The default attribute matches a single color palette id. -->
        <xs:attribute name="default" type="xs:string"/>
</xs:complexType>
```

An example portrayal_catalogue.xml with default palette set to "Day":

```
<portrayalCatalog …>
        <pixmaps/>
        …
        <colorProfiles default="Day">
                <!—Note: a color profile can contain more than one color palette -->
                <colorProfile id="1">
                        <description>
                                <name>Color Profile 1</name>
                                <description>A file containing the day and dusk palettes</description>
                                <language>en</language>
                        </description>
                        <fileName>colorProfile1.xml</fileName>
                        <fileType>ColorProfile</filetype>
                        <fileFormat>XML</fileFormat>
                </colorProfile>
                <colorProfile id="2">
                        <description>
                                <name>Color Profile 2</name>
                                <description>A file containing the night palette</description>
                                <language>en</language>
                        </description>
                        <fileName>colorProfile2.xml</fileName>
                        <fileType>ColorProfile</filetype>
                        <fileFormat>XML</fileFormat>
                </colorProfile>
        </colorProfiles>
        …
</ portrayalCatalog>
```

9. Related to the previous recommendation, we recommend simplifying the catalogue structure with regard to colour support. Currently, *Color* entries must be duplicated in each *ColorProfile.* The schema also allows multiple *Palette's* to be stored within a single *ColorProfile*, which means that selecting a *ColorProfile* is not sufficient for choosing a colour palette.

   For example, the following is currently valid, and must be supported by applications:

```
<portrayalCatalog …>
        <pixmaps/>
        <colorProfiles>
                <colorProfile id="1">
                        …
                        <fileName>colorProfile1.xml</fileName>
                </colorProfile>
                <colorProfile id="2">
                        …
                        <fileName>colorProfile2.xml</fileName>
                </colorProfile>
        </colorProfiles>
</portrayalCatalog>

<!-- Color Profile 1, contained in colorProfile1.xml -->
<colorProfile>
        <colors>
                …
        </colors>
        <palette name="Day">
                …
        </palette>
        <palette name="Dusk">
                …
        </palette>
</colorProfile>

<!-- Color Profile 2, contained in colorProfile2.xml -->
<colorProfile>
        <colors>
                <!-- These colors must all be duplicates of the entries in colorProfile1.xml -->
                …
        </colors>
        <palette name="Night">
                …
        </palette>
</colorProfile>
```

   We recommend modifying the structure so that there is a single copy of *Colors*, and each *ColorProfile* stores a single *Palette*. This change allows the selection of a *ColorProfile* to by synonymous with the selection of a *Palette*. An example of this change follows:

```
<portrayalCatalog …>
        <pixmaps/>
        <colorFile>
                <fileName>colors.xml</fileName>
                <fileType>colors</fileType>
                <fileFormat>XML</fileFormat>
        </colorFile>
        <colorP~~rofiles~~Palettes default="Day">
                <color~~Profile~~Palette id="Day">
                        <description>
                                <name>Day Color Palette</name>
                                <description>A file containing the day color palette</description>
                <language>en</language>
            </description>
            <fileName>dayColorProfile.xml</fileName>
            <fileType>ColorProfile</filetype>
            <fileFormat>XML</fileFormat>
        </color~~Profile~~Palette>
        <color~~Profile~~Palette id="Dusk">
            <description>
                <name>Dusk Color Palette</name>
                <description>A file containing the dusk color palette</description>
                <language>en</language>
            </description>
            <fileName>duskColorProfile.xml</fileName>
            <fileType>ColorProfile</filetype>
            <fileFormat>XML</fileFormat>
        </color~~Profile~~Palette>
    </color~~Profile~~Palettes>
</portrayalCatalog>
```

Supporting schema changes:

9-14.3.1 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
…	…	…	…	…
Role	color~~Profiles~~Palettes	Container of XML Colour ~~Profile~~Palette file references	1	Color~~Profiles~~Palettes
…	…	…	…	…
Role	colorFile	A file containing color token names and descriptions	0..1	ExternalFile
…	…	…	…	…

9-14.3.25 FileType

Role Name	Name	Description
Type	FileType	The type of an external file
Enumeration	Font	A font file.
…	…	…
Enumeration	colors	A file containing color token names and descriptions.

9-14.3.6 Color~~Profiles~~Palettes

Role Name	Name	Description	Mult	Type
Class	Color~~Profiles~~Palettes	A container of colour ~~profile~~palette references	-	-
Attribute	default	Specifies a default colour palette	1	string
Role	color~~Profile~~Palette	The file reference. The type is XML	0..*	ExternalFile

9-A-6 *Palette* should also be modified as follows:

```
<xs:complexType name="Palette">
    <xs:sequence>
        <xs:element name="item" type="PaletteItem" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    ~~<xs:attribute name="name" type="xs:string" use="required"/>~~
```

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

10. We recommend adding a mechanism to specify a default or preferred *DisplayMode*. Currently, the application must make an arbitrary selection of a default *DisplayMode*, or code product specific knowledge. We recommend modifying 9-14.3.15 *DisplayModes* as follows:

9-14.3.15 DisplayModes

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

This change requires modifying 9-A-5 *DisplayModes* as follows:

```
<xs:complexType name="DisplayModes">
        <xs:sequence>
                <xs:element name="displayMode" type="DisplayMode" minOccurs="0"
        maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- The default attribute matches a single DisplayMode id -->
        <xs:attribute name="default" type="xs:string"/>
</xs:complexType>
```

An example portrayal_catalogue.xml with default displayMode set to the standard display:

```
<portrayalCatalog …>
        <pixmaps/>
        …
        <displayModes default="2">
                <displayMode id="1">
                        <description>
                                <name>Base</name>
                                …
                        </description>
                        <viewingGroupLayer>1</viewingGroupLayer>
                </displayMode>
                <displayMode id="2">
                        <description>
                                <name>Standard</name>
                                …
                        </description>
                        <viewingGroupLayer>1</viewingGroupLayer>
                </displayMode>
                <displayMode id="3">
                        <description>
                                <name>Other</name>
                                …
                        </description>
                        <viewingGroupLayer>1</viewingGroupLayer>
                </displayMode>
        </displayModes>
</portrayalCatalog>
```

11. There should be a way for the portrayal to specify a default *TopLevelTemplate*. We recommend adding an attribute to the container to allow this.

    9-14.3.21 Rules

Role Name	Name	Description	Mult	Type
Class	Rules	A container of XSLT rule file references	-	-
Attribute	default	Identifies the default TopLevelTemplate	1	string
Role	ruleFile	Reference to an XSLT file containing template rules	1..*	RuleFile

This change requires modifying 9-A-5 *Rules* as follows:

```
<xs:complexType name="Rules">
    <xs:sequence>
        <xs:element name="ruleFile" type="RuleFile" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- The default attribute matches the id of a single TopLevelTemplate rule file -->
    <xs:attribute name="default" type="xs:string"/>
</xs:complexType>
```

12. Part 9, and in particular parts 9-A and 9-B, needs to be reviewed and updated. We recommend a small group be formed to provide an update based on the actions resulting from the current meeting.

Current S-100 Viewer Limitations

In order to work around the challenges noted in this paper, the current version of the S-100 testbed viewer has the following limitations:

- Importing of the portrayal catalogue is not supported.
- *S-101:spatialQuality* is not passed to the S-101 portrayal.
- User modification of portrayal context parameters is unconstrained.
- Folder names for the various portrayal catalogue elements is hardcoded ("Symbols/", "ColorProfiles/", etc.)
- Symbols always use the day palette, regardless of the selected palette.
- The default value for *DisplayMode*, *ColorPalette*, and *TopLevelTemplate* is the first entry found in the catalogue.

Recommendations

- Modify S-100 Part 9 as described herein (items 1 through 11)
- Form a group to review and update S-100 Part 9 parts 9-A and 9-B (item 12)

Action Required of S-100 WG

The S-100 working group is invited to:

a. note the paper
b. discuss the recommendations
c. take action on each recommendation