

SBML Level 3 Package: Render ('render')

Ralph Gauges

gauges@hs-albsig.de sven.sahle@bioquant.uni-heidelberg.de

Hochschule Albstadt-Sigmaringen
Sigmaringen, Germany

Sven Sahle

Modelling of Biological Processes
University of Heidelberg
Heidelberg, Germany

Katja Wengler

k.wengler@herts.ac.uk

University of Hertfordshire
Computer Science
Hatfield, UK

Frank T. Bergmann

fbergman@caltech.edu

Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA, US

Sarah M. Keating

skeating@ebi.ac.uk

European Bioinformatics Institute
EMBL-EBI
Hinxton, Cambs, UK

Version 1, Release 1

April 15, 2015

This is a working draft of release 1 of the specification for the package "render" and not a normative document. Please send feedback to the Package Working Group mailing list at sbml-render@lists.sourceforge.net.

The latest release, past releases, and other materials related to this specification are available at [http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Rendering_\(render\)](http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Rendering_(render))

This release of the specification is available at



Contents

1	Introduction and motivation	4
1.1	Proposal corresponding to this package specification	4
1.2	Tracking number	4
1.3	Package dependencies	4
1.4	Document conventions	4
2	Background	6
2.1	Design decisions	6
3	Package syntax and semantics	7
3.1	Namespace URI and other declarations necessary for using this package	7
3.2	Primitive data types	8
3.2.1	Type <code>StyleType</code>	8
3.2.2	Type <code>GradientSpreadMethod</code>	8
3.2.3	Type <code>FillRule</code>	8
3.2.4	Type <code>FontFamily</code>	9
3.2.5	Type <code>FontWeight</code>	9
3.2.6	Type <code>FontStyle</code>	9
3.2.7	Type <code>VTextAnchor</code>	9
3.2.8	Type <code>HTextAnchor</code>	10
3.2.9	Type <code>RelAbsVector</code>	10
3.3	General features	10
3.3.1	Uniqueness of ids	10
3.3.2	Default Values	12
3.4	Extended elements from the Layout package	13
3.4.1	The extended <code>GraphicalObject</code> class	13
3.4.2	The extended <code>ListOfLayouts</code> class	14
3.4.3	The extended <code>Layout</code> class	14
3.5	Render Information	15
3.5.1	The <code>RenderInformationBase</code> class	15
3.5.2	The <code>LocalRenderInformation</code> class	16
3.5.3	The <code>GlobalRenderInformation</code> class	16
3.6	Styles	17
3.6.1	The <code>Style</code> class	18
3.6.2	The <code>GlobalStyle</code> class	19
3.6.3	The <code>LocalStyle</code> class	19
3.7	Colors and Gradients	19
3.7.1	The <code>ColorDefinition</code> class	19
3.7.2	The <code>GradientBase</code> class	20
3.7.3	The <code>GradientStop</code> class	20
3.7.4	The <code>LinearGradient</code> class	22
3.7.5	The <code>RadialGradient</code> class	22
3.8	Transformation	23
3.8.1	The <code>Transformation</code> class	23
3.8.2	The <code>Transformation2D</code> class	24
3.9	GraphicalPrimitives	25
3.9.1	The <code>GraphicalPrimitive1D</code> class	25
3.9.2	The <code>GraphicalPrimitive2D</code> class	25
3.9.3	The <code>RenderCurve</code> class	26
3.9.4	The <code>RenderPoint</code> class	27
3.9.5	The <code>RenderCubicBezier</code> class	27
3.10	Geometric Shapes	28
3.10.1	The <code>Polygon</code> class	30
3.10.2	The <code>Rectangle</code> class	30
3.10.3	The <code>Ellipse</code> class	31
3.10.4	The <code>Text</code> class	31
3.10.5	The <code>Image</code> class	32
3.10.6	The <code>RenderGroup</code> class	33
3.11	The <code>LineEnding</code> class	33
4	Illustrative examples of the Render syntax	36
5	Best practices	59
5.1	Text	59
5.2	Image	59

6	Future development	60
6.1	Values for StyleType	60
6.2	3D drawings	60
6.3	Text	60
6.4	Image	60
A	Text Anchor Examples	61
A.1	Vertical Text Anchor Examples	61
A.1.1	Top	61
A.1.2	Bottom	61
A.1.3	Middle	62
A.1.4	Baseline	62
A.2	Horizontal Text Anchor Examples	63
A.2.1	Start	63
A.2.2	Middle	63
A.2.3	End	64
B	Transformations	65
B.1	Translation	65
B.2	Scaling	65
B.3	Rotation	65
B.4	Skewing	65
C	Resolving render information	67
C.1	Mapping line endings to curves	67
C.2	Style resolution	71
C.3	Role resolution	71
C.4	Style information for reaction glyphs and species reference glyphs	71
C.5	Style information for text glyphs	72
D	Validation of SBML documents	73
D.1	Validation and consistency rules	73
	References	96

1 Introduction and motivation

1.1 Proposal corresponding to this package specification

This specification for Rendering in SBML Level 3 Version 1 is based on the proposal, by this documents authors, located at the following URL:

http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Rendering

The tracking number in the SBML issue tracking system (SBML Team, 2010) for Render package activities is 234. The version of the proposal used as the starting point for this specification is the version of May 2011. Previous versions of the current proposal are:

Version 5 (May 2011)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20110525/sbml-render-specification-20110525.pdf>

Version 4 (October 2009)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20091029/SBMLRenderExtension-20091029.pdf>

Version 3 (January 2008)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20080130/RenderExtensionDraft-20080130.pdf>

Version 2 (March 2010)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20070309/RenderExtensionDraft-20070309.pdf>

Version 1 (October 2006)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20061012/RenderExtensionDraft-20061012.pdf>

Version 0.2 (October 2003)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20031028/SBMLRenderExtension-20031028.pdf>

Version 0.1 (September 2003)

<http://otto.bioquant.uni-heidelberg.de/sbml/level2/20030911/sbml-render-extension-20030918.pdf>

Details of earlier independent proposals are provided in [Section 2](#).

1.2 Tracking number

As initially listed in the SBML issue tracking system under:

<http://sourceforge.net/p/sbml/sbml-specifications/234/>.

1.3 Package dependencies

The Render package adds additional classes to SBML Level 3 Version 1 Core and extends the SBML Level 3 Layout package.

1.4 Document conventions

Following the precedent set by the SBML Level 3 Core specification document, we use UML 1.0 (Unified Modeling Language; Eriksson and Penker 1998; Oestereich 1999) class diagram notation to define the constructs provided by this package. We also use color in the diagrams to carry additional information for the benefit of those viewing the document on media that can display color. The following are the colors we use and what they represent:

- **Black:** Items colored black in the UML diagrams are components taken unchanged from their definition in the SBML Level 3 Core specification document.
- **Green:** Items colored green are components that exist in SBML Level 3 Core, but are extended by this package. Class boxes are also drawn with dashed lines to further distinguish them.
- **Blue:** Items colored blue are new components introduced in this package specification. They have no equivalent in the SBML Level 3 Core specification.
- **Red:** Items colored red are new components introduced in this package specification but without full definition. They have no equivalent in the SBML Level 3 Core specification and will be defined fully elsewhere in the text.

We also use the following typographical conventions to distinguish the names of objects and data types from other entities; these conventions are identical to the conventions used in the SBML Level 3 Core specification document:

AbstractClass: Abstract classes are classes that are never instantiated directly, but rather serve as parents of other object classes. Their names begin with a capital letter and they are printed in a slanted, bold, sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of that class. (However, for classes that are unchanged from their definitions in SBML Level 3 Core, the class names are not hyperlinked because they are not defined within this document.)

Class: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, sans-serif typeface. In electronic document formats, the class names are also hyperlinked to their definitions in this specification document. (However, as in the previous case, class names are not hyperlinked if they are for classes that are unchanged from their definitions in the SBML Level 3 Core specification.)

Something, otherThing: Attributes of classes, data type names, literal XML, and generally all tokens *other* than SBML UML class names, are printed in an upright typewriter typeface. Primitive types defined by SBML begin with a capital letter; SBML also makes use of primitive types defined by XML Schema 1.0 ([Biron and Malhotra, 2000](#); [Fallside, 2000](#); [Thompson et al., 2000](#)), but unfortunately, XML Schema does not follow any capitalization convention and primitive types drawn from the XML Schema language may or may not start with a capital letter.

For other matters involving the use of UML and XML, we follow the conventions used in the SBML Level 3 Core specification document.

2 Background

In 2003 the authors proposed an extension to the SBML file format that allowed programs to include layout and render information in SBML files to store one or more graphical representations of the SBML model. During the discussions on the SBML mailing list, it soon became evident that a consensus for both layout and render information would not be reached easily, therefore the layout specification was separated from the render part of the specification and concentrated on the inclusion of layout information into SBML files. The Layout Specification has since been publicly accepted as SBML Level 3 Package.

This document describes now an extension to the SBML Layout Package that describes the precise rendering of elements. Where the Layout package only describes the size and location of objects, the Render package complements this description by detailing precisely how they are to be rendered.

2.1 Design decisions

The render extension is based on the existing layout extension. Secondly, the render extension is made as flexible as possible in order to not impose any artificial limits on how programs can display their reaction networks.

The render extension is independent of the underlying SBML model as well as of the layout extension, thus the render information will be stored as one or more separate blocks. There can be one block of render information that applies to all layouts and an additional block for each layout. For SBML Level 2 this render information will be stored in the annotation of the `listOfLayouts` element or the annotation of a `layout` element respectively.

The render information consists of a set of styles that are associated with objects from the Layout Package either by a list of `ids` of layout objects or by roles of layout objects or `ids` of their corresponding model elements. For example a style can be defined that applies to all **SpeciesReference** objects or to all objects that have the **role product**.

Global render information included in the annotation of the `listOfLayouts` element will only be able to define styles that associate render information with roles of elements, it can not associate styles with individual objects from a layout via their `ids`.

Many of the elements used in the current render specification are based on corresponding elements from the SVG specification. This allows us to easily convert a combination of layout information and render information into a SVG drawing. At the same time we profit from the work that has already been done while creating the SVG specification.

3 Package syntax and semantics

In this section, we define the syntax and semantics of the Render package for SBML Level 3 Version 1 Core. We expound on the various data types and constructs defined in this package, then in [Appendix 4 on page 36](#), we provide complete examples of using the constructs in example SBML models.

3.1 Namespace URI and other declarations necessary for using this package

Every SBML Level 3 package is identified uniquely by an XML namespace URI. For an SBML document to be able to use a given SBML Level 3 package, it must declare the use of that package by referencing its URI. The following is the namespace URI for this version of the Render package for SBML Level 3 Version 1:

`"http://www.sbml.org/sbml/level3/version1/render/version1"`

In addition, SBML documents using a given package must indicate whether understanding the package is required for complete mathematical interpretation of a model, or whether the package is optional. This is done using the attribute **required** on the `<sbml>` element in the SBML document. For the Render package the value of the required attribute is `"false"`.

The following fragment illustrates the beginning of a typical SBML model using SBML Level 3 Version 1 and this version of the Render package (note, that the Layout package is also needed):

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1" layout:required="false"
  xmlns:render="http://www.sbml.org/sbml/level3/version1/render/version1" render:required="false"
>
```

Originally the layout and render extension were developed for use with SBML Level 2 files, where the information was stored in annotations to SBML models, layout lists and layouts. The namespace for the version of the SBML render extension for SBML level 2 is:

`"http://projects.eml.org/bcb/sbml/render/level2"`

An example using the render extension in this context would look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
  <model id="model1" name="Model_with_L2_Render_Annotation">
    <annotation>
      <listOfLayouts xmlns="http://projects.eml.org/bcb/sbml/level2">
        <layout id="layout1">
          <annotation>
            <listOfRenderInformation xmlns="http://projects.eml.org/bcb/sbml/render/level2">
              ...
            </listOfRenderInformation>
          </annotation>
          ...
        </layout>
        ...
      </listOfLayouts>
      ...
    </annotation>
    ...
  </model>
</sbml>
```

3.2 Primitive data types

Section 3.1 of the SBML Level 3 specification defines a number of primitive data types and also uses a number of XML Schema 1.0 data types (Biron and Malhotra, 2000). We assume and use some of them in the rest of this specification, specifically **boolean**, **integer**, **double**, **ID**, **SId**, **SIdRef**, and **string**. The Render package defines other primitive types; these are described below.

3.2.1 Type **StyleType**

The type **StyleType** is used by **LocalStyle** and **GlobalStyle** elements, in order to apply a particular **Style** to a **GraphicalObject**. This is done via the **typeList** attribute that uses the **StyleType** as its data type.

A valid **StyleType** instance is a combination of one or more of the following values separated by white spaces:

- "COMPARTMENTGLYPH",
- "SPECIESGLYPH",
- "REACTIONGLYPH",
- "SPECIESREFERENCEGLYPH",
- "TEXTGLYPH",
- "GENERALGLYPH",
- "GRAPHICALOBJECT"
- "ANY"

The **ANY** keyword specifies that this styles applies to any type of glyph and would be equivalent to listing all the other keywords.

3.2.2 Type **GradientSpreadMethod**

The type **GradientSpreadMethod** is being used by **GradientBase** elements to decide how gradients propagate over the whole element they are applied to. It is an enumeration consisting of the following three values called **pad**, **reflect** or **repeat**:

- **pad**: the gradient color at the endpoint of the vector defines how the gradient is continued beyond that point (default value).
- **reflect**: the gradient continues from end to start and then from start to end again and again.
- **repeat**: the gradient pattern is repeated from start to end over and over again.

3.2.3 Type **FillRule**

The type **FillRule** describes how a surface created by connecting points on a **Polygon** are to be filled when rendered. Allowed values for a valid instance of type **FillRule** are:

- **nonzero**
- **evenodd**

For a detailed description on how these values should be applied, we would like to refer you to the corresponding documentation in the SVG specification ¹.

¹ <http://www.w3.org/TR/SVG/painting.html#FillRuleProperty>

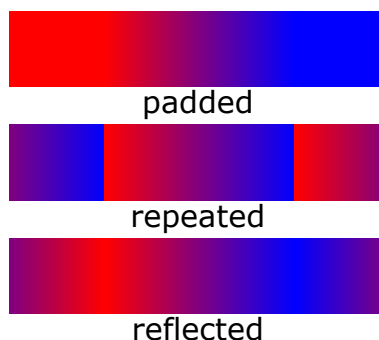


Figure 1: example of different SVG spreadMethod values

3.2.4 Type FontFamily

The **FontFamily** type gives a hint as to which font is to be used when rendering **Text** elements. This type extends the type **string**. The following values are pre-defined:

- serif,
- sans-serif
- monospace

However, applications are free to use the **FontFamily** to store the name of the font the writing application used as a string. It has not been a issue for reading applications to find a similar font.

3.2.5 Type FontWeight

The type **FontWeight** indicates whether the font is to be used in its normal form, or in its bold form. Consequently, the only values allowed for this enumeration are:

- bold
- normal

3.2.6 Type FontStyle

The type **FontStyle** determines whether a font is to be drawn use italic or normal styles. Thus the only allowed values are:

- italic
- normal

3.2.7 Type VTextAnchor

The type **VTextAnchor** allows models to specify how text elements are to be vertically aligned within their bounding box. This enumeration has the following allowed values:

- top,
- middle,
- bottom
- baseline

Examples illustration the use of the different **VTextAnchor** values are given in [Appendix A on page 61](#).

3.2.8 Type `HTextAnchor`

The type `HTextAnchor` defines the horizontal alignment of text elements. This enumeration can use the following values:

- `start`,
- `middle`
- `end`

Examples illustrating the use of the different `HTextAnchor` values are given in [Appendix A on page 61](#).

3.2.9 Type `RelAbsVector`

The position and size of render elements can be specified as a combination of an absolute value and a relative value. The absolute value is a numerical value in units of “**pt**” (1/72 inch) indicating the position of the object. The relative value is a percentage indicating the size of the object. All values are relative to the bounding box of the corresponding element in the layout. This bounding box basically specifies a canvas for the render elements to be drawn on.

In order to avoid populating the resulting XML with numerous attributes the Render package encodes this information in the `RelAbsVector` class with the two attributes `abs` and `rel` by extending the `string` such that it encodes optionally an absolute number first followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required.

Examples of the `RelAbsVector` construct for the `x` coordinate are shown in the table below.

string	Coordinate
<code>-5 + 100%</code>	5 points left of the right edge of the current bounding box.
<code>50%</code>	50% from the left edge of the current bounding box.
<code>2</code>	2 points from the left edge of the bounding box.

It should be noted that when applying transformations to elements with relative values, the relative values have to be converted to absolute values first.

3.3 General features

The render extension provides two locations where styles can be defined. First each layout can have its own set of render information located as a child element of the `Layout` element ([Figure 2](#)). This is considered to be **local** render information. Secondly, **global** render information objects can be located as child elements of the `ListOfLayouts` element ([Figure 3](#)).

It is important to note that each layout can have more than one set of local render information and that it is also possible to define more than one global style. Each style can also reference another style that complements it, this way the user can create styles that are based on other styles. In contrast to local styles, the global styles can not reference individual layout elements by an id, they can only define role based or type based styles.

3.3.1 Uniqueness of ids

Since local and global render information objects can reference other render information objects, programs creating render information need to make sure that all the ids are unique within the reference history. In other words, a render information object that references another render information object must make sure that none of its ids is equal to an id in any of the directly or indirectly referenced render information objects.

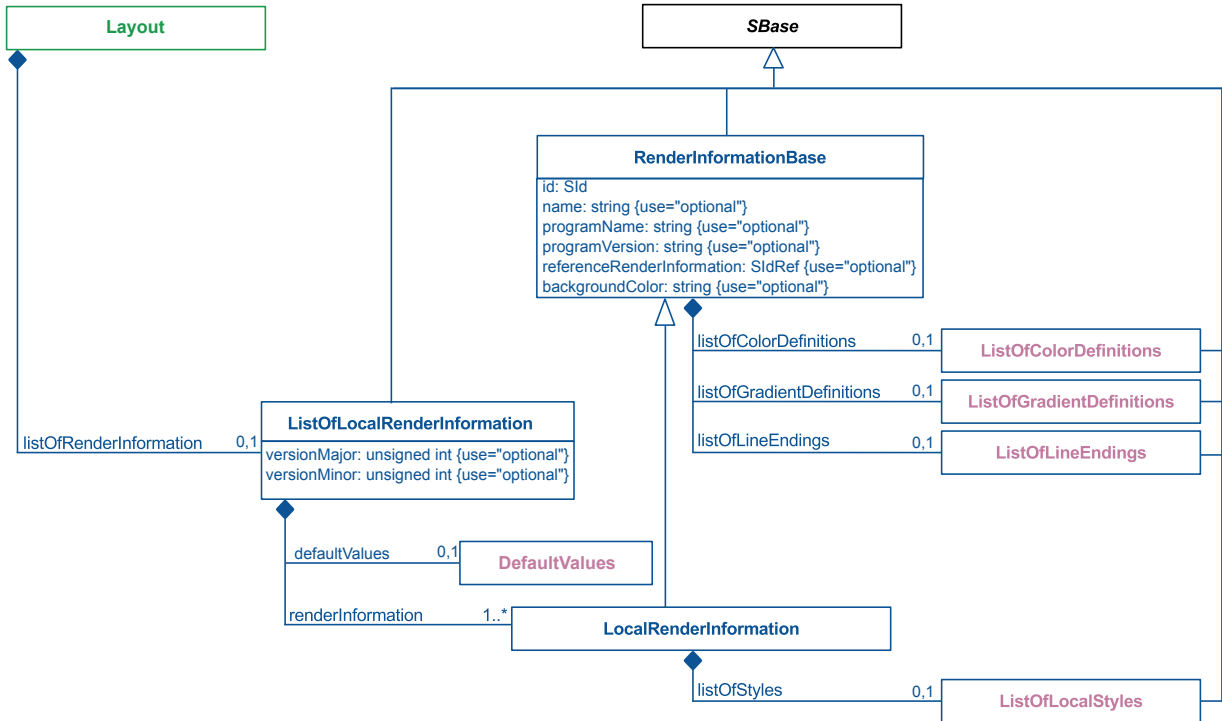


Figure 2: A UML representation of the extended **Layout** class for the **Render** package. See Section 1.4 for conventions related to this figure.

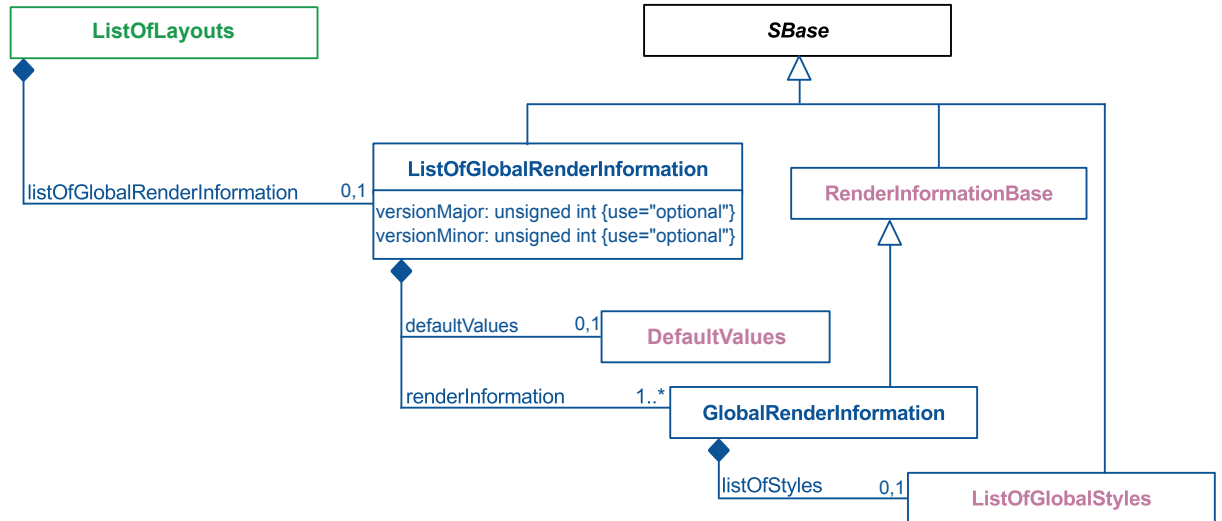


Figure 3: A UML representation of the extended **ListOfLayouts** object for the **Render** package. See Section 1.4 for conventions related to this figure.

An exception to this rule is that a **ColorDefinition** may have the same **id** as a **ColorDefinition** in a referenced style. In this case interpreting programs can assume that this **ColorDefinition** is supposed to override the **ColorDefinition** with the same **id** in the referenced render information object. Likewise it is also possible to override a **ColorDefinition** with a gradient (that is either a **LinearGradient** or a **RadialGradient**) and vice versa. **LineEnding** definitions on the other hand can only be replaced by other **LineEnding** definitions.

3.3.2 Default Values

Previously the render package specified default values and inheritance in a similar fashion to the specification used by SVG. However, in order to comply with the **SBML** development guidelines for Level 3 packages, we introduced a new class **DefaultValues** to encode these values within the model. The **DefaultValues** class can occur as a child of either the **ListOfGlobalRenderInformation** or a **ListOfLocalRenderInformation**.

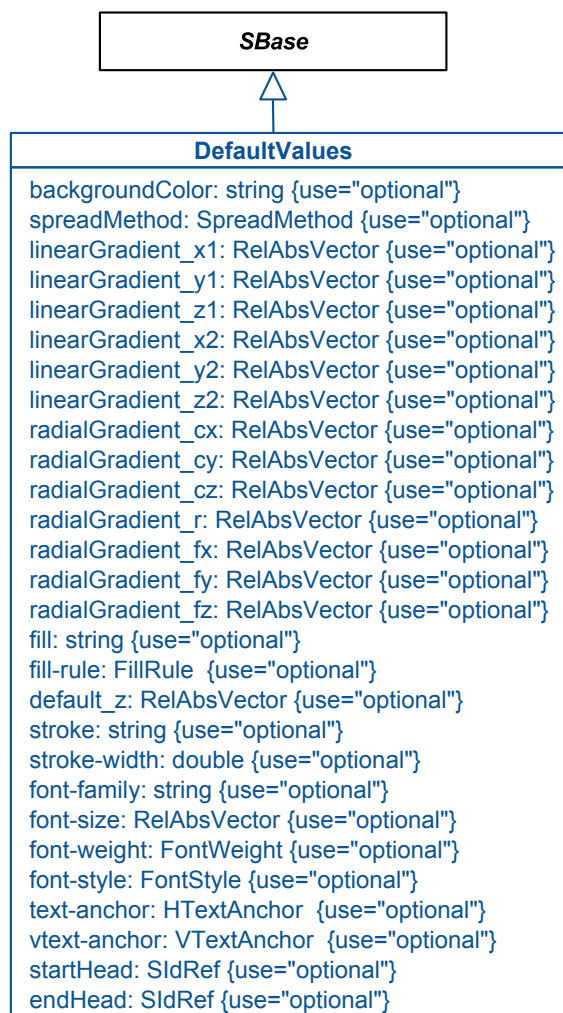


Figure 4: A UML representation of the **DefaultValues** class for the Render package. See [Section 1.4](#) for conventions related to this figure.

The values from the **DefaultValues** class are to be taken as default source for the values of any optional attribute that is not explicitly declared. An example on how to use the **DefaultValues** class is below. For the meaning of the individual attributes, please see the corresponding sections later in this document.

```

<defaultValues
  backgroundColor = "#FFFFFF"
  spreadMethod="pad"

  <!-- linear gradient -->
  linearGradient_x1 = "0%"
  linearGradient_y1 = "0%"
  linearGradient_z1 = "0%"
  linearGradient_x2 = "100%"
  linearGradient_y2 = "100%"
  linearGradient_z2 = "100%"

  <!-- radial gradient -->
  radialGradient_cx = "50%"
  radialGradient_cy = "50%"
  radialGradient_cz = "50%"
  radialGradient_r = "50%"
  radialGradient_fx = "50%"
  radialGradient_fy = "50%"
  radialGradient_fz = "50%"

  <!-- stroke -->
  stroke="none" stroke-width="0"

  <!-- fill -->
  fill="none" fill-rule="nonzero"

  <!-- font -->
  font-family = "sans-serif"
  font-size = "0"
  font-weight = "normal"
  font-style = "normal"
  text-anchor = "start"
  vtext-anchor = "top"

  default_z = "0.0"
/>

```

3.4 Extended elements from the Layout package

3.4.1 The extended *GraphicalObject* class

The Render package extends the **GraphicalObject** object from the Layout package with the addition of the **objectRole** attributes.

The objectRole attribute

A **GraphicalObject** has an optional attribute **objectRole** of type **string**. This attribute specifies with which **Style** the object should be rendered. In the example below a **SpeciesGlyph** is tagged with the role “SBO-0000285-clone” later on a style in a **GlobalRenderInformation** element includes that role in its **roleList** attribute and will be applied.

```

<speciesGlyph id="sGlyph_9" render:objectRole="SBO-0000285-clone"
  species="NAD" xmlns:render="http://www.sbml.org/sbml/level3/version1/render/version1">
  <boundingBox>
    <position x="138" y="247" />
    <dimensions width="36" height="24" />
  </boundingBox>
</speciesGlyph>

...

<style roleList="SBO-0000285-clone_N0-SBO-clone">
  <g stroke="black" stroke-width="2" fill="EPNBackgroundGradient">
    <ellipse cx="50%" cy="50%" cz="0.0" rx="50%" ry="50%" />
    <ellipse cx="50%" cy="50%" cz="0.0" rx="50%" ry="50%" />
  </g>
</style>

```

3.4.2 The extended *ListOfLayouts* class

The Render package extends the **ListOfLayouts** object from the Layout package with the addition of an optional **ListOfGlobalRenderInformation** object (Figure 3).

The *ListOfGlobalRenderInformation* class

The **ListOfGlobalRenderInformation** object derives from the **ListOf** class and inherits the core attributes and sub-objects from the **SBase** class. It contains one or more objects of type **GlobalRenderInformation**.

In addition the **ListOfGlobalRenderInformation** object has the optional attributes **versionMajor** and **versionMinor** as well as an optional **DefaultValues** element that provides the default values for the **GlobalRenderInformation** objects contained in the list.

The *versionMajor* attribute

A **ListOfGlobalRenderInformation** has a required attribute **versionMajor** of type **unsigned integer** which specifies the major version of the render information. Major versions do not have to be backwards compatible with any lower major version of the render specification.

The *versionMinor* attribute

A **ListOfGlobalRenderInformation** has a required attribute **versionMinor** of type **unsigned integer** which specifies the minor version of the render information. All minor versions within a major version have to be compatible.

The **versionMajor** and **versionMinor** have been introduced so that applications, that will likely go through quicker iterations than the **SBML** standard can encode that version information there is no relation to the **SBML** attributes **level** and **version**.

3.4.3 The extended *Layout* class

The Render package extends the **Layout** object from the Layout package with the addition of an optional **ListOfLocalRenderInformation** object (Figure 2).

The *ListOfLocalRenderInformation* class

The **ListOfLocalRenderInformation** object derives from the **ListOf** class and inherits the core attributes and sub-objects from the **SBase** class. It contains one or more objects of type **LocalRenderInformation**.

In addition the **ListOfLocalRenderInformation** object has the optional attributes **versionMajor** and **versionMinor** as well as an optional **DefaultValues** element that provides the default values for the **LocalRenderInformation** objects contained in the list.

The `versionMajor` attribute

A [ListOfLocalRenderInformation](#) has a required attribute `versionMajor` of type `unsigned integer` which specifies the major version of the render information. Major versions do not have to be backwards compatible with any lower major version of the render specification.

The `versionMinor` attribute

A [ListOfLocalRenderInformation](#) has a required attribute `versionMinor` of type `unsigned integer` which specifies the minor version of the render information. All minor versions within a major version have to be compatible.

The `versionMajor` and `versionMinor` have been introduced so that applications, that will likely go through quicker iterations than the **SBML** standard can encode that version information there is no relation to the **SBML** attributes `level` and `version`.

3.5 Render Information

The render information classes hold all information about the rendering. The information is stored between three classes: [RenderInformationBase](#), the base class with common features; [GlobalRenderInformation](#) a class applying to types and roles of elements on a global level; and [LocalRenderInformation](#) that provides additional information that can be applied to individual elements from the Layout package. These classes are illustrated in [Figure 2](#) and [Figure 3](#).

3.5.1 The `RenderInformationBase` class

The [RenderInformationBase](#) class is an abstract class that holds all the information that is common to both local and global render information objects. It derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. In addition the [RenderInformationBase](#) has the required attribute `id` and the optional attributes `name`, `programName`, `programVersion`, `referenceRenderInformation` and `backgroundColor`. Additionally it may contain a [ListOfColorDefinitions](#), [ListOfGradientDefinitions](#) and / or a [ListOfLineEndings](#). These lists are optional, however if present may not be empty. There may only be one of each of those lists.

The `id` attribute

A [RenderInformationBase](#) has a required attribute `id` of type `SId`. This `id` may be used to reference this [RenderInformation](#) object from other elements within the Render package.

The `name` attribute

A [RenderInformationBase](#) has an optional attribute `name` of type `string`. This `name` attribute can be used to give the object a more user friendly identifier.

The `programName` attribute

A [RenderInformationBase](#) has an optional attribute `programName` of type `string` which can be used to store the name of the program that was used to create the render information.

The `programVersion` attribute

A [RenderInformationBase](#) has an optional attribute `programVersion` of type `string` which can be used to store the version number of the program used to create the render information.

The `referenceRenderInformation` attribute

A [RenderInformationBase](#) has an optional attribute `referenceRenderInformation` of type `SIdRef` which can be used to specify the `id` of another local or global render information object that complements the current render information object. A program reading and interpreting the render information can use this information to access another render information object, should the current object contain unsuitable information. In order to avoid

loops, only render information objects that have already been defined may be referenced. Thus a **LocalRenderInformation** object may reference any **GlobalRenderInformation** object but may only reference **LocalRenderInformation** objects that have already been defined with the parent **Layout** object. A **GlobalRenderInformation** object may only reference **GlobalRenderInformation** objects that have already been defined in the parent **ListOfLayouts** object.

The backgroundColor attribute

A **RenderInformationBase** has an optional attribute **backgroundColor** of type **string** which defines the background color for rendering.

The ListOfColorDefinitions class

The **ListOfColorDefinitions** object derives from the **ListOf** class and inherits the core attributes and subobjects from the **SBase** class. It contains one or more objects of type **ColorDefinition** which are used to predefine a set of colors to be referenced by **Styles**.

The ListOfGradientDefinitions class

The **ListOfGradientDefinitions** object derives from the **ListOf** class and inherits the core attributes and subobjects from the **SBase** class. It contains one or more objects of type **GradientBase** which are used to define either **LinearGradient** or **RadialGradient** objects to be used in **Styles**.

The ListOfLineEndings class

The **ListOfLineEndings** object derives from the **ListOf** class and inherits the core attributes and subobjects from the **SBase** class. It contains one or more objects of type **LineEnding** which can be used to define a set of **LineEndings** that can be applied to path objects.

3.5.2 The LocalRenderInformation class

The **RenderInformation** element of type **LocalRenderInformation** is the primary container that holds the render information for a **Layout** instance.

The **LocalRenderInformation** object derives from the **RenderInformationBase** class and thus inherits any attributes and elements that are present on this class. A **LocalRenderInformation** may contain exactly one element named **listOfStyles** of type **ListOfLocalStyles**.

The ListOfLocalStyles class

The **ListOfLocalStyles** object derives from the **ListOf** class and inherits the core attributes and subobjects from the **SBase** class. It is optional but if present has to contain one or more objects of type **LocalStyle**.

3.5.3 The GlobalRenderInformation class

Global render information is specified in a very similar way as local render information. The attributes and elements of **GlobalRenderInformation** objects and **LocalRenderInformation** objects are the same with the exception of the **listOfStyles** element. In the case of a **GlobalRenderInformation** object the **listOfStyles** element is of type **ListOfGlobalStyles**.

It should be noted that another difference between **GlobalRenderInformation** and **LocalRenderInformation** is the fact that **GlobalRenderInformation** objects may only reference ids of other **GlobalRenderInformation** objects in their **referenceRenderInformation** attribute.

The ListOfGlobalStyles class

The **ListOfGlobalStyles** object derives from the **ListOf** class and inherits the core attributes and subobjects from the **SBase** class. It contains one or more objects of type **GlobalStyle**.

The following snippet shows the general outline of a [ListOfGlobalRenderInformation](#) object:

```
<layout:listOfLayouts>
  <render:listOfGlobalRenderInformation>
    <render:renderInformation render:id="FancyRenderer_GlobalDefault"
      render:name="default_global_style"
      render:programName="FancyRenderer"
      render:programVersion="0.1.1">
      <render:listOfColorDefinitions>
        ...
      </render:listOfColorDefinitions>
      <render:listOfGradientDefinitions>
        ...
      </render:listOfGradientDefinitions>
      <render:listOfLineEndings>
        ...
      </render:listOfLineEndings>
      <render:listOfStyles>
        ...
      </render:listOfStyles>
    </render:renderInformation>
  </render:listOfGlobalRenderInformation>
</layout:listOfLayouts>
```

3.6 Styles

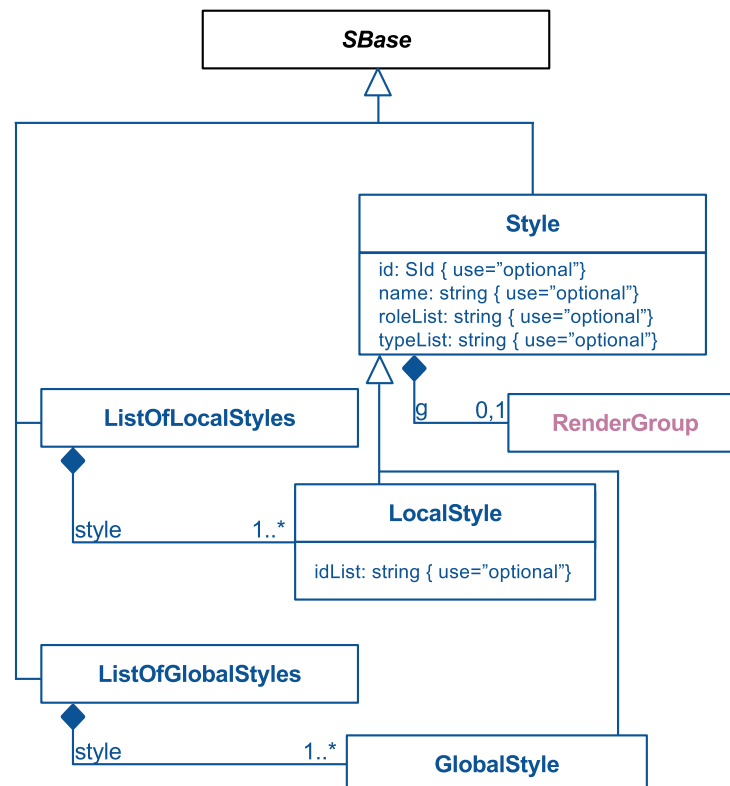


Figure 5: A UML representation of the [Style](#) object for the Render package. See [Section 1.4](#) for conventions related to this figure.

3.6.1 The Style class

Style is an abstract class that holds all the information that is common to both local and global styles (Figure 5). The **Style** object derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. A **Style** element may contain exactly one **RenderGroup** element. In addition the **Style** object has the optional attributes **id**, **name**, **roleList** and **typeList**.

The **RenderGroup** element, “**g**”, is used to specify how the elements covered by this **Style** object are to be rendered and is discussed fully in Appendix 3.10.6.

The id attribute

A **Style** has an optional attribute **id** of type **SIId** which can be used to uniquely identify this **Style** object.

The name attribute

A **Style** has an optional attribute **name** of type **string** which can be used to provide a more user friendly identifier.

The roleList attribute

A **Style** has an optional attribute **roleList** of type **string**. The string value of the **roleList** attribute contains a space separated list of all the roles to which this **Style** should be applied.

This attribute can be used in conjunction with the **objectRole** attribute that is used to extend the **GraphicalObject** class from the Layout package. If the string given as an **objectRole** value appears in the **roleList** attribute of some render information object, then that render information object applies to the graphical object as shown in the snippet below. Note this relationship is only valid if there is no render information object that is more specific. For example another **LocalStyle** could be defined with **idList** that references the **layout:id="go1"** explicitly, in which case that style would be chosen. For more information see also Appendix C.2 on page 71.

```
<layout:layout>
  <layout:listOfAdditionalGraphicalObjects>
    <layout:graphicalObject layout:id="go1" render:objectRole="Parameter">
      ...
    </layout:graphicalObject>
  </layout:listOfAdditionalGraphicalObjects>
  <render:listOfLocalRenderInformation>
    <render:renderInformation render:id="FancyRenderer_GlobalDefault">
      ...
      <render:listOfStyles>
        <render:style render:id="style_1" render:roleList="Parameter">
          <g> ... </g>
        </render:style>
      </render:listOfStyles>
    </render:renderInformation>
  </render:listOfLocalRenderInformation>
</layout:layout>
```

The typeList attribute

A **Style** has an optional attribute **typeList** of type **string**. The string value of the **typeList** attributes contains a space separated list of one or more of the values from the **StyleType** enumeration. The example snippet shows a particular style that is to be applied to both **SpeciesGlyph** and **SpeciesReferenceGlyph** objects from the Layout package.

```

<layout:listOfLayouts>
  <render:listOfGlobalRenderInformation>
    <render:renderInformation render:id="FancyRenderer_GlobalDefault">
      ...
    <render:listOfStyles>
      <render:style render:id="style_1" render:typeList="SPECIESGLYPH_SPECIESREFERENCEGLYPH">
        <g> ... </g>
      </render:style>
    </render:listOfStyles>
  </render:renderInformation>
</render:listOfGlobalRenderInformation>
</layout:listOfLayouts>

```

3.6.2 The GlobalStyle class

The **GlobalStyle** object derives from the **Style** class and thus inherits any attributes and elements that are present on this class. The **GlobalStyle** class is used for objects in the **ListOfGlobalStyles** element of a **GlobalRenderInformation** object.

3.6.3 The LocalStyle class

The **LocalStyle** object derives from the **Style** class and thus inherits any attributes and elements that are present on this class. It is identical to the **GlobalStyle** object but has an additional optional **idList** attribute.

The **LocalStyle** class is used for objects in the **ListOfLocalStyles** element of a **LocalRenderInformation** object.

The idList attribute

A **LocalStyle** has an optional attribute **idList** of type **string** which is a space separated list of ids of layout objects to which this **Style** should be applied.

3.7 Colors and Gradients

All **RenderInformation** objects may contain a **ListOfColorDefinitions** containing objects of type **ColorDefinition** and a **ListOfGradientDefinitions** containing objects of type **GradientBase**. Gradients consist of continuously smooth color transitions along a vector from one color to another, possibly followed by additional transitions along the same vector to other colors. Here the Render package borrows heavily from the SVG specification. These are described in more detail in this section.

3.7.1 The ColorDefinition class

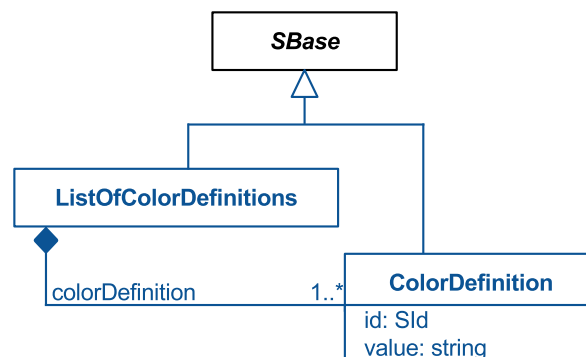


Figure 6: A UML representation of the **ColorDefinition** object for the Render package. See [Section 1.4](#) for conventions related to this figure.

The **ColorDefinition** object derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. In addition the **ColorDefinition** object has the optional attributes **id** and **value**.

The **id** attribute

A **ColorDefinition** has an optional attribute **id** of type **SId** which is used to give the **ColorDefinition** an unique identifier within the **RenderInformation** object.

The **value** attribute

A **ColorDefinition** has an optional attribute **value** of type **string**. Color values are specified as a 6 to 8 digit hex string which defines the RGBA value of the color. If only the first six digits for the RGB value are given, the alpha value (also known as transparency or opacity of the color) is assumed to be 0xFF which means that the color is totally opaque. Instead of specifying a color value, the value “none” can be given which is equivalent to no drawing at all.

The example snippet defines a dark red color, with a red component of 0x20, green component of 0x00, and blue component of 0x00. Since it is not specifying the alpha component, it will have the value of 0xff.

```
<listOfColorDefinitions>
  <colorDefinition id="darkred" value="#200000" />
  ...
</listOfColorDefinitions>
```

3.7.2 The **GradientBase** class

GradientBase is an abstract class that holds all the information that is common to both **RadialGradient** and **LinearGradient** objects (Figure 5). The **GradientBase** object derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. A **GradientBase** may contain one or more **GradientStop** elements. In addition the **GradientBase** object has a required **id** attribute and an optional **spreadMethod** attribute.

The **id** attribute

A **GradientBase** has a required attribute **id** of type **SId** which is used to uniquely identify or reference a gradient within an **RenderInformation** object.

The **spreadMethod** attribute

A **GradientBase** has an optional attribute **spreadMethod** of type **GradientSpreadMethod** that specifies the method that is used to continue the gradient pattern if the vector points do not span the whole bounding box of the object to which the gradient is applied.

3.7.3 The **GradientStop** class

As the name suggests the **GradientStop** object is used to define "gradient stops" which are used in line with the SVG specification. The **GradientStop** object derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. In addition the **GradientStop** object has the required attributes **offset** and **stop-color**.

The **offset** attribute

A **GradientStop** has a required attribute **offset** of type **RelAbsVector** which represents the relative distance from the starting point of the gradient. Depending on the type of gradient, this is either the point defined by the **x1,y1** and **z1** attributes (**LinearGradient**) or the **fx, fy** and **fz** attributes (**RadialGradient**). This value is given as a positive percentage value.

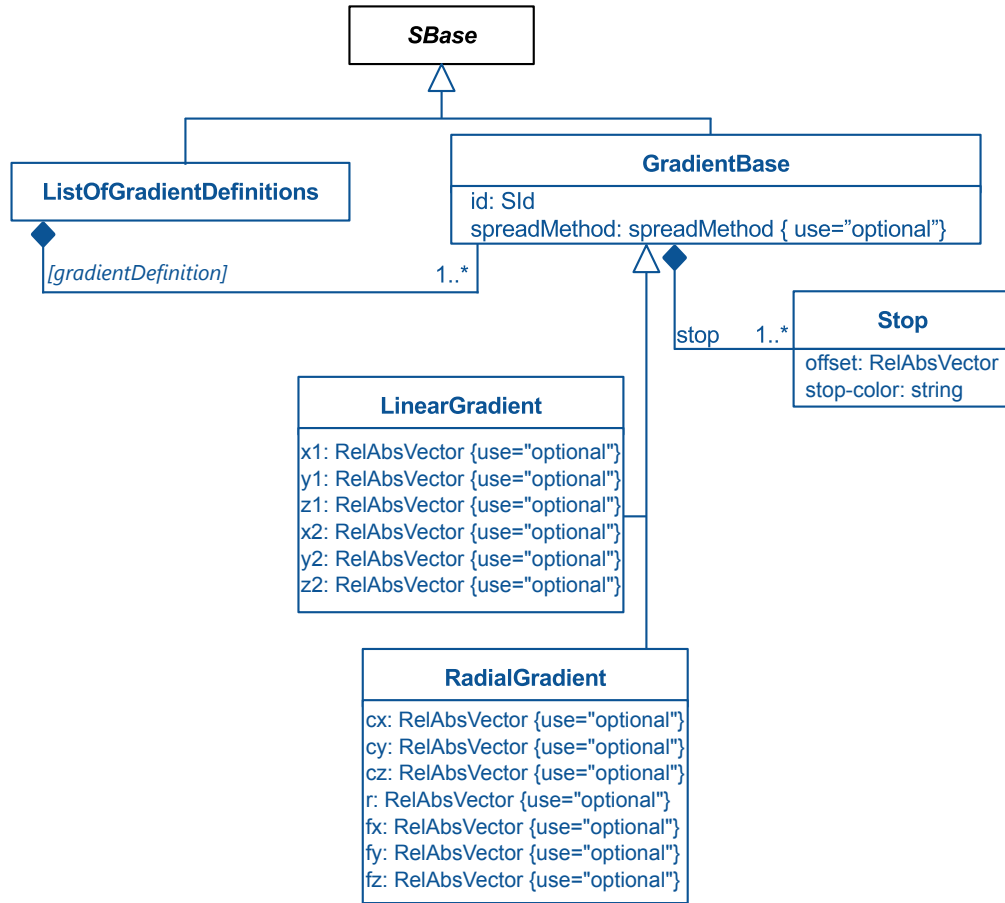


Figure 7: A UML representation of the gradient objects for the Render package. See [Section 1.4](#) for conventions related to this figure.

The stop-color attribute

A **GradientStop** has a required attribute **stop-color** of type **string** which defines the color for the given gradient stop. The attribute's value can either be given as a hexadecimal color value or as the id of a **ColorDefinition** object from the **ListOfColorDefinitions** unless that **ColorDefinition** specifies “none”. To specify the id of another gradient as the value of a **stop-color** attribute is considered an error. In case the two points that define the gradient vector are identical, the area is to be painted with a single color taken from the last gradient stop element.

There are a few rules that need to be considered when working with gradient stops. Basically these rules are the same as defined by the SVG specification.

1. The offset value of a gradient stop should be between 0% and 100%. If the offset lies outside of this value, the value is adjusted to be either 0% if the given value is smaller than 0% or to 100% if the value is greater than 100%.
2. The absolute part in any offset value is ignored, meaning it is considered to be 0.0 even if specified otherwise in a gradient stop.
3. The offset of any gradient stop has to be greater or equal to the offset of the preceding gradient stop. If a gradient stop has an offset that is smaller than the offset of the preceding stop, the offset is considered to have the same value as the offset of the preceding stop.

4. If two gradient stops have the same offset value, the last gradient stop with this offset value determines the color at this point in the gradient.

3.7.4 The *LinearGradient* class

The **LinearGradient** provides the vector points that define the start and end points to which the **GradientStop** elements should be mapped.

The **LinearGradient** object derives from the **GradientBase** class and thus inherits any attributes and elements that are present on this class. In addition the **LinearGradient** object has the attributes **x1**, **y1**, **z1**, **x2**, **y2** and **z2**. As the names suggest these represent the x, y and z coordinates in a three dimensional Cartesian system. If only the x and y attributes are used a two dimensional viewport is assumed.

Since the value for the vector can be specified as an absolute value or one that is relative to the current viewport these attributes all have values of type **RelAbsVector**.

The **x1**, **y1** and **z1** attributes

The attributes **x1**, **y1** and **tokenz1** define the start point of the gradient in either two (**z1** undefined) or three dimensions.

The **x2**, **y2** and **z2** attributes

The attributes **x2**, **y2** and **tokenz2** define the end point of the gradient in either two (**z2** undefined) or three dimensions.

Example of specifying the **LinearGradient** shown in Figure 8:

```
<listOfGradientDefinitions>
  <linearGradient x1="12.5%" y1="25%" x2="87.5%" y2="75%">
    <stop offset="5%" stop-color="#000F60" />
    <stop offset="95%" stop-color="#000FF6" />
  </linearGradient>
  ...
</listOfGradientDefinitions>
```



Figure 8: Example of a **LinearGradient**

3.7.5 The *RadialGradient* class

The **RadialGradient** object derives from the **GradientBase** class and thus inherits any attributes and elements that are present on this class. In addition the **RadialGradient** object has seven attributes (each of type **RelAbsVector**) that are used to define the center, radius and focal point of the gradient.

The **cx**, **cy** and **cz** attributes

The attributes **cx**, **cy** and **cz** define the center of the gradient as a point in either two (**cz** undefined) or three dimensions.

The *r* attribute

The attribute *r* defines the radius of the gradient and should be positive. If the radius is given in relative values, the relation is to the width as well as the height. This means that if the width of the bounding box and the height of the bounding box are not equal, *cx*, *cy*, *cz* and *r* don't actually specify a circle, but an ellipse.

The *fx*, *fy* and *fz* attributes

The attributes *fx*, *fy* and *fz* define the focal point of the gradient as a point in either two (*fz* undefined) or three dimensions. The gradient is drawn such that this point is mapped to the 0% [GradientStop](#). If one of these attributes is left undeclared it is considered to be equal to the corresponding coordinate of the center point. If the focal point lies outside the circle, the focal point is considered to be located on the intersection of the line from the center point to the focal point and the sphere determined by the center point and the radius.

Example of specifying the [RadialGradient](#) shown in [Figure 9](#):

```
<listOfGradientDefinitions>
  <radialGradient cx="50%" cy="50%" r="300" fx="50%" fy="50%">
    <stop offset="0%" stop-color="#FF0000" />
    <stop offset="50%" stop-color="#0000FF" />
    <stop offset="100%" stop-color="#FF0000" />
  </radialGradient>
  ...
</listOfGradientDefinitions>
```

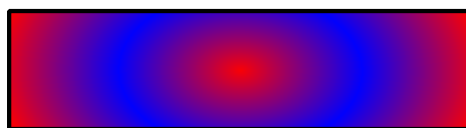


Figure 9: Example of a [RadialGradient](#)

3.8 Transformation

In order to be able to display text that is not aligned horizontally or vertically or to effectively compose groups of objects from primitives, transformations like rotation, translation and scaling are needed. SVG, among other options, allows the user to specify a 3x3 matrix transformation matrix:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Since the last row of the matrix is always 0 0 1, the matrix is specified as a six value vector. In the render extension each group or graphical primitive is derived from the class [Transformation2D](#) and can have a **transform** attribute just as in SVG.

3.8.1 The Transformation class

The [Transformation](#) class is a common base class for all elements that can be drawn. Since both the Layout package and the Render package are currently limited to two dimensions, this class is only used as a base class for [Transformation2D](#) and we leave the complete specification of this class for a future version of this document.

The [Transformation](#) object derives from the *SBase* class and thus inherits any attributes and elements that are

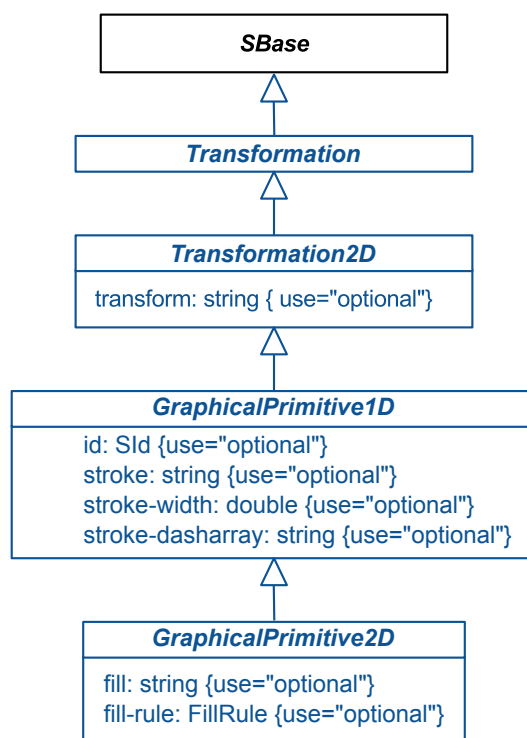


Figure 10: A UML representation of the base graphical primitive classes for the Render package. See [Section 1.4](#) for conventions related to this figure.

present on this class. In addition the [Transformation](#) object has the required **transform** attribute.

The **transform** attribute

A [Transformation](#) has a required attribute **transform** consisting of an array of type **double**. This specifies an affine transformation matrix in three dimensions in which case the array must consist of exactly 12 values.

3.8.2 The [Transformation2D](#) class

Since the current render information specification only defines two dimensional objects, we derive a second class called [Transformation2D](#) from [Transformation](#). As illustrated in [Figure 10](#) the class [Transformation2D](#) serves as the base class for all drawable 1D and 2D objects.

The **transform** attribute

The [Transformation2D](#) class restricts the transformation matrix to specify the six values of a 2D affine transformation. Thus the **transform** attribute consists of an array of exactly 6 values of type **double**. Thus the allowed value for the attribute has the form: **a**, **b**, **c**, **d**, **e**, **f**.

The values for **a**, **b**, **c**, **d**, **e** and **f** depend on the transformation operation components and the order in which those transformation components are executed.

There are four basic transformation operations that can be combined in a affine transformation matrix. Details of these are given in [Appendix B on page 65](#).

All objects that are derived from [Transformation2D](#) can have a transformation, this includes group elements. In contrast to other attributes on groups and children of groups, the transformation is not overwritten if it is specified in a child, but rather all transformations that are defined in an object hierarchy accumulate. Thus when a group specifies a transformation and a child of the group also sets a transformation, the transformation for the child has

to be applied to the child only and the transformation that is set on the group has to be applied to the whole group, i.e. to all children of the group.

3.9 GraphicalPrimitives

The graphical primitives polygons, rectangles and ellipses are based on the corresponding elements from SVG. For lines, arcs and general path primitives, we introduce the **RenderCurve** element which differs slightly from the Layout package **Curve**. Whereas **Point** objects in the Layout package could only contain absolute values for their coordinates, **RenderPoint** objects in the Render package can contain relative coordinate values. Two primitive abstract classes are defined to specify the common properties of 1D and 2D shapes.

3.9.1 The GraphicalPrimitive1D class

The **GraphicalPrimitive1D** object derives from the **Transformation2D** class and thus inherits any attributes and elements that are present on this class (Figure 10). In addition the **GraphicalPrimitive1D** object has the optional **id**, **stroke**, **stroke-width** and **stroke-dasharray** attributes.

The id attribute

A **GraphicalPrimitive1D** has an optional attribute **id** of type **SId** which can be used to uniquely identify the object.

The stroke attribute

A **GraphicalPrimitive1D** has an optional attribute **stroke** of type **string**. This is used to specify the color of the stroke that is used to draw the curve or the outline of geometric shapes. This **stroke** attribute can either hold a color value or it can hold the id of a predefined **ColorDefinition** object.

The stroke-width attribute

A **GraphicalPrimitive1D** has an optional attribute **stroke-width** of type **double** which specifies the width of the stroke to be used.

The stroke-dasharray attribute

A **GraphicalPrimitive1D** has an optional attribute **stroke-dasharray** consisting of an array of values of type **unsigned integer**. This list specifies the lengths of dashes and gaps that are used to draw the line. The individual numbers in the list are separated by commas. For example, "5,10" would mean to draw 5 points, make a 10 point gap, draw 5 points etc. If the pattern is to start with a gap, the first number has to be 0.

It should be noted that if a style defines a stroke dasharray and this style is applied to a **Curve** from the Layout package, one has to watch out for the fact that the layout curves may contain breaks (if the end point of segment *n* is not identical to the starting point of segment *n+1*). In this case each of the unbroken line stretches is considered a separate curve object and the line stippling is applied to each curve. That means the line stippling is not continuously applied through the gap, but it starts again after the gap.

3.9.2 The GraphicalPrimitive2D class

The **GraphicalPrimitive2D** object derives from the **GraphicalPrimitive1D** class and thus inherits any attributes and elements that are present on this class (Figure 10). In addition the **GraphicalPrimitive2D** object has the optional **fill** and **fill-rule** attributes.

The fill attribute

A **GraphicalPrimitive2D** has an optional attribute **fill** of type **string** which specifies the fill style of the object. The fill style can either be a hexadecimal color value, the id of a **ColorDefinition** object or the id of a **GradientBase** object. Instead of a color or gradient id, "none" can be specified which means that the object is unfilled.

The fill-rule attribute

A **GraphicalPrimitive2D** has an optional attribute **fill-rule** of type **FillRule** that can be used to specify how the shape should be filled.

Currently the **fill-rule** attribute is only useful for polygons. No other shapes have alternating areas.

3.9.3 The RenderCurve class

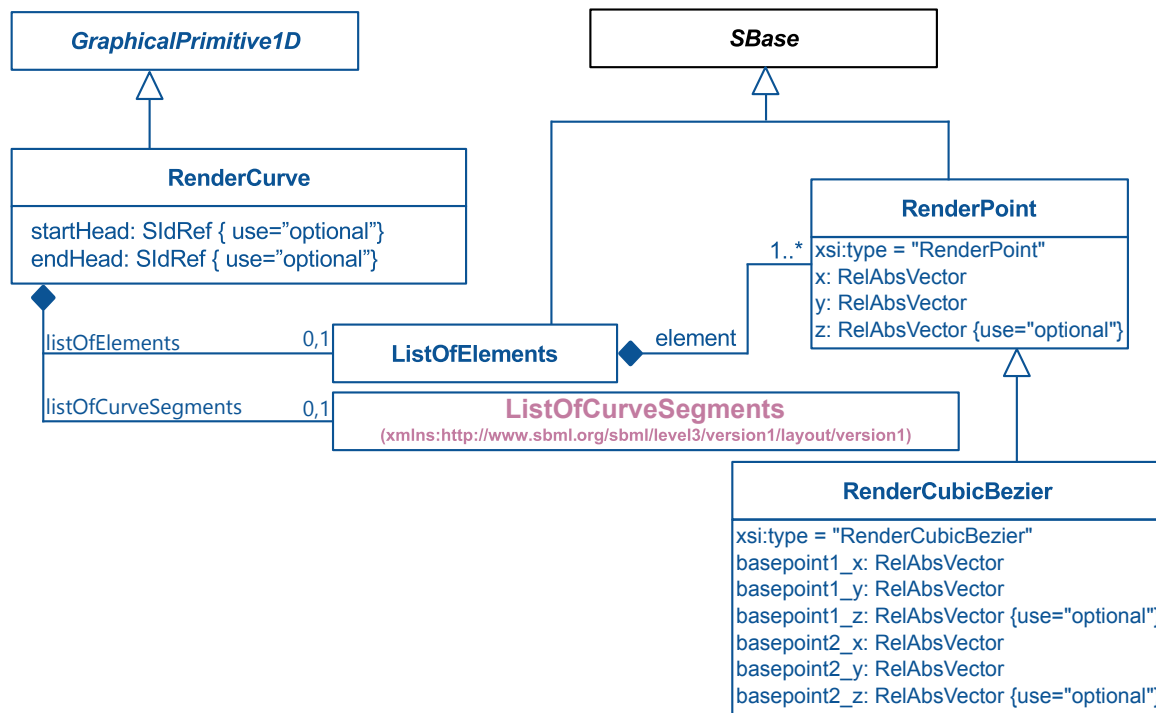


Figure 11: A UML representation of the **RenderCurve** classes for the **Render** package. See [Section 1.4](#) for conventions related to this figure.

Simple lines and complex curves are represented by a **RenderCurve** element.

The **RenderCurve** object derives from the **GraphicalPrimitive1D** class (see [Figure 11](#)) and thus inherits any attributes and elements that are present on this class. A **RenderCurve** contains at most one **ListOfElements** element and at most one **ListOfCurveSegments** from the **Layout** package. In addition the **RenderCurve** object has the optional attributes **startHead** and **endHead**.

The startHead attribute

A **RenderCurve** has an optional attribute **startHead** of type **SIdRef** and points to the **LineEnding** that should be applied to the start of the path.

The endHead attribute

A **RenderCurve** has an optional attribute **endHead** of type **SIdRef** and points to the **LineEnding** that should be applied to the end of the path.

The ListOfElements class

The **ListOfElements** object derives from the **ListOf** and inherits the core attributes and subobjects from the **SBase** class. It contains one or more objects of type **RenderPoint** or of the derived type **RenderCubicBezier**. The only restriction is that the first element must be a **RenderPoint**.

Thus the first point specifies the start point of the curve. If the next element is another **RenderPoint**, we have a straight line segment, going from the start point to the second point. Should the second point be a **RenderCubicBezier** a cubic bezier curve will be added from the start point with its values. Thus the **ListOfElements** holds a concise definition of the curve specifying start and end points for all line segments.

The ListOfCurveSegments

The Layout package defines a similar **Curve** that has identical specification except it is restricted to using absolute values. The classes involved have thus been redefined for the Render package which facilitates the use of relative values. However it is perfectly valid to use the **ListOfLineSegments** object from the Layout package either in place of the **ListOfElements** or in addition to it.

The example in Section 3.9.5 illustrates both the **ListOfElements** and **ListOfCurveSegments** objects.

3.9.4 The RenderPoint class

RenderPoint objects are used to specify the individual curve segments.

The **RenderPoint** object derives from the **SBase** class and thus inherits any attributes and elements that are present on this class. In addition the **RenderPoint** object has the required attributes **x** and **y** and the optional attribute **z**. It also has the required attribute **type** from the “**xsi**” namespaces.

The x, y and z attributes

These three attributes are used to specify the coordinates of a **RenderPoint** in two (missing **z**) or three dimensions. They are of type **RelAbsVector** and can thus specify a coordinate as either an absolute or relative value. The coordinate values are always with respect to the bounding box of the layout object to which the render information applies.

The xsi:type attribute

For a **RenderPoint** object this attribute will always have the value “**RenderPoint**”.

3.9.5 The RenderCubicBezier class

The **RenderCubicBezier** object derives from the **RenderPoint** class and thus inherits any attributes and elements that are present on this class. In addition the **RenderCubicBezier** object has the required attributes **basepoint1_x**, **basepoint1_y**, **basepoint2_x** and **basepoint2_y**. It also has the optional attributes **basepoint1_z** and **basepoint2_z**.

The basepoint1_x, basepoint1_y and basepoint1_z attributes

These three attributes are used to specify the coordinates of the first basepoint of a **RenderCubicBezier** in two (missing **basepoint1_z**) or three dimensions. They are of type **RelAbsVector** and can thus specify a coordinate as either an absolute or relative value. The coordinate values are always with respect to the bounding box of the layout object to which the render information applies.

The basepoint2_x, basepoint2_y and basepoint2_z attributes

These three attributes are used to specify the coordinates of the second basepoint of a **RenderCubicBezier** in two (missing **basepoint2_z**) or three dimensions. They are of type **RelAbsVector** and can thus specify a coordinate as either an absolute or relative value. The coordinate values are always with respect to the bounding box of the

layout object to which the render information applies.

The `xsi:type` attribute

For a [RenderCubicBezier](#) object this attribute will always have the value “RenderCubicBezier”.

The example snippet illustrates the definition of a [RenderCurve](#) with two line segments that are to be painted using a black stroke with width 2.0. The first line segment is a straight segment going from the objects left middle (0%, 50%) to the right middle(100%, 50%). The second segment represents a cubic bezier, that continues from the right middle(100%, 50%) back to the left middle(0%, 50%) with two control points at (50%, 90%). The equivalent curve defined using the [ListOfLineSegments](#) from the Layout package is also included (assuming a 100x100 square object).

```
<render:g ...>

  <!-- the curve is defined in the render namespace -->
  <render:curve render:stroke-width="2.0" render:stroke="#000000" >

    <!-- using the listOfElements from the render namespace -->
    <render:listOfElements>
      <!-- define the first point -->
      <render:element xsi:type="RenderPoint" render:x="0%" render:y="50%" />

      <!-- The next item starts at the previous point -->
      <!-- It is also a point so draw a straight line from the start point to here -->
      <render:element xsi:type="RenderPoint" render:x="100%" render:y="50%" />

      <!-- The next item starts at the previous point -->
      <!-- It is a cubic bezier so draw a curve using the basepoints from the start point to here -->
      <render:element xsi:type="RenderCubicBezier" render:x="0%" render:y="50%"
        render:basepoint1_x="50%" render:basepoint1_y="90%"
        render:basepoint2_x="50%" render:basepoint2_y="90%" />
    </render:listOfElements>

    <!-- using the listOfCurveSegments from the layout namespace -->
    <layout:listOfCurveSegments>

      <!-- the first segment is a line from start to end point -->
      <layout:curveSegment xsi:type="LineSegment">
        <layout:start layout:x="0" layout:y="50" />
        <layout:end layout:x="100" layout:y="50"/>
      </layout:curveSegment>

      <!-- the second segment is a curve from start to end with given basepoints -->
      <layout:curveSegment xsi:type="CubicBezier">
        <layout:start layout:x="100" layout:y="50" />
        <layout:end layout:x="0" layout:y="50"/>
        <layout:basePoint1 layout:x="50" layout:y="90"/>
        <layout:basePoint2 layout:x="50" layout:y="90"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>

  </render:curve>
  ...
</render:g>
```

3.10 Geometric Shapes

This section details the classes of geometric objects that can be defined using the transformations and graphical primitives described (see [Figure 12](#)).

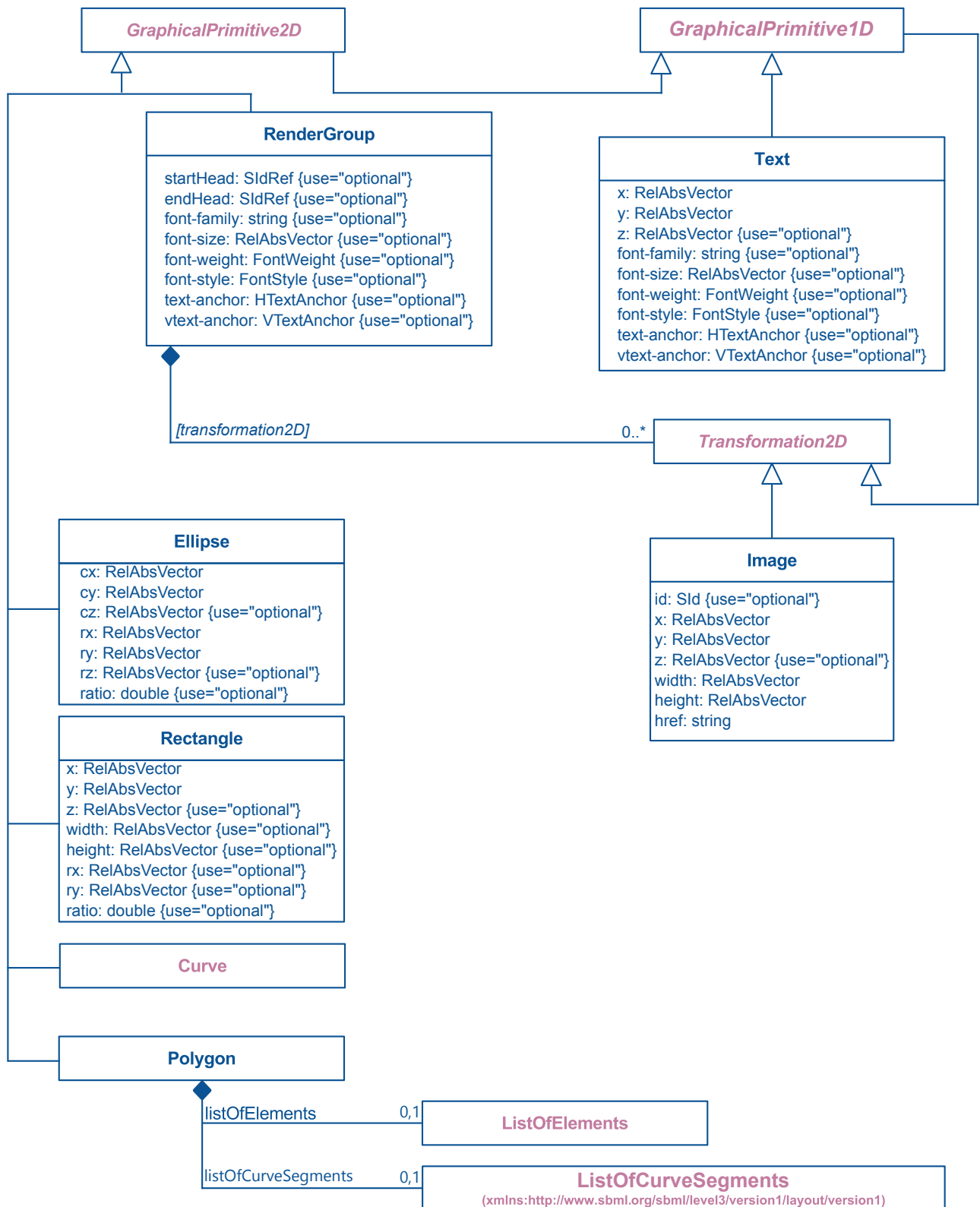


Figure 12: A UML representation of the graphical primitive classes for the Render package. See [Section 1.4](#) for conventions related to this figure.

3.10.1 The Polygon class

A **Polygon** object is made up of a **polygon** element which contains at most one **ListOfElements** and/or one **ListOfCurveSegments** used to define the edges of the polygon.

The major difference to the **RenderCurve** object is that the object is always closed. That is the last point of the curve is connected to the first. Therefore, the polygon can have a fill style that determines how the inside of the polygon is to be rendered.

The example snippet shows the render specification of a **Polygon** and of an unclosed path. It uses a black pen with width 3, and a red fill brush. [Figure 13](#) illustrates these shapes (without the red fill!).

```
<g ...>
  <!-- define a path with three points -->
  <curve stroke="#000000" stroke-width="3">
    <listOfElements>
      <element xsi:type="RenderPoint" x="0%" y="0%"/>
      <element xsi:type="RenderPoint" x="100%" y="0%"/>
      <element xsi:type="RenderPoint" x="0%" y="100%"/>
    </listOfElements>
  </curve>

  <!-- the same points defined as a polygon
        so the last point draws a line to the first point -->
  <polygon stroke="#000000" stroke-width="3" fill="#FF0000">
    <listOfElements>
      <element xsi:type="RenderPoint" x="0%" y="0%"/>
      <element xsi:type="RenderPoint" x="100%" y="0%"/>
      <element xsi:type="RenderPoint" x="0%" y="100%"/>
    </listOfElements>
  </polygon>
</g>
```

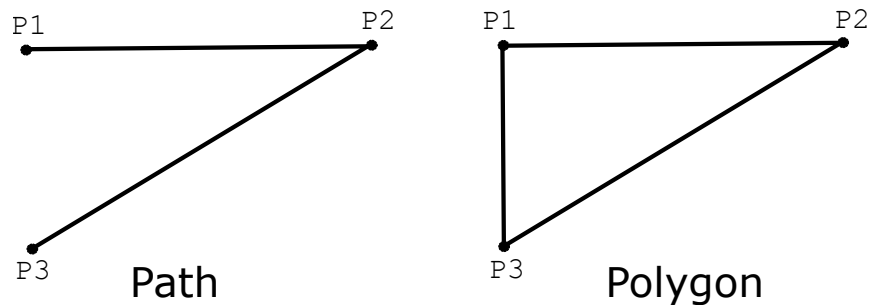


Figure 13: Rendering of a Path vs. rendering of a Polygon with the same base points

3.10.2 The Rectangle class

The **Rectangle** object was taken from the SVG specification and allows the definition of rectangles with or without rounded edges.

The **Rectangle** object derives from the **GraphicalPrimitive2D** class and thus inherits any attributes and elements that are present on this class. In addition the **Rectangle** object has the required attributes **x**, **y**, **height**, and **width** and the optional attributes **z**, **rx**, **ry** and **ratio**.

The **x**, **y** and **z** attributes

These attributes are of type **RelAbsVector** and specify its position within the bounding box of the enclosing layout object.

The width and height attribute

These attributes are of type [RelAbsVector](#) and specify the width and height of the rectangle, either in absolute values or as a percentage of the width and height of the enclosing bounding box.

The rx and ry attributes

These attributes are of type [RelAbsVector](#) and specify the radius of the corner curvature. If only **rx** or **ry** is specified, the other is presumed to have the same value as the one given. If no values are supplied this means that the edges are not rounded. The relative values in **rx** and **ry** are in relation to the width and the height of the rectangle respectively. So a value of 10% for **rx** means the radius of the corner is 10% of the width of the rectangle.

The ratio attribute

If the optional **ratio** attribute of **double** is set, the biggest rectangle with the desired ratio of width to height is to be drawn centered in the objects bounding box. Using this approach makes it possible to always encode a square (by specifying **ratio="1"**), even if used with relative radii and a rectangular bounding box.

3.10.3 The Ellipse class

The [Ellipse](#) object derives from the [GraphicalPrimitive2D](#) class and thus inherits any attributes and elements that are present on this class. In addition the [Ellipse](#) object has the required attributes **cx**, **cy** and **rx** and the optional attributes **ry**, **cz** and **ratio**.

The cx, cy and cz attributes

These attributes are of type [RelAbsVector](#) and specify the center of the ellipse.

The rx and ry attribute

These attributes are of type [RelAbsVector](#) and specify the radius of the ellipse along the x-axis and y-axis respectively. If only one value is specified the other is assumed to have the same value.

Circles are a special case where the **rx** and **ry** attributes have the same value. However, a circle will only be encoded if either the radii are specified absolutely, or if the bounding box is square. To encode circles for arbitrary bounding boxes and relative positioning please see the **ratio** attribute below.

The ratio attribute

If the optional **ratio** attribute of **double** is set, the biggest ellipse with the desired ratio of width to height is to be drawn centered in the objects bounding box. Using this approach makes it possible to always encode a circle (by specifying **ratio="1"**), even if used with relative radii and a rectangular bounding box.

3.10.4 The Text class

In order to draw text, we use the **text** element from SVG with slight modifications. For reasons of simplicity, we limit the display of text to normal text, outlined or filled-outlined text are not supported.

Since we have a right handed coordinate system, the positive y axis normally faces downward on the screen if the positive z-axis goes into the screen. This means that text actually has to be rendered with the top towards lower y-values.

The [Text](#) object derives from the [GraphicalPrimitive1D](#) class and thus inherits any attributes and elements that are present on this class. In addition the [Text](#) object has the required attributes **x** and **y** and the optional attributes **z**, **font-size**, **font-family**, **font-weight**, **font-style**, **text-anchor** and **vtext-anchor**.

The x attribute

The **x** attribute is of type [RelAbsVector](#) and specifies the position of the horizontal text anchor.

The y attribute

The **y** attribute is of type [RelAbsVector](#) and specifies the position of the vertical text anchor.

The z attribute

The **z** attribute is of type [RelAbsVector](#) and directly specifies the depth value of the text element since there is no alignment attribute for text in the third dimension.

The font-size attribute

A [Text](#) has an optional attribute **font-size** of type [RelAbsVector](#) which must have a positive value. In the case of a relative value it specifies a percentage of the height of the corresponding object. Combinations of relative and absolute values are not allowed.

The font-family attribute

A [Text](#) has an optional attribute **font-family** of type [string](#) that allows to specify the font or font-family to be used for the text element. For maximum interoperability the font families specified in [FontFamily](#) have to be supported at a minimum. Those are the generic families “[serif](#)”, “[sans-serif](#)” and “[monospace](#)”.

The font-weight attribute

A [Text](#) has an optional attribute **font-weight** of type [FontWeight](#) and specifies if the text is to be “[normal](#)” or “[bold](#)”.

The font-style attribute

A [Text](#) has an optional attribute **font-style** of type [FontStyle](#) which specifies whether the style for the text is to be “[italic](#)” or “[normal](#)”.

The text-anchor attribute

A [Text](#) has an optional attribute **text-anchor** of type [HTextAnchor](#) which specifies the horizontal alignment of the text (see [Appendix A on page 61](#)).

The vtext-anchor attribute

A [Text](#) has an optional attribute **vtext-anchor** of type [VTextAnchor](#) which specifies the vertical alignment of the text (see [Appendix A on page 61](#)).

Note that since the way text is drawn is completely determined by the font specification, text elements should ignore the stroke-width attribute that they inherit from [GraphicalPrimitive1D](#).

3.10.5 The Image class

To include bitmaps into a graphical representation we use the [Image](#) element from SVG. However the use of the [Image](#) element to include complete SVG vector images has been excluded.

The [Image](#) object derives from the [Transformation2D](#) class and thus inherits any attributes and elements that are present on this class. In addition the [Image](#) object has the optional attributes **id** and **z** and the attributes **x**, **y**, **width**, **height** and **href** that are required..

The id attribute

An [Image](#) has an optional attribute **id** of type [SId](#) that can be used to give the [Image](#) a unique identifier.

The x, y and z attributes

These attributes are of type [RelAbsVector](#) and specify the position of the [Image](#) within it's bounding box.

The **width** and **height** attributes

These attributes are of type **RelAbsVector** and specify the width and height to be used for the **Image**. These attributes are both required.

The **href** attribute

An **Image** has a required attribute **href** of type **string** which encodes a reference to an external JPEG or PNG file. The reference must be an absolute or relative path to a local file. Non-local image resources (e.g. from the net) are currently not supported.

Note that if the referenced image is larger than the given width and height, it has to be scaled to the given dimensions. If the referenced resource can not be found, it is up to the application if nothing is drawn or some placeholder is displayed. Preferably the user would get some kind of notification about the missing resource.

The example shows the encoding for including the file `Glucose.pdf`.

```
<g ...>
  <image x="10%" y="10%" width="80" height="100" href="Glucose.pdf" />
  ...
</g>
```

3.10.6 The **RenderGroup** class

Similar to the technique used by SVG, several graphical primitives can be grouped inside a **g** element to generate more complex render information.

The **RenderGroup** object derives from the **GraphicalPrimitive2D** class and thus inherits any attributes and elements that are present on this class. A **RenderGroup** contains one or more child elements that can be any class derived from the **Transformation2D** class. In addition the **RenderGroup** object has the following attributes.

The **startHead** and **endHead** attributes

A **RenderGroup** has optional attributes **startHead** and **endHead** of type **SIIDRef** which point to a **LineEnding** for the start and end of curves respectively. These attributes only apply to **RenderCurve** objects from the layout, not to **RenderCurve** objects within the group/needs that not a contradiction. Since those two attributes only make sense on the outermost group of a style, they are to be ignored on all other groups.

The **font-size**, **font-family**, **font-weight**, **font-style**, **text-anchor** and **text-anchor** attributes

These attributes are of the same types as the identically named attributes specified on the **Text** object. If any of those attributes is specified for a **RenderGroup** object, it specifies the corresponding attribute for all graphical primitives and groups defined within this group. If a graphical primitive or a group redefines one or more of those attributes, the newly defined values take effect.

3.11 The **LineEnding** class

In many graphs the relations between nodes are depicted by lines and often the type of relation is encoded in the line ending. For this reason, the render extension provides ways to specify a set of arbitrary line endings and means to apply those to other objects. More information is provided in [Appendix C.1](#).

The **LineEnding** object derives from the **GraphicalPrimitive2D** class and thus inherits any attributes and elements that are present on this class. A **LineEnding** contains exactly one **BoundingBox** element from the Layout package which allows the **position** and **dimensions** to be specified. It also contains a **RenderGroup** element which provides the necessary render information for the line ending.

In addition the **LineEnding** object has the a required **id** attribute and an optional **enableRotationalMapping** attribute.

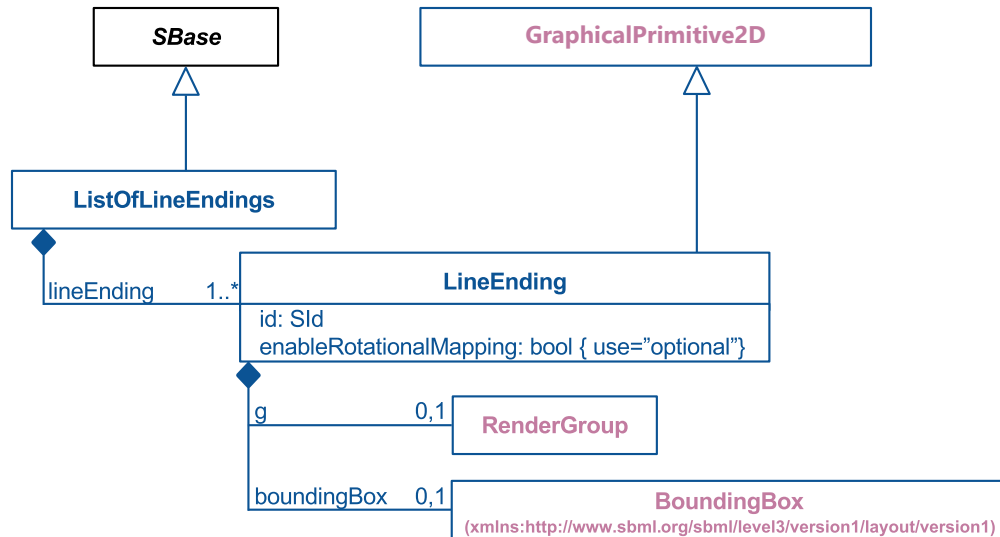


Figure 14: A UML representation of the **LineEnding** class for the **Render** package. See [Section 1.4](#) for conventions related to this figure.

The **id** attribute

A **LineEnding** has a required attribute **id** of type **SId** which allows a unique identifier to be provided for this **LineEnding** so that it may be referenced by other objects. The **startHead** and **endHead** attributes on a **RenderCurve** expect to point to the **id** of a **LineEnding**.

The **enableRotationalMapping** attribute

A **LineEnding** has an optional attribute **enableRotationalMapping** of type **boolean** which specifies whether a line ending will be rotated depending on the slope of the line it is applied to (if “**true**”) or if it is drawn just the way it was specified (if “**false**”).

It should be noted that the top level **RenderGroup** in a **LineEnding** differs from top level groups in normal graphical elements in one respect; that is, the top level **RenderGroup** of a **LineEnding** inherits all attributes from the line it is applied to except for the attributes for the line endings themselves. This way a style sheet can define one line ending which can be applied to lines of different colors and it inherits the color from the line. If the group also inherited the attributes for the line endings and it contained a **curve** element itself, we would have generated an endless loop.

The example snippet shows the definition of an arrow head.

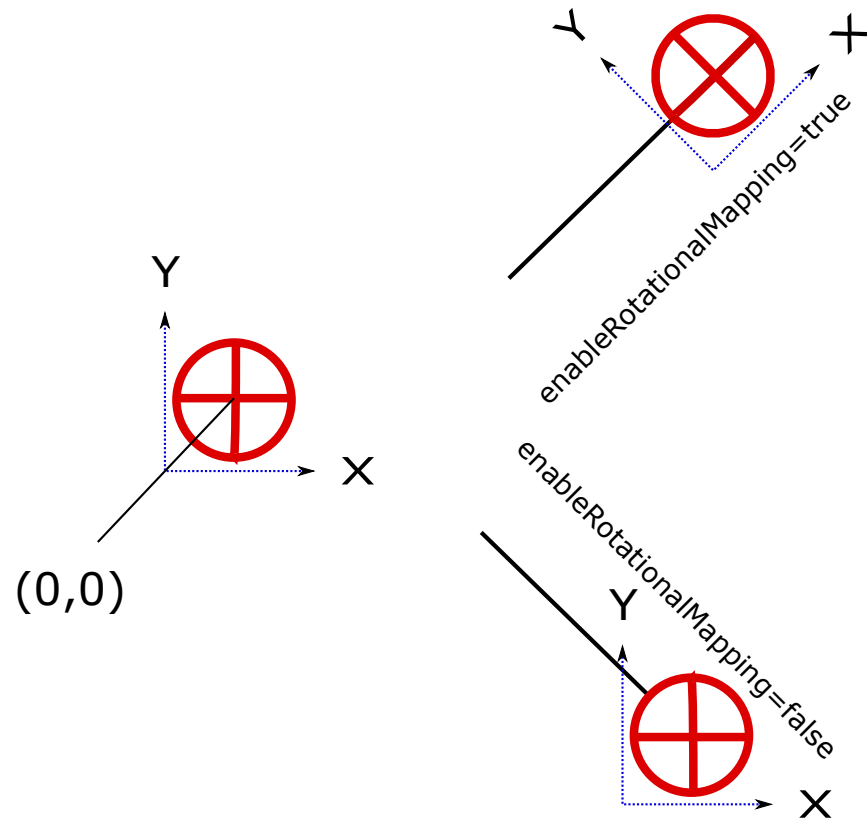


Figure 15: example of a line ending with and without rotation mapping enabled

```

<lineEnding id="SimpleArrowHead">
  <boundingBox>
    <position x="-10.0" y="-4.0" />
    <dimensions width="12.0" height="8.0"/>
  </boundingBox>
  <g>
    <polygon>
      <curve>
        <listOfCurveSegments>
          <curveSegment xsi:type="LineSegment">
            <start x="100%" y="50%" />
            <end x="0%" y="100%" />
          </curveSegment>
          <curveSegment xsi:type="LineSegment">
            <start x="0%" y="100%" />
            <end x="0%" y="0%" />
          </curveSegment>
        </listOfCurveSegments>
      </curve>
    </polygon>
  </g>
</lineEnding>

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

4 Illustrative examples of the Render syntax

This is an example on how an SBGN document could be represented using the SBML layout and render extensions. The example represented once as an SBML Level 2 Version 1 document using annotations and once as an SBML Level 3 Version 1 document with the layout and render extension packages.

The example contains only one simple layout and three global as well as one local style. Although this example does not show all features of the render extension, it should give a good overview on how the layout and the render extension are used together.

The following four figures are generated from this example using the xslt style sheet implementation with xsltproc. The SVG images were then rendered with the Chrome Browser from Google.

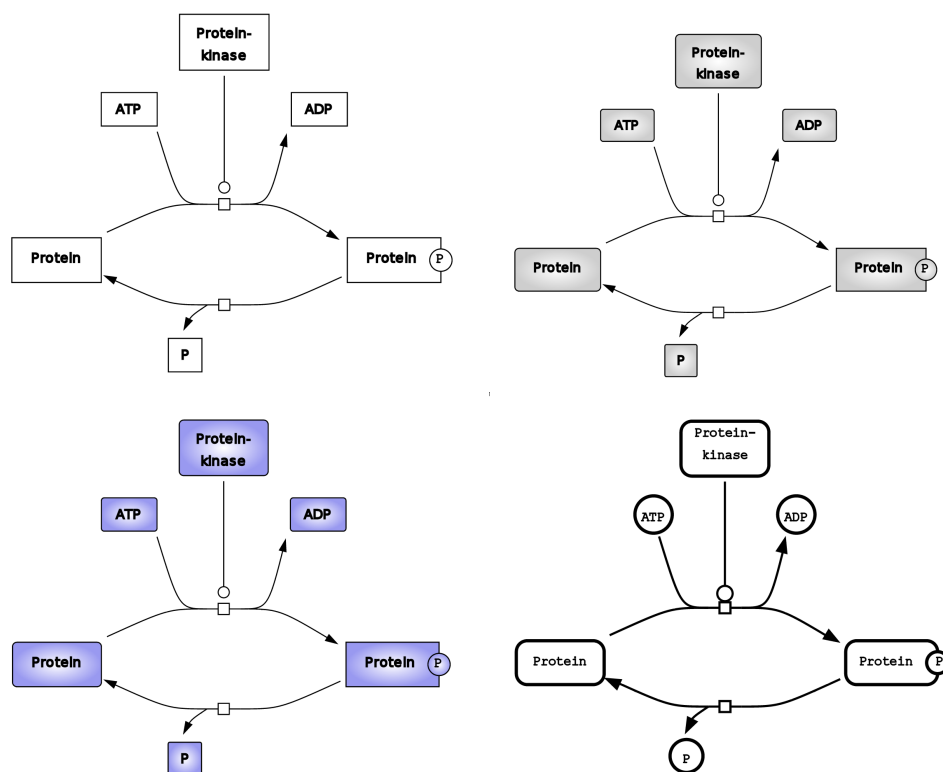


Figure 16: example converted to SVG and rendered with Google Chrome browser

SBML Level 3 Version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
  xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
  xmlns:render="http://www.sbml.org/sbml/level3/version1/render/version1"
  level="3" version="1" layout:required="false" render:required="false">
  <model id="ProteinPhosphorylation" substanceUnits="substance"
    timeUnits="second" volumeUnits="volume" areaUnits="area"
    lengthUnits="metre" extentUnits="substance">
    <listOfUnitDefinitions>
      <unitDefinition id="volume">
```

```

    <listOfUnits>
      <unit kind="litre" exponent="1" scale="0" multiplier="1"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="substance">
    <listOfUnits>
      <unit kind="mole" exponent="1" scale="0" multiplier="1"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="area">
    <listOfUnits>
      <unit kind="metre" exponent="2" scale="0" multiplier="1"/>
    </listOfUnits>
  </unitDefinition>
</listOfUnitDefinitions>
<listOfCompartments>
  <compartment id="Cell" spatialDimensions="3" units="volume" constant="true"/>
</listOfCompartments>
<listOfSpecies>
  <species id="Protein" name="Protein" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
  <species id="ProteinP" name="Protein" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
  <species id="ATP" name="ATP" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
  <species id="ADP" name="ADP" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
  <species id="P" name="P" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
  <species id="ProteinKinase" name="Protein_Kinase" compartment="Cell"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
</listOfSpecies>
<listOfReactions>
  <reaction id="Phosphorylation" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference id="SpeciesReference_Protein"
        species="Protein" stoichiometry="1" constant="true"/>
      <speciesReference id="SpeciesReference_ATP"
        species="ATP" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference id="SpeciesReference_ProteinP"
        species="ProteinP" stoichiometry="1" constant="true"/>
      <speciesReference id="SpeciesReference_ADAP"
        species="ADP" stoichiometry="1" constant="true"/>
    </listOfProducts>
    <listOfModifiers>
      <modifierSpeciesReference id="ModifierSpeciesReference_ProteinKinase"
        species="ProteinKinase"/>
    </listOfModifiers>
  </reaction>
  <reaction id="Dephosphorylation" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference id="SpeciesReference_ProteinP_rev"
        species="ProteinP" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference id="SpeciesReference_Protein_rev"
        species="Protein" stoichiometry="1" constant="true"/>
      <speciesReference id="SpeciesReference_P"

```

```

    species="P" stoichiometry="1" constant="true"/>
  </listOfProducts>
</reaction>
</listOfReactions>
<layout:listOfLayouts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1">
  <layout:layout layout:id="Layout_1">
    <layout:dimensions layout:width="450" layout:height="400"/>
    <layout:listOfSpeciesGlyphs>
      <layout:speciesGlyph layout:id="SpeciesGlyph_Protein" layout:species="Protein">
        <layout:boundingBox layout:id="bb1">
          <layout:position layout:x="30" layout:y="230"/>
          <layout:dimensions layout:width="80" layout:height="40"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
      <layout:speciesGlyph layout:id="SpeciesGlyph_ProteinP"
render:objectRole="phosphorylated" layout:species="ProteinP">
        <layout:boundingBox layout:id="bb2">
          <layout:position layout:x="330" layout:y="230"/>
          <layout:dimensions layout:width="93" layout:height="40"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
      <layout:speciesGlyph layout:id="SpeciesGlyph_ATP" layout:species="ATP">
        <layout:boundingBox layout:id="bb3">
          <layout:position layout:x="110" layout:y="100"/>
          <layout:dimensions layout:width="50" layout:height="30"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
      <layout:speciesGlyph layout:id="SpeciesGlyph_ADP" layout:species="ADP">
        <layout:boundingBox layout:id="bb4">
          <layout:position layout:x="280" layout:y="100"/>
          <layout:dimensions layout:width="50" layout:height="30"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
      <layout:speciesGlyph layout:id="SpeciesGlyph_P" layout:species="P">
        <layout:boundingBox layout:id="bb5">
          <layout:position layout:x="170" layout:y="320"/>
          <layout:dimensions layout:width="30" layout:height="30"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
      <layout:speciesGlyph layout:id="SpeciesGlyph_ProteinKinase"
layout:species="ProteinKinase">
        <layout:boundingBox layout:id="bb6">
          <layout:position layout:x="180" layout:y="30"/>
          <layout:dimensions layout:width="80" layout:height="50"/>
        </layout:boundingBox>
      </layout:speciesGlyph>
    </layout:listOfSpeciesGlyphs>
    <layout:listOfReactionGlyphs>
      <layout:reactionGlyph layout:id="ReactionGlyph_Phosphorylation"
layout:reaction="Phosphorylation">
        <layout:curve>
          <layout:listOfCurveSegments>
            <layout:curveSegment
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CubicBezier">
              <layout:start layout:x="115" layout:y="225"/>
              <layout:end layout:x="205" layout:y="200"/>
              <layout:basePoint1 layout:x="170" layout:y="200"/>
              <layout:basePoint2 layout:x="170" layout:y="200"/>
            </layout:curveSegment>
          </layout:listOfCurveSegments>
        </layout:curve>
      </layout:listOfReactionGlyphs>
      <layout:listOfSpeciesReferenceGlyphs>
        <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_Protein"
render:objectRole="substrate" layout:speciesReference="SpeciesReference_Protein"
layout:speciesGlyph="SpeciesGlyph_Protein" layout:role="substrate">

```

```

<layout:curve>
  <layout:listOfCurveSegments>
    <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="CubicBezier">
      <layout:start layout:x="115" layout:y="225"/>
      <layout:end layout:x="205" layout:y="200"/>
      <layout:basePoint1 layout:x="170" layout:y="200"/>
      <layout:basePoint2 layout:x="170" layout:y="200"/>
    </layout:curveSegment>
  </layout:listOfCurveSegments>
</layout:curve>
</layout:speciesReferenceGlyph>
<layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ATP"
  render:objectRole="sidesubstrate" layout:speciesReference="SpeciesReference_ATP"
  layout:speciesGlyph="SpeciesGlyph_ATP" layout:role="sidesubstrate">
  <layout:curve>
    <layout:listOfCurveSegments>
      <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="CubicBezier">
        <layout:start layout:x="160" layout:y="135"/>
        <layout:end layout:x="205" layout:y="200"/>
        <layout:basePoint1 layout:x="180" layout:y="200"/>
        <layout:basePoint2 layout:x="180" layout:y="200"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>
  </layout:curve>
</layout:speciesReferenceGlyph>
<layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ProteinP"
  render:objectRole="product" layout:speciesReference="SpeciesReference_ProteinP"
  layout:speciesGlyph="SpeciesGlyph_ProteinP" layout:role="product">
  <layout:curve>
    <layout:listOfCurveSegments>
      <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="CubicBezier">
        <layout:start layout:x="235" layout:y="200"/>
        <layout:end layout:x="320" layout:y="230"/>
        <layout:basePoint1 layout:x="270" layout:y="200"/>
        <layout:basePoint2 layout:x="270" layout:y="200"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>
  </layout:curve>
</layout:speciesReferenceGlyph>
<layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ADAP"
  render:objectRole="sideproduct" layout:speciesReference="SpeciesReference_ADAP"
  layout:speciesGlyph="SpeciesGlyph_ADAP" layout:role="sideproduct">
  <layout:curve>
    <layout:listOfCurveSegments>
      <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="CubicBezier">
        <layout:start layout:x="235" layout:y="200"/>
        <layout:end layout:x="275" layout:y="140"/>
        <layout:basePoint1 layout:x="260" layout:y="200"/>
        <layout:basePoint2 layout:x="260" layout:y="200"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>
  </layout:curve>
</layout:speciesReferenceGlyph>
<layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ProteinKinase"
  render:objectRole="catalyst"
  layout:speciesReference="ModifierSpeciesReference_ProteinKinase"
  layout:speciesGlyph="SpeciesGlyph_ProteinKinase" layout:role="activator">
  <layout:curve>
    <layout:listOfCurveSegments>
      <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="LineSegment">
        <layout:start layout:x="220" layout:y="85"/>

```

```

        <layout:end layout:x="220" layout:y="180"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>
  </layout:curve>
</layout:speciesReferenceGlyph>
</layout:listOfSpeciesReferenceGlyphs>
</layout:reactionGlyph>
<layout:reactionGlyph layout:id="ReactionGlyph_Dephosphorylation"
layout:reaction="Dephosphorylation">
  <layout:curve>
    <layout:listOfCurveSegments>
      <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CubicBezier">
        <layout:start layout:x="325" layout:y="265"/>
        <layout:end layout:x="235" layout:y="290"/>
        <layout:basePoint1 layout:x="270" layout:y="290"/>
        <layout:basePoint2 layout:x="270" layout:y="290"/>
      </layout:curveSegment>
    </layout:listOfCurveSegments>
  </layout:curve>
<layout:listOfSpeciesReferenceGlyphs>
  <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ProteinP_rev"
render:objectRole="substrate" layout:speciesReference="SpeciesReference_ProteinP_rev"
layout:speciesGlyph="SpeciesGlyph_ProteinP" layout:role="substrate">
    <layout:curve>
      <layout:listOfCurveSegments>
        <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CubicBezier">
          <layout:start layout:x="325" layout:y="265"/>
          <layout:end layout:x="235" layout:y="290"/>
          <layout:basePoint1 layout:x="270" layout:y="290"/>
          <layout:basePoint2 layout:x="270" layout:y="290"/>
        </layout:curveSegment>
      </layout:listOfCurveSegments>
    </layout:curve>
  </layout:speciesReferenceGlyph>
  <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_Protein_rev"
render:objectRole="product" layout:speciesReference="SpeciesReference_Protein_rev"
layout:speciesGlyph="SpeciesGlyph_Protein" layout:role="product">
    <layout:curve>
      <layout:listOfCurveSegments>
        <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CubicBezier">
          <layout:start layout:x="205" layout:y="290"/>
          <layout:end layout:x="115" layout:y="265"/>
          <layout:basePoint1 layout:x="170" layout:y="290"/>
          <layout:basePoint2 layout:x="170" layout:y="290"/>
        </layout:curveSegment>
      </layout:listOfCurveSegments>
    </layout:curve>
  </layout:speciesReferenceGlyph>
  <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_P"
render:objectRole="sideproduct" layout:speciesReference="SpeciesReference_P"
layout:speciesGlyph="SpeciesGlyph_P" layout:role="sideproduct">
    <layout:curve>
      <layout:listOfCurveSegments>
        <layout:curveSegment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CubicBezier">
          <layout:start layout:x="205" layout:y="290"/>
          <layout:end layout:x="185" layout:y="310"/>
          <layout:basePoint1 layout:x="190" layout:y="300"/>
          <layout:basePoint2 layout:x="190" layout:y="300"/>
        </layout:curveSegment>
      </layout:listOfCurveSegments>
    </layout:curve>
  </layout:speciesReferenceGlyph>

```



```

    </layout:listOfSpeciesReferenceGlyphs>
  </layout:reactionGlyph>
</layout:listOfReactionGlyphs>
<layout:listOfTextGlyphs>
  <layout:textGlyph layout:id="TextGlyph_Protein" layout:originOfText="Protein"
    layout:graphicalObject="SpeciesGlyph_Protein">
    <layout:boundingBox layout:id="bb9">
      <layout:position layout:x="30" layout:y="220"/>
      <layout:dimensions layout:width="80" layout:height="40"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_ProteinP" layout:originOfText="ProteinP"
    layout:graphicalObject="SpeciesGlyph_ProteinP">
    <layout:boundingBox layout:id="bb10">
      <layout:position layout:x="330" layout:y="220"/>
      <layout:dimensions layout:width="80" layout:height="40"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_ATP" layout:originOfText="ATP"
    layout:graphicalObject="SpeciesGlyph_ATP">
    <layout:boundingBox layout:id="bb11">
      <layout:position layout:x="110" layout:y="95"/>
      <layout:dimensions layout:width="50" layout:height="30"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_ADP" layout:originOfText="ADP"
    layout:graphicalObject="SpeciesGlyph_ADP">
    <layout:boundingBox layout:id="bb12">
      <layout:position layout:x="280" layout:y="95"/>
      <layout:dimensions layout:width="50" layout:height="30"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_P" layout:originOfText="P"
    layout:graphicalObject="SpeciesGlyph_P">
    <layout:boundingBox layout:id="bb13">
      <layout:position layout:x="170" layout:y="315"/>
      <layout:dimensions layout:width="30" layout:height="30"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_ProteinKinase1" layout:text="Protein-"
    layout:graphicalObject="SpeciesGlyph_ProteinKinase">
    <layout:boundingBox layout:id="bb14">
      <layout:position layout:x="180" layout:y="35"/>
      <layout:dimensions layout:width="80" layout:height="20"/>
    </layout:boundingBox>
  </layout:textGlyph>
  <layout:textGlyph layout:id="TextGlyph_ProteinKinase2" layout:text="kinase"
    layout:graphicalObject="SpeciesGlyph_ProteinKinase">
    <layout:boundingBox layout:id="bb15">
      <layout:position layout:x="180" layout:y="55"/>
      <layout:dimensions layout:width="80" layout:height="20"/>
    </layout:boundingBox>
  </layout:textGlyph>
</layout:listOfTextGlyphs>
<render:listOfRenderInformation
  xmlns:render="http://www.sbml.org/sbml/level3/version1/render/version1"
  render:versionMajor="1" render:versionMinor="0">
  <render:renderInformation render:id="SBGN"
    render:programName="Ralph_Gauges" render:programVersion="1.0"
    render:backgroundColor="#FFFFFF">
    <render:listOfColorDefinitions>
      <render:colorDefinition render:id="black" render:value="#000000"/>
      <render:colorDefinition render:id="white" render:value="#ffffff"/>
    </render:listOfColorDefinitions>
    <render:listOfLineEndings>
      <render:lineEnding

```

```

xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="productionHead">
  <layout:boundingBox>
    <layout:position layout:x="-10" layout:y="-6"/>
    <layout:dimensions layout:width="14" layout:height="10"/>
  </layout:boundingBox>
  <render:g render:stroke="black" render:stroke-width="1" render:fill="black">
    <render:polygon>
      <render:ListOfElements>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="RenderPoint" render:x="0" render:y="0"/>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="RenderPoint" render:x="14" render:y="5"/>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="RenderPoint" render:x="14" render:y="5"/>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="RenderPoint" render:x="0" render:y="10"/>
      </render:ListOfElements>
    </render:polygon>
  </render:g>
</render:lineEnding>
<render:lineEnding
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="catalysisHead">
  <layout:boundingBox>
    <layout:position layout:x="0" layout:y="-7"/>
    <layout:dimensions layout:width="14" layout:height="14"/>
  </layout:boundingBox>
  <render:g render:stroke="black" render:stroke-width="2">
    <render:ellipse render:cx="50%" render:cy="50%" render:rx="50%"/>
  </render:g>
</render:lineEnding>
</render:ListOfLineEndings>
<render:ListOfStyles>
  <render:style render:id="proteinKinaseStyle"
    render:idList="SpeciesGlyph_ProteinKinase">
    <render:g render:stroke="black" render:stroke-width="3"
      render:fill-rule="nonzero" render:font-size="12" render:font-family="monospace"
      render:font-style="normal" render:font-weight="normal"
      render:text-anchor="middle" render:vtext-anchor="top">
      <render:rectangle render:x="0" render:y="0" render:width="100%"
        render:height="100%" render:rx="10" render:ry="10"/>
      <text render:x="0" render:y="-4.6" render:text-anchor="middle"
        render:vtext-anchor="top">Protein-</text>
      <text render:x="0" render:y="15.4" render:text-anchor="middle"
        render:vtext-anchor="top">kinase</text>
    </render:g>
  </render:style>
  <render:style render:id="proteinStyle" render:idList="SpeciesGlyph_Protein">
    <render:g render:stroke="black" render:stroke-width="3"
      render:fill-rule="nonzero" render:font-size="12" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"
      render:vtext-anchor="middle">
      <render:rectangle render:x="0" render:y="0"
        render:width="100%" render:height="100%" render:rx="10" render:ry="10"/>
      <text render:x="0" render:y="-9.6" render:font-family="monospace"
        render:vtext-anchor="middle" render:vtext-anchor="middle">Protein</text>
    </render:g>
  </render:style>
  <render:style render:id="proteinPStyle" render:idList="SpeciesGlyph_ProteinP">
    <render:g render:stroke="black" render:stroke-width="3"
      render:fill-rule="nonzero" render:font-size="12" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"
      render:vtext-anchor="middle">
      <render:rectangle render:x="0" render:y="0"
        render:width="90%" render:height="100%" render:rx="10" render:ry="10"/>

```

```

<render:ellipse render:fill="white" render:cx="90%" render:cy="50%" render:rx="10"/>
<text render:x="-10" render:y="-9.6" render:font-family="monospace"
render:text-anchor="middle" render:vtext-anchor="middle">Protein</text>
<text render:x="-5%" render:y="-9.6" render:font-family="monospace"
render:text-anchor="end" render:vtext-anchor="middle">P</text>
</render:g>
</render:style>
<render:style render:id="ATPStyle" render:idList="SpeciesGlyph_ATP">
<render:g render:stroke="black" render:stroke-width="3"
render:fill-rule="nonzero" render:font-size="12" render:font-family="sans-serif"
render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"
render:vtext-anchor="middle">
<render:ellipse render:cx="20+50%" render:cy="50%" render:rx="17"/>
<text render:x="20" render:y="-5.6" render:font-family="monospace"
render:text-anchor="middle" render:vtext-anchor="middle">ATP</text>
</render:g>
</render:style>
<render:style render:id="ADPStyle" render:idList="SpeciesGlyph_ADAP">
<render:g render:stroke="black" render:stroke-width="3"
render:fill-rule="nonzero" render:font-size="12" render:font-family="sans-serif"
render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"
render:vtext-anchor="middle">
<render:ellipse render:cx="-20+50%" render:cy="50%" render:rx="17"/>
<text render:x="-20" render:y="-5.6" render:font-family="monospace"
render:text-anchor="middle" render:vtext-anchor="middle">ADP</text>
</render:g>
</render:style>
<render:style render:id="PStyle" render:idList="SpeciesGlyph_P">
<render:g render:stroke="black" render:stroke-width="3"
render:fill-rule="nonzero" render:font-size="12" render:font-family="sans-serif"
render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"
render:vtext-anchor="middle">
<render:ellipse render:cx="50%" render:cy="50%" render:rx="15"/>
<text render:x="0" render:y="-5.6" render:font-family="monospace"
render:text-anchor="middle" render:vtext-anchor="middle">P</text>
</render:g>
</render:style>
<render:style render:id="reactionGlyphStyle" render:typeList="REACTIONGLYPH">
<render:g render:stroke="black" render:stroke-width="2"
render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
render:font-style="normal" render:font-weight="normal">
<render:curve>
<render:listOfElements>
<render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="0" render:y="5"/>
<render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="10" render:y="5"/>
</render:listOfElements>
</render:curve>
<render:curve>
<render:listOfElements>
<render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="20" render:y="5"/>
<render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="30" render:y="5"/>
</render:listOfElements>
</render:curve>
<render:rectangle render:x="10" render:y="0" render:width="10" render:height="10"/>
</render:g>
</render:style>
<render:style render:id="textGlyphStyle" render:typeList="TEXTGLYPH">
<render:g render:stroke-width="0" render:fill-rule="nonzero"
render:font-size="0" render:font-family="sans-serif" render:font-style="normal"
render:font-weight="normal"/>
</render:style>
<render:style render:id="substrateSpeciesReferenceGlyphStyle"

```

```

render:roleList="sidesubstrate_substrate">
  <render:g render:stroke="#000000" render:stroke-width="2"
    render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
    render:font-style="normal" render:font-weight="normal"/>
</render:style>
<render:style render:id="productSpeciesReferenceGlyphStyle"
render:roleList="product_sideproduct">
  <render:g render:stroke="#000000" render:stroke-width="2"
    render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
    render:font-style="normal" render:font-weight="normal"
    render:endHead="productionHead"/>
</render:style>
<render:style render:id="activatorSpeciesReferenceGlyphStyle"
render:roleList="activator_catalyst">
  <render:g render:stroke="black" render:stroke-width="2"
    render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
    render:font-style="normal" render:font-weight="normal"
    render:endHead="catalysisHead"/>
</render:style>
</render:listOfStyles>
</render:renderInformation>
</render:listOfRenderInformation>
</layout:layout>
<render:listOfGlobalRenderInformation
xmlns:render="http://www.sbml.org/sbml/level3/version1/render/version1"
render:versionMajor="1" render:versionMinor="0">
  <render:renderInformation render:id="wireFrame" render:programName="Ralph_Gauges"
render:programVersion="1.0" render:backgroundColor="#FFFFFF">
    <render:listOfColorDefinitions>
      <render:colorDefinition render:id="white" render:value="#ffffff"/>
      <render:colorDefinition render:id="black" render:value="#000000"/>
    </render:listOfColorDefinitions>
    <render:listOfLineEndings>
      <render:lineEnding
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="simpleHead_black">
        <layout:boundingBox>
          <layout:position layout:x="-8" layout:y="-3"/>
          <layout:dimensions layout:width="10" layout:height="6"/>
        </layout:boundingBox>
        <render:g render:stroke="black" render:stroke-width="1" render:fill="black">
          <render:polygon>
            <render:listOfElements>
              <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="0" render:y="0"/>
              <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="10" render:y="3"/>
              <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="10" render:y="3"/>
              <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="0" render:y="6"/>
            </render:listOfElements>
          </render:polygon>
        </render:g>
      </render:lineEnding>
    <render:lineEnding
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="catalysisHead_black">
      <layout:boundingBox>
        <layout:position layout:x="0" layout:y="-5"/>
        <layout:dimensions layout:width="10" layout:height="10"/>
      </layout:boundingBox>
      <render:g render:stroke="black" render:stroke-width="1">
        <render:ellipse render:cx="50%" render:cy="50%" render:rx="50%"/>
      </render:g>
    </render:lineEnding>
  </render:renderInformation>
</render:listOfGlobalRenderInformation>

```

```

</render:listOfLineEndings>
<render:listOfStyles>
  <render:style render:id="speciesGlyphStyle" render:typeList="SPECIESGLYPH">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal">
      <render:rectangle render:x="0" render:y="0" render:width="100%"
        render:height="100%"/>
    </render:g>
  </render:style>
  <render:style render:id="phosphorylatedSpeciesGlyphStyle"
    render:roleList="phosphorylated">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="12" render:font-family="monospace"
      render:font-style="normal" render:font-weight="normal">
      <render:rectangle render:x="0" render:y="0" render:width="90%"
        render:height="100%"/>
      <render:ellipse render:fill="white" render:cx="90%" render:cy="50%"
        render:rx="10"/>
      <text render:x="85%" render:y="-9.6" render:vtext-anchor="middle">P</text>
    </render:g>
  </render:style>
  <render:style render:id="speciesReferenceAndTextGlyphStyle"
    render:typeList="SPECIESREFERENCEGLYPH_TEXTGLYPH">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="12" render:font-family="sans"
      render:font-style="normal" render:font-weight="normal" render:text-anchor="middle"/>
  </render:style>
  <render:style render:id="productStyle" render:roleList="product_sideproduct">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal"
      render:endHead="simpleHead_black"/>
  </render:style>
  <render:style render:id="activatorStyle" render:roleList="activator_catalyst">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal"
      render:endHead="catalysisHead_black"/>
  </render:style>
  <render:style render:id="reactionGlyphStyle" render:typeList="REACTIONGLYPH">
    <render:g render:stroke="black" render:stroke-width="1"
      render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
      render:font-style="normal" render:font-weight="normal">
      <render:curve>
        <render:listOfElements>
          <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="RenderPoint" render:x="0" render:y="5"/>
          <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="RenderPoint" render:x="10" render:y="5"/>
        </render:listOfElements>
      </render:curve>
      <render:curve>
        <render:listOfElements>
          <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="RenderPoint" render:x="20" render:y="5"/>
          <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="RenderPoint" render:x="30" render:y="5"/>
        </render:listOfElements>
      </render:curve>
      <render:rectangle render:x="10" render:y="0"
        render:width="10" render:height="10"/>
    </render:g>
  </render:style>
</render:listOfStyles>
</render:renderInformation>

```

```

<render:renderInformation render:id="defaultGrayStyle"
render:programName="Ralph_Gauges" render:programVersion="1.0"
render:backgroundColor="#FFFFFF">
  <render:listOfColorDefinitions>
    <render:colorDefinition render:id="lightGray" render:value="#cecece"/>
    <render:colorDefinition render:id="white" render:value="#ffffff"/>
    <render:colorDefinition render:id="black" render:value="#000000"/>
    <render:colorDefinition render:id="lightGray2" render:value="#f0f0f0"/>
    <render:colorDefinition render:id="gray" render:value="#0b0b0b"/>
  </render:listOfColorDefinitions>
  <render:listOfGradientDefinitions>
    <render:radialGradient render:id="speciesGlyphGradient">
      <render:stop render:offset="0" render:stop-color="white"/>
      <render:stop render:offset="100%" render:stop-color="lightGray"/>
    </render:radialGradient>
  </render:listOfGradientDefinitions>
  <render:listOfLineEndings>
    <render:lineEnding
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="simpleHead_black">
      <layout:boundingBox>
        <layout:position layout:x="-8" layout:y="-3"/>
        <layout:dimensions layout:width="10" layout:height="6"/>
      </layout:boundingBox>
      <render:g render:stroke="black" render:stroke-width="1" render:fill="black">
        <render:polygon>
          <render:listOfElements>
            <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="0" render:y="0"/>
            <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="10" render:y="3"/>
            <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="10" render:y="3"/>
            <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="RenderPoint" render:x="0" render:y="6"/>
          </render:listOfElements>
        </render:polygon>
      </render:g>
    </render:lineEnding>
    <render:lineEnding
xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
render:id="catalysisHead_black">
      <layout:boundingBox>
        <layout:position layout:x="0" layout:y="-5"/>
        <layout:dimensions layout:width="10" layout:height="10"/>
      </layout:boundingBox>
      <render:g render:stroke="black" render:stroke-width="1">
        <render:ellipse render:cx="50%" render:cy="50%" render:rx="50%"/>
      </render:g>
    </render:lineEnding>
  </render:listOfLineEndings>
  <render:listOfStyles>
    <render:style render:id="speciesGlyphStyle" render:typeList="SPECIESGLYPH">
      <render:g render:stroke="black" render:stroke-width="1"
render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
render:font-style="normal" render:font-weight="normal">
        <render:rectangle render:fill="speciesGlyphGradient" render:x="0"
render:y="0" render:width="100%" render:height="100%"
render:rx="5%" render:ry="5%"/>
      </render:g>
    </render:style>
    <render:style render:id="phosphorylatedSpeciesGlyphStyle"
render:roleList="phosphorylated">
      <render:g render:stroke="black" render:stroke-width="1"
render:fill-rule="nonzero" render:font-size="12" render:font-family="monospace"
render:font-style="normal" render:font-weight="normal">

```

```

    <render:rectangle render:fill="speciesGlyphGradient" render:x="0" render:y="0"
    render:width="90%" render:height="100%">/>
    <render:ellipse render:fill="speciesGlyphGradient"
    render:cx="90%" render:cy="50%" render:rx="10"/>
    <text render:x="85%" render:y="-9.6" render:vtext-anchor="middle">P</text>
  </render:g>
</render:style>
<render:style render:id="speciesReferenceAndTextGlyphStyle"
render:roleList="SPECIESREFERENCEGLYPH_TEXTGLYPH">
  <render:g render:stroke="black" render:stroke-width="1"
  render:fill-rule="nonzero" render:font-size="12" render:font-family="sans"
  render:font-style="normal" render:font-weight="normal"
  render:text-anchor="middle"/>
</render:style>
<render:style render:id="speciesReferenceGlyphStyle"
render:roleList="product_sideproduct">
  <render:g render:stroke="#000000" render:stroke-width="1"
  render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
  render:font-style="normal" render:font-weight="normal"
  render:endHead="simpleHead_black"/>
</render:style>
<render:style render:id="activatorStyle" render:roleList="activator_catalyst">
  <render:g render:stroke="black" render:stroke-width="1"
  render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
  render:font-style="normal" render:font-weight="normal"
  render:endHead="catalysisHead_black"/>
</render:style>
<render:style render:id="reactionGlyphStyle" render:roleList="REACTIONGLYPH">
  <render:g render:stroke="black" render:stroke-width="1"
  render:fill-rule="nonzero" render:font-size="0" render:font-family="sans-serif"
  render:font-style="normal" render:font-weight="normal">
    <render:curve>
      <render:listOfElements>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="RenderPoint" render:x="0" render:y="5"/>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="RenderPoint" render:x="10" render:y="5"/>
      </render:listOfElements>
    </render:curve>
    <render:curve>
      <render:listOfElements>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="RenderPoint" render:x="20" render:y="5"/>
        <render:element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="RenderPoint" render:x="30" render:y="5"/>
      </render:listOfElements>
    </render:curve>
    <render:rectangle render:x="10" render:y="0" render:width="10" render:height="10"/>
  </render:g>
</render:style>
</render:listOfStyles>
</render:renderInformation>
<render:renderInformation render:id="colorStyle"
render:referenceRenderInformation="defaultGrayStyle"
render:programName="Ralph_Gauges" render:programVersion="1.0"
render:backgroundcolor="#FFFFFF">
  <render:listOfColorDefinitions>
    <render:colorDefinition render:id="lightGray" render:value="#9999f0"/>
    <render:colorDefinition render:id="lightGray2" render:value="#9999f0"/>
    <render:colorDefinition render:id="gray" render:value="#cecece"/>
  </render:listOfColorDefinitions>
</render:renderInformation>
</render:listOfGlobalRenderInformation>
</layout:listOfLayouts>
</model>
</sbml>

```

SBML Level 2 Version 1

```

<?xml version="1.0" encoding="utf-8"?>
<!--Created on: 4/14/2015 12:14:01 PM-->
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
  <model id="ProteinPhosphorylation">
    <annotation>
      <listOfLayouts xmlns="http://projects.embl.org/bcb/sbml/level2"
        xmlns:render="http://projects.embl.org/bcb/sbml/render/level2"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <annotation>
          <listOfGlobalRenderInformation xmlns="http://projects.embl.org/bcb/sbml/render/level2">
            <renderInformation id="wireFrame" programName="Ralph_Gauges" programVersion="1.0">
              <listOfColorDefinitions>
                <colorDefinition id="white" value="#FFFFFF" />
                <colorDefinition id="black" value="#000000" />
              </listOfColorDefinitions>
              <listOfGradientDefinitions />
              <listOfLineEndings>
                <lineEnding id="simpleHead_black" enableRotationalMapping="true">
                  <boundingBox>
                    <position x="-8" y="-3" />
                    <dimensions width="10" height="6" />
                  </boundingBox>
                  <g stroke="black" stroke-width="1.0" fill="black">
                    <polygon>
                      <listOfElements>
                        <element xsi:type="RenderPoint" x="0" y="0" z="0"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="10" y="3" z="0"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="10" y="3" z="0"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="0" y="6" z="0"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                      </listOfElements>
                    </polygon>
                  </g>
                </lineEnding>
                <lineEnding id="catalysisHead_black" enableRotationalMapping="true">
                  <boundingBox>
                    <position x="0" y="-5" />
                    <dimensions width="10" height="10" />
                  </boundingBox>
                  <g stroke="black" stroke-width="1.0" fill="none">
                    <ellipse cx="50%" cy="50%" rx="50%" ry="50%" />
                  </g>
                </lineEnding>
              </listOfLineEndings>
              <listOfStyles>
                <style id="speciesGlyphStyle" typeList="SPECIESGLYPH">
                  <g stroke="black" stroke-width="1.0">
                    <rectangle fill="none" x="0%" y="0%"
                      width="100%" height="100%" rx="0%" ry="0%" />
                  </g>
                </style>
                <style id="phosphorylatedSpeciesGlyphStyle" roleList="phosphorylated">
                  <g stroke="black" stroke-width="1.0" font-family="monospace" font-size="12.0">
                    <rectangle fill="none" x="0%" y="0%" width="90%" height="100%" rx="0%" ry="0%" />
                    <ellipse fill="white" cx="90%" cy="50%" rx="10.0" ry="10.0" />
                    <text x="85%" y="0.0" vtext-anchor="middle">P</text>
                  </g>
                </style>
                <style id="speciesReferenceAndTextGlyphStyle"
                  typeList="SPECIESREFERENCEGLYPH_TEXTGLYPH">
                  <g stroke="black" stroke-width="1.0" font-family="sans" font-size="12"

```



```

        text-anchor="middle" />
    </style>
    <style id="productStyle" roleList="product_sideproduct">
        <g stroke="black" stroke-width="1.0" endHead="simpleHead_black" />
    </style>
    <style id="activatorStyle" roleList="activator_catalyst">
        <g stroke="black" stroke-width="1.0" endHead="catalysisHead_black" />
    </style>
    <style id="reactionGlyphStyle" typeList="REACTIONGLYPH">
        <g stroke="black" stroke-width="1.0">
            <curve>
                <listOfElements>
                    <element xsi:type="RenderPoint" x="0" y="5" z="0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                    <element xsi:type="RenderPoint" x="10" y="5" z="0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                </listOfElements>
            </curve>
            <curve>
                <listOfElements>
                    <element xsi:type="RenderPoint" x="20" y="5" z="0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                    <element xsi:type="RenderPoint" x="30" y="5" z="0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                </listOfElements>
            </curve>
            <rectangle x="10.0" y="0.0" width="10.0" height="10.0" />
        </g>
    </style>
</listOfStyles>
</renderInformation>
<renderInformation id="defaultGrayStyle"
    programName="Ralph_Gauges" programVersion="1.0">
    <listOfColorDefinitions>
        <colorDefinition id="lightGray" value="#CECECE" />
        <colorDefinition id="white" value="#FFFFFF" />
        <colorDefinition id="black" value="#000000" />
        <colorDefinition id="lightGray2" value="#F0F0F0" />
        <colorDefinition id="gray" value="#0B0B0B" />
    </listOfColorDefinitions>
    <listOfGradientDefinitions>
        <radialGradient id="speciesGlyphGradient" cx="50%" cy="50%" cz="50%" r="50%">
            <stop offset="0%" stop-color="white" />
            <stop offset="100%" stop-color="lightGray" />
        </radialGradient>
    </listOfGradientDefinitions>
    <listOfLineEndings>
        <lineEnding id="simpleHead_black" enableRotationalMapping="true">
            <boundingBox>
                <position x="-8" y="-3" />
                <dimensions width="10" height="6" />
            </boundingBox>
            <g stroke="black" stroke-width="1.0" fill="black">
                <polygon>
                    <listOfElements>
                        <element xsi:type="RenderPoint" x="0" y="0" z="0"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="10" y="3" z="0"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="10" y="3" z="0"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                        <element xsi:type="RenderPoint" x="0" y="6" z="0"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                    </listOfElements>
                </polygon>
            </g>
        </lineEnding>
    </listOfLineEndings>

```

```

</lineEnding>
<lineEnding id="catalysisHead_black" enableRotationalMapping="true">
  <boundingBox>
    <position x="0" y="-5" />
    <dimensions width="10" height="10" />
  </boundingBox>
  <g stroke="black" stroke-width="1.0" fill="none">
    <ellipse cx="50%" cy="50%" rx="50%" ry="50%" />
  </g>
</lineEnding>
</listOfLineEndings>
<listOfStyles>
  <style id="speciesGlyphStyle" typeList="SPECIESGLYPH">
    <g stroke="black" stroke-width="1.0">
      <rectangle fill="speciesGlyphGradient" x="0%" y="0%"
        width="100%" height="100%" rx="5%" />
    </g>
  </style>
  <style id="phosphorylatedSpeciesGlyphStyle" roleList="phosphorylated">
    <g stroke="black" stroke-width="1.0" font-family="monospace" font-size="12.0">
      <rectangle fill="speciesGlyphGradient" x="0%" y="0%"
        width="90%" height="100%" rx="0%" ry="0%" />
      <ellipse fill="speciesGlyphGradient" cx="90%" cy="50%" rx="10.0" ry="10.0" />
      <text x="85%" y="0.0" vtext-anchor="middle">P</text>
    </g>
  </style>
  <style id="speciesReferenceAndTextGlyphStyle"
    typeList="SPECIESREFERENCEGLYPH_TEXTGLYPH">
    <g stroke="black" stroke-width="1.0" font-family="sans"
      font-size="12" text-anchor="middle" />
  </style>
  <style id="speciesReferenceGlyphStyle" roleList="product_sideproduct">
    <g stroke="#000000" stroke-width="1.0" endHead="simpleHead_black" />
  </style>
  <style id="activatorStyle" roleList="activator_catalyst">
    <g stroke="black" stroke-width="1.0" endHead="catalysisHead_black" />
  </style>
  <style id="reactionGlyphStyle" typeList="REACTIONGLYPH">
    <g stroke="black" stroke-width="1.0">
      <curve>
        <listOfElements>
          <element xsi:type="RenderPoint" x="0" y="5" z="0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
          <element xsi:type="RenderPoint" x="10" y="5" z="0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
        </listOfElements>
      </curve>
      <curve>
        <listOfElements>
          <element xsi:type="RenderPoint" x="20" y="5" z="0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
          <element xsi:type="RenderPoint" x="30" y="5" z="0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
        </listOfElements>
      </curve>
      <rectangle x="10.0" y="0.0" width="10.0" height="10.0" />
    </g>
  </style>
</listOfStyles>
</renderInformation>
<renderInformation id="colorStyle"
  programName="Ralph_Gauges" programVersion="1.0"
  referenceRenderInformation="defaultGrayStyle">
  <listOfColorDefinitions>
    <colorDefinition id="lightGray" value="#9999F0" />
    <colorDefinition id="lightGray2" value="#9999F0" />
  </listOfColorDefinitions>

```

```

        <colorDefinition id="gray" value="#CECECE" />
    </listOfColorDefinitions>
    <listOfGradientDefinitions />
    <listOfLineEndings />
    <listOfStyles />
</renderInformation>
</listOfGlobalRenderInformation>
</annotation>
<layout id="Layout_1" xmlns="http://projects.eml.org/bcb/sbml/level2">
    <annotation>
        <listOfRenderInformation xmlns="http://projects.eml.org/bcb/sbml/render/level2">
            <renderInformation id="SBGN" programName="Ralph_Gauges" programVersion="1.0">
                <listOfColorDefinitions>
                    <colorDefinition id="black" value="#000000" />
                    <colorDefinition id="white" value="#FFFFFF" />
                </listOfColorDefinitions>
                <listOfGradientDefinitions />
                <listOfLineEndings>
                    <lineEnding id="productionHead" enableRotationalMapping="true">
                        <boundingBox>
                            <position x="-10" y="-6" />
                            <dimensions width="14" height="10" />
                        </boundingBox>
                        <g stroke="black" stroke-width="1.0" fill="black">
                            <polygon>
                                <listOfElements>
                                    <element xsi:type="RenderPoint" x="0" y="0" z="0"
                                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                                    <element xsi:type="RenderPoint" x="14" y="5" z="0"
                                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                                    <element xsi:type="RenderPoint" x="14" y="5" z="0"
                                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                                    <element xsi:type="RenderPoint" x="0" y="10" z="0"
                                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
                                </listOfElements>
                            </polygon>
                        </g>
                    </lineEnding>
                    <lineEnding id="catalysisHead" enableRotationalMapping="true">
                        <boundingBox>
                            <position x="0" y="-7" />
                            <dimensions width="14" height="14" />
                        </boundingBox>
                        <g stroke="black" stroke-width="2.0" fill="none">
                            <ellipse cx="50%" cy="50%" rx="50%" ry="50%" />
                        </g>
                    </lineEnding>
                </listOfLineEndings>
                <listOfStyles>
                    <style id="proteinKinaseStyle" idList="SpeciesGlyph_ProteinKinase">
                        <g stroke="black" stroke-width="3.0" font-family="monospace"
                            font-size="12.0" text-anchor="middle" vtext-anchor="top">
                            <rectangle fill="none" x="0%" y="0%"
                                width="100%" height="100%" rx="10.0" ry="10.0" />
                            <text x="0.0" y="5.0" text-anchor="middle" vtext-anchor="top">Protein</text>
                            <text x="0.0" y="25.0" text-anchor="middle" vtext-anchor="top">kinase</text>
                        </g>
                    </style>
                    <style id="proteinStyle" idList="SpeciesGlyph_Protein">
                        <g stroke="black" stroke-width="3.0" font-size="12.0"
                            text-anchor="middle" vtext-anchor="middle">
                            <rectangle fill="none" x="0%" y="0%"
                                width="100%" height="100%" rx="10.0" ry="10.0" />
                            <text x="0.0" y="0.0" font-family="monospace"
                                text-anchor="middle" vtext-anchor="middle">Protein</text>
                        </g>
                    </style>
                </listOfStyles>
            </renderInformation>
        </listOfRenderInformation>
    </annotation>
</layout>

```

```

</style>
<style id="proteinPStyle" idList="SpeciesGlyph_ProteinP">
  <g stroke="black" stroke-width="3.0" font-size="12.0"
    text-anchor="middle" vtext-anchor="middle">
    <rectangle fill="none" x="0%" y="0%"
      width="90%" height="100%" rx="10.0" ry="10.0" />
    <ellipse fill="white" cx="90%" cy="50%" rx="10.0" ry="10.0" />
    <text x="-10.0" y="0.0" font-family="monospace"
      text-anchor="middle" vtext-anchor="middle">Protein</text>
    <text x="-5%" y="0.0" font-family="monospace"
      text-anchor="end" vtext-anchor="middle">P</text>
  </g>
</style>
<style id="ATPStyle" idList="SpeciesGlyph_ATP">
  <g stroke="black" stroke-width="3.0" font-size="12.0"
    text-anchor="middle" vtext-anchor="middle">
    <ellipse fill="none" cx="20.0+50%" cy="50%" rx="17.0" ry="17.0" />
    <text x="20.0" y="4.0" font-family="monospace"
      text-anchor="middle" vtext-anchor="middle">ATP</text>
  </g>
</style>
<style id="ADPStyle" idList="SpeciesGlyph_ADAP">
  <g stroke="black" stroke-width="3.0" font-size="12.0"
    text-anchor="middle" vtext-anchor="middle">
    <ellipse fill="none" cx="-20.0+50%" cy="50%" rx="17.0" ry="17.0" />
    <text x="-20.0" y="4.0" font-family="monospace"
      text-anchor="middle" vtext-anchor="middle">ADP</text>
  </g>
</style>
<style id="PStyle" idList="SpeciesGlyph_P">
  <g stroke="black" stroke-width="3.0" font-size="12.0"
    text-anchor="middle" vtext-anchor="middle">
    <ellipse fill="none" cx="50%" cy="50%" rx="15.0" ry="15.0" />
    <text x="0.0" y="4.0" font-family="monospace"
      text-anchor="middle" vtext-anchor="middle">P</text>
  </g>
</style>
<style id="reactionGlyphStyle" typeList="REACTIONGLYPH">
  <g stroke="black" stroke-width="2.0">
    <curve>
      <listOfElements>
        <element xsi:type="RenderPoint" x="0" y="5" z="0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
        <element xsi:type="RenderPoint" x="10" y="5" z="0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
      </listOfElements>
    </curve>
    <curve>
      <listOfElements>
        <element xsi:type="RenderPoint" x="20" y="5" z="0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
        <element xsi:type="RenderPoint" x="30" y="5" z="0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
      </listOfElements>
    </curve>
    <rectangle x="10.0" y="0.0" width="10.0" height="10.0" />
  </g>
</style>
<style id="textGlyphStyle" typeList="TEXTGLYPH">
  <g stroke="none" stroke-width="0.0" font-size="0.0" />
</style>
<style id="substrateSpeciesReferenceGlyphStyle" roleList="substrate_sidesubstrate">
  <g stroke="#000000" stroke-width="2.0" />
</style>
<style id="productSpeciesReferenceGlyphStyle" roleList="product_sideproduct">
  <g stroke="#000000" stroke-width="2.0" endHead="productionHead" />

```

```

        </style>
        <style id="activatorSpeciesReferenceGlyphStyle" roleList="activator_catalyst">
          <g stroke="black" stroke-width="2.0" endHead="catalysisHead" />
        </style>
      </listOfStyles>
    </renderInformation>
  </listOfRenderInformation>
</annotation>
<dimensions width="450" height="400" />
<listOfSpeciesGlyphs>
  <speciesGlyph id="SpeciesGlyph_Protein" species="Protein">
    <boundingBox id="bb1">
      <position x="30" y="230" />
      <dimensions width="80" height="40" />
    </boundingBox>
  </speciesGlyph>
  <speciesGlyph id="SpeciesGlyph_ProteinP" render:objectRole="phosphorylated"
species="ProteinP" xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
    <boundingBox id="bb2">
      <position x="330" y="230" />
      <dimensions width="93" height="40" />
    </boundingBox>
  </speciesGlyph>
  <speciesGlyph id="SpeciesGlyph_ATP" species="ATP">
    <boundingBox id="bb3">
      <position x="110" y="100" />
      <dimensions width="50" height="30" />
    </boundingBox>
  </speciesGlyph>
  <speciesGlyph id="SpeciesGlyph_ADP" species="ADP">
    <boundingBox id="bb4">
      <position x="280" y="100" />
      <dimensions width="50" height="30" />
    </boundingBox>
  </speciesGlyph>
  <speciesGlyph id="SpeciesGlyph_P" species="P">
    <boundingBox id="bb5">
      <position x="170" y="320" />
      <dimensions width="30" height="30" />
    </boundingBox>
  </speciesGlyph>
  <speciesGlyph id="SpeciesGlyph_ProteinKinase" species="ProteinKinase">
    <boundingBox id="bb6">
      <position x="180" y="30" />
      <dimensions width="80" height="50" />
    </boundingBox>
  </speciesGlyph>
</listOfSpeciesGlyphs>
<listOfReactionGlyphs>
  <reactionGlyph id="ReactionGlyph_Phosphorylation" reaction="Phosphorylation">
    <boundingBox id="bb7">
      <position x="205" y="195" />
      <dimensions width="30" height="10" />
    </boundingBox>
    <curve>
      <listOfCurveSegments>
        <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <start x="115" y="225" />
          <end x="205" y="200" />
          <basePoint1 x="170" y="200" />
          <basePoint2 x="170" y="200" />
        </curveSegment>
      </listOfCurveSegments>
    </curve>
  </listOfReactionGlyphs>
</listOfSpeciesReferenceGlyphs>

```

```

<speciesReferenceGlyph id="SpeciesReferenceGlyph_Protein"
render:objectRole="substrate" speciesReference="SpeciesReference_Protein"
speciesGlyph="SpeciesGlyph_Protein" role="substrate"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />
    <dimensions width="0" height="0" />
  </boundingBox>
  <curve>
    <listOfCurveSegments>
      <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <start x="115" y="225" />
        <end x="205" y="200" />
        <basePoint1 x="170" y="200" />
        <basePoint2 x="170" y="200" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_ATP"
render:objectRole="sidesubstrate" speciesReference="SpeciesReference_ATP"
speciesGlyph="SpeciesGlyph_ATP" role="sidesubstrate"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />
    <dimensions width="0" height="0" />
  </boundingBox>
  <curve>
    <listOfCurveSegments>
      <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <start x="160" y="135" />
        <end x="205" y="200" />
        <basePoint1 x="180" y="200" />
        <basePoint2 x="180" y="200" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_ProteinP"
render:objectRole="product" speciesReference="SpeciesReference_ProteinP"
speciesGlyph="SpeciesGlyph_ProteinP" role="product"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />
    <dimensions width="0" height="0" />
  </boundingBox>
  <curve>
    <listOfCurveSegments>
      <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <start x="235" y="200" />
        <end x="320" y="230" />
        <basePoint1 x="270" y="200" />
        <basePoint2 x="270" y="200" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_ADAP"
render:objectRole="sideproduct" speciesReference="SpeciesReference_ADAP"
speciesGlyph="SpeciesGlyph_ADAP" role="sideproduct"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />

```

```

        <dimensions width="0" height="0" />
    </boundingBox>
    <curve>
        <listOfCurveSegments>
            <curveSegment xsi:type="CubicBezier"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <start x="235" y="200" />
                <end x="275" y="140" />
                <basePoint1 x="260" y="200" />
                <basePoint2 x="260" y="200" />
            </curveSegment>
        </listOfCurveSegments>
    </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_ProteinKinase"
    render:objectRole="catalyst" speciesReference="ModifierSpeciesReference_ProteinKinase"
    speciesGlyph="SpeciesGlyph_ProteinKinase" role="activator"
    xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
    <boundingBox>
        <position x="0" y="0" />
        <dimensions width="0" height="0" />
    </boundingBox>
    <curve>
        <listOfCurveSegments>
            <curveSegment xsi:type="LineSegment"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <start x="220" y="85" />
                <end x="220" y="180" />
            </curveSegment>
        </listOfCurveSegments>
    </curve>
</speciesReferenceGlyph>
</listOfSpeciesReferenceGlyphs>
</reactionGlyph>
<reactionGlyph id="ReactionGlyph_Dephosphorylation" reaction="Dephosphorylation">
    <boundingBox id="bb8">
        <position x="205" y="285" />
        <dimensions width="30" height="10" />
    </boundingBox>
    <curve>
        <listOfCurveSegments>
            <curveSegment xsi:type="CubicBezier"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <start x="325" y="265" />
                <end x="235" y="290" />
                <basePoint1 x="270" y="290" />
                <basePoint2 x="270" y="290" />
            </curveSegment>
        </listOfCurveSegments>
    </curve>
<listOfSpeciesReferenceGlyphs>
    <speciesReferenceGlyph id="SpeciesReferenceGlyph_ProteinP_rev"
        render:objectRole="substrate" speciesReference="SpeciesReference_ProteinP_rev"
        speciesGlyph="SpeciesGlyph_ProteinP" role="substrate"
        xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
        <boundingBox>
            <position x="0" y="0" />
            <dimensions width="0" height="0" />
        </boundingBox>
        <curve>
            <listOfCurveSegments>
                <curveSegment xsi:type="CubicBezier"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <start x="325" y="265" />
                    <end x="235" y="290" />
                    <basePoint1 x="270" y="290" />

```

```

        <basePoint2 x="270" y="290" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_Protein_rev"
render:objectRole="product" speciesReference="SpeciesReference_Protein_rev"
speciesGlyph="SpeciesGlyph_Protein" role="product"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />
    <dimensions width="0" height="0" />
  </boundingBox>
  <curve>
    <listOfCurveSegments>
      <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <start x="205" y="290" />
        <end x="115" y="265" />
        <basePoint1 x="170" y="290" />
        <basePoint2 x="170" y="290" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
<speciesReferenceGlyph id="SpeciesReferenceGlyph_P"
render:objectRole="sideproduct" speciesReference="SpeciesReference_P"
speciesGlyph="SpeciesGlyph_P" role="sideproduct"
xmlns:render="http://projects.embl.org/bcb/sbml/render/level2">
  <boundingBox>
    <position x="0" y="0" />
    <dimensions width="0" height="0" />
  </boundingBox>
  <curve>
    <listOfCurveSegments>
      <curveSegment xsi:type="CubicBezier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <start x="205" y="290" />
        <end x="185" y="310" />
        <basePoint1 x="190" y="300" />
        <basePoint2 x="190" y="300" />
      </curveSegment>
    </listOfCurveSegments>
  </curve>
</speciesReferenceGlyph>
</listOfSpeciesReferenceGlyphs>
</reactionGlyph>
</listOfReactionGlyphs>
<listOfTextGlyphs>
  <textGlyph id="TextGlyph_Protein" graphicalObject="SpeciesGlyph_Protein"
originOfText="Protein">
    <boundingBox id="bb9">
      <position x="30" y="220" />
      <dimensions width="80" height="40" />
    </boundingBox>
  </textGlyph>
  <textGlyph id="TextGlyph_ProteinP" graphicalObject="SpeciesGlyph_ProteinP"
originOfText="ProteinP">
    <boundingBox id="bb10">
      <position x="330" y="220" />
      <dimensions width="80" height="40" />
    </boundingBox>
  </textGlyph>
  <textGlyph id="TextGlyph_ATP" graphicalObject="SpeciesGlyph_ATP" originOfText="ATP">
    <boundingBox id="bb11">
      <position x="110" y="95" />
    </boundingBox>
  </textGlyph>

```



```

        <dimensions width="50" height="30" />
      </boundingBox>
    </textGlyph>
    <textGlyph id="TextGlyph_ADP" graphicalObject="SpeciesGlyph_ADP" originOfText="ADP">
      <boundingBox id="bb12">
        <position x="280" y="95" />
        <dimensions width="50" height="30" />
      </boundingBox>
    </textGlyph>
    <textGlyph id="TextGlyph_P" graphicalObject="SpeciesGlyph_P" originOfText="P">
      <boundingBox id="bb13">
        <position x="170" y="315" />
        <dimensions width="30" height="30" />
      </boundingBox>
    </textGlyph>
    <textGlyph id="TextGlyph_ProteinKinase1"
graphicalObject="SpeciesGlyph_ProteinKinase" text="Protein-">
      <boundingBox id="bb14">
        <position x="180" y="35" />
        <dimensions width="80" height="20" />
      </boundingBox>
    </textGlyph>
    <textGlyph id="TextGlyph_ProteinKinase2"
graphicalObject="SpeciesGlyph_ProteinKinase" text="kinase">
      <boundingBox id="bb15">
        <position x="180" y="55" />
        <dimensions width="80" height="20" />
      </boundingBox>
    </textGlyph>
  </listOfTextGlyphs>
</layout>
</listOfLayouts>
</annotation>
<listOfCompartments>
  <compartment id="Cell" />
</listOfCompartments>
<listOfSpecies>
  <species id="Protein" name="Protein" compartment="Cell" />
  <species id="ProteinP" name="Protein" compartment="Cell" />
  <species id="ATP" name="ATP" compartment="Cell" />
  <species id="ADP" name="ADP" compartment="Cell" />
  <species id="P" name="P" compartment="Cell" />
  <species id="ProteinKinase" name="Protein_Kinase" compartment="Cell" />
</listOfSpecies>
<listOfReactions>
  <reaction id="Phosphorylation" reversible="false">
    <listOfReactants>
      <speciesReference species="Protein">
        <annotation>
          <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
id="SpeciesReference_Protein" />
        </annotation>
      </speciesReference>
      <speciesReference species="ATP">
        <annotation>
          <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
id="SpeciesReference_ATP" />
        </annotation>
      </speciesReference>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="ProteinP">
        <annotation>
          <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
id="SpeciesReference_ProteinP" />
        </annotation>
      </speciesReference>
    </listOfProducts>
  </reaction>
</listOfReactions>

```

```

    </speciesReference>
    <speciesReference species="ADP">
      <annotation>
        <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
          id="SpeciesReference_ADAP" />
      </annotation>
    </speciesReference>
  </listOfProducts>
  <listOfModifiers>
    <modifierSpeciesReference species="ProteinKinase">
      <annotation>
        <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
          id="ModifierSpeciesReference_ProteinKinase" />
      </annotation>
    </modifierSpeciesReference>
  </listOfModifiers>
</reaction>
<reaction id="Dephosphorylation" reversible="false">
  <listOfReactants>
    <speciesReference species="ProteinP">
      <annotation>
        <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
          id="SpeciesReference_ProteinP_rev" />
      </annotation>
    </speciesReference>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="Protein">
      <annotation>
        <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
          id="SpeciesReference_Protein_rev" />
      </annotation>
    </speciesReference>
    <speciesReference species="P">
      <annotation>
        <layoutId xmlns="http://projects.embl.org/bcb/sbml/level2"
          id="SpeciesReference_P" />
      </annotation>
    </speciesReference>
  </listOfProducts>
</reaction>
</listOfReactions>
</model>
</sbml>

```

5 Best practices

In this section, we recommend a number of practices for using and interpreting various constructs in the Render package. These recommendations are non-normative, but we advocate them strongly; ignoring them will not render a model invalid, but may reduce interoperability between software and models.

5.1 Text

A **Text** has an optional attribute **font-family** of type **string** that allows to specify the font or font-family to be used for the text element. For maximum interoperability the font families specified in **FontFamily** have to be supported at a minimum. Those are the generic families “**serif**”, “**sans-serif**” and “**monospace**”. It is recommended good practice to use only these fonts to allow for greater exchangeability.

5.2 Image

If the referenced resource can not be found, it is up to the application if nothing is drawn or some place holder is displayed. Preferably the user would get some kind of notification about the missing resource.

6 Future development

In this section we highlight some open issues not addressed in this version of the Render specification.

6.1 Values for `StyleType`

Concerning the valid keywords for the `roleList` attribute we had thought about taking those from some kind of controlled vocabulary. Preferably, this would be some kind of ontology like SBO. The specifics of this will have to be discussed with other interested parties

6.2 3D drawings

Both the transformations and graphical primitives considered here have been limited to the 2D situation.

6.3 Text

outlined or filled-outlined text are currently not covered by this specification.

6.4 Image

To include bitmaps into a graphical representation we use the `Image` element from SVG. The `Image` element in SVG can also be used to include complete SVG vector images which we explicitly exclude in this version of the proposal since we think it would be too complex. If the need for the inclusion of SVG drawings arises, it is only a matter of rephrasing this specification.

Non-local image resources (e.g. from the web) are currently not supported.

A Text Anchor Examples

A.1 Vertical Text Anchor Examples

The following figures illustrate the use of the different `VTextAnchor` values.

A.1.1 Top

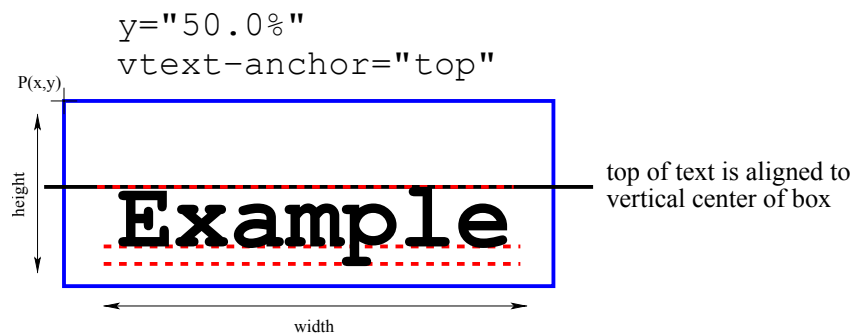


Figure 17: vertical text alignment `top`

A.1.2 Bottom

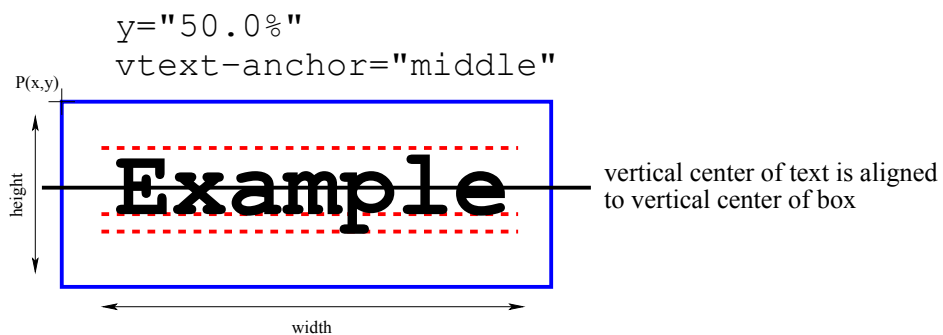


Figure 18: vertical text alignment `bottom`

A.1.3 Middle

1

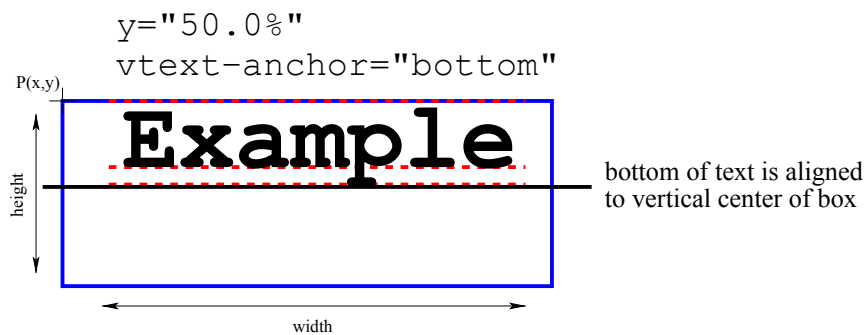


Figure 19: vertical text alignment middle

A.1.4 Baseline

2

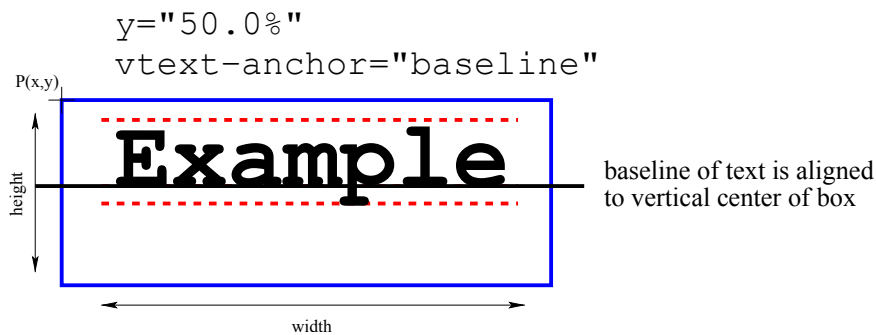


Figure 20: vertical text alignment baseline

A.2 Horizontal Text Anchor Examples

The following figures illustrate the use of different `HTextAnchor` values.

A.2.1 Start

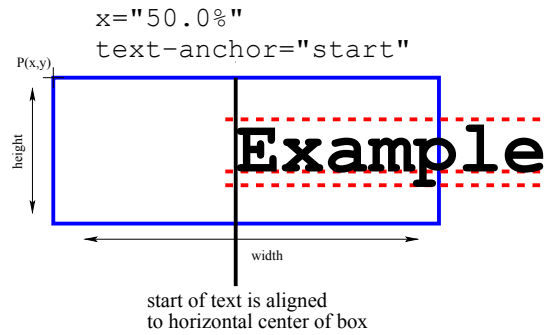


Figure 21: horizontal text alignment start

A.2.2 Middle

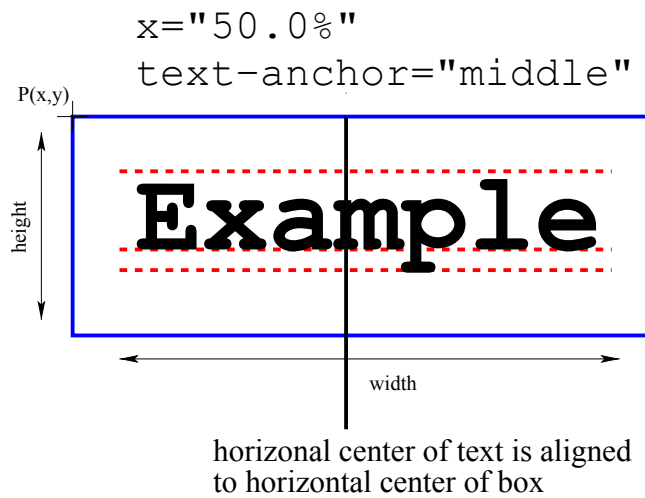


Figure 22: horizontal text alignment middle

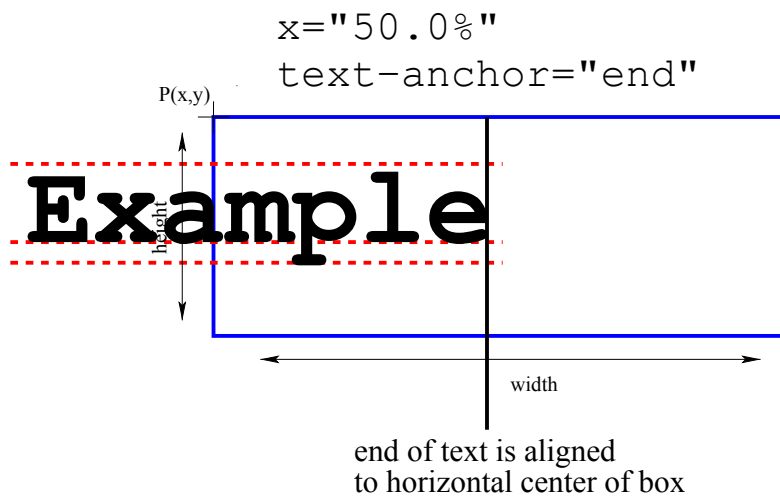
A.2.3 End

Figure 23: horizontal text alignment **end**

B Transformations

There are four basic transformation operations that can be combined in a affine transformation matrix.

B.1 Translation

Translating something means moving it some distance along one or more of the axes. The corresponding 2D transformation matrix is

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

where tx and ty are the distance along the x and y axes by which the object shall be moved.

B.2 Scaling

Scaling means to multiply all coordinate components of an object by a certain value. The corresponding 2D transformation matrix is

$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where sx and sy are the scaling factors along the x and y axis respectively.

B.3 Rotation

With a rotation, an object can be rotated around the origin of the coordinate system. The corresponding 2D transformation matrix is

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where α is the angle of rotation around the origin.

B.4 Skewing

Skewing is the least used operation and we have to distinguish between skewing along the x or the y axis. The corresponding 2D transformation matrices are

$$\begin{bmatrix} 1 & \tan(\alpha) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan(\beta) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where α is the skewing angle of skewing along the x axis and β is the angle for skewing along the y axis.

Combining several of the operations above means multiplying the transformation matrices that belong to the individual operations. Depending on the matrices that are multiplied, the order of the operations matter, e.g. it makes a difference if an object is translated before it is rotated or if it is rotated first.

If an object specifies a transformation, this transformation is to be applied to the object prior to any other coordinate properties of the object. E.g. if a rectangle specifies a position of $x = 10$ and $y = 20$ and it also specifies a rotation by 45 degrees, the rotation is applied before the object is placed at $P(10,20)$. The transformation for an object is always in relation to the objects view port. For most render objects, this would be the bounding box of the corresponding layout object. For layout curves, e.g. in reaction glyphs or species reference glyphs, the view port is the complete diagram. For objects defined in line endings, the view port is the bounding box of the line ending before it is applied to the line.

example:

```
<g ...>
  <text x="50%" y="50%" text-anchor="middle" stroke="#FF0000"
    font-family="serif" font-size="20.0"
    transform="1.0,3.0,2.5,1.4,4.0,5.0">This is a Text</text>
  ...
</g>
```

C Resolving render information

C.1 Mapping line endings to curves

In order to apply a line ending which is defined using only 2D coordinates onto a line which has been defined using 3D coordinates, we need to define a kind of mapping. The first definition we make is that the origin of the line ending view port is mapped to the end of the line to which the line ending is applied. If the **enableRotationalMapping** attribute is set to “false”, the line endings coordinate system is the same as the global coordinate system used to draw the layout, only the origin is moved to that end of the line the line ending is applied to. If the **enableRotationalMapping** attribute is set to “true”, which is the default, we define that the x,y-plane of the line endings view port is mapped to the plane that results from taking the unit vector of the slope of the line and the unit vector that results from ortho-normalizing the slope vector and a second vector that has no component along the z axis. If the slope of the line has a positive component along the x axis, the ortho-normalized vector also has to have a positive component along the y axis. In order to retain the right handed coordinate system, the z axis of the line endings coordinate system is perpendicular to the plane created by the other two vectors and has a positive component along the global coordinate systems z-axis. Likewise if the slope has a negative component along the global coordinate systems x axis, the y component of the ortho-normalized second vector has a negative component along the y axis of the global coordinate system and to retain the right handed coordinate system, the third vector which is perpendicular to the plane made by the slope and its ortho-normalized vector, has a positive component along the global coordinate systems z axis.

If the slope of the line points directly along the positive z axis of the global coordinate system, the line endings coordinate system is mapped to the line ending by a -90 rotation around the y axis of the line endings coordinate system and a translation of the origin of the line endings coordinate system to the end of the line. If the slope points directly down the negative z axis, the line endings coordinate system has to be rotated by +90 around its y axis before translation to the position of the curves end.

This may all sound very complicated, but in the end, the calculations to be done are not difficult and straight forward.

The mapping of arrow heads to line endings involves some transformations which we would like to illustrate with two examples. The first example as depicted in Figure [Figure 24](#) defines a straight line and a line ending which is to be applied to the end of the line. The line ending specifies a bounding box with a size of 4x4 and a position of $P(-2, -2)$. The origin of the line ending is at $o(0.0, 0.0, 0.0)$ and the bounding box extends along the positive x- and y-axes. The position of the bounding box is the offset by which the origin of the bounding box has to be translated from the endpoint of the curve.

Since the arrow head in the first example explicitly disables rotation mapping by specifying **enableRotationalMapping=false** in the definition of the line ending, the process of mapping the arrow head to the line is simply a matter of moving the origin of the line endings coordinate system to the end point of the line $E(ex, ey)$ plus the offset that is specified as the position $P(px, py, pz)$ of the line endings bounding box $F = E + P = (ex + px, ey + py, ez + pz)$. In our example the origin of the line endings coordinate system has to be moved 2 units up and two to the left of the end of the curve that the line ending is applied to.

The result of this operation is depicted in Figure [Figure 25](#).

The second example is very similar to the first example, only now, the rotational mapping for the arrow head is enabled. This means that we now have to execute two steps in order to map the arrow head to the line ending.

First we need to rotate the arrow head so that the x-axis of the arrow heads coordinate system is aligned with the slope $s = \frac{dy}{dx}$ of the curve.

The rotation of the arrow head involves the following steps:

1. calculate the normalized direction vector of the slope:

We first need to find the two points that determine the slope at the end of the line. One point is always the

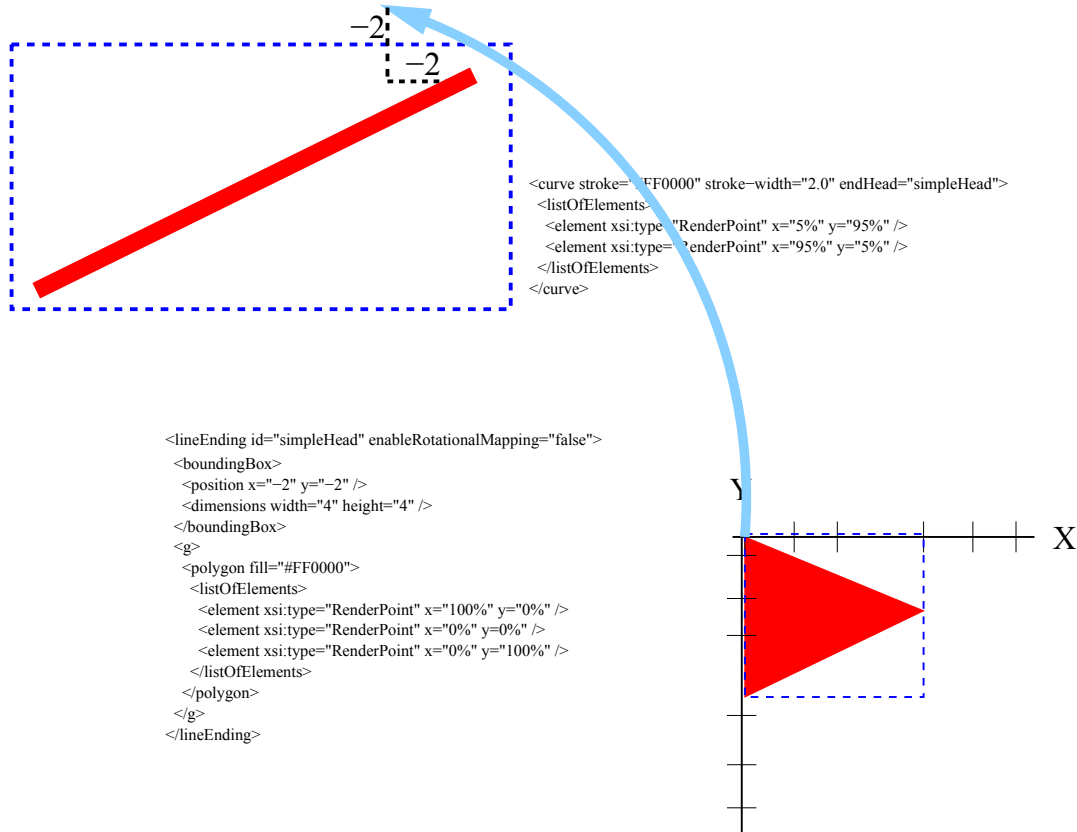


Figure 24: Curve with arrow head and no rotational mapping

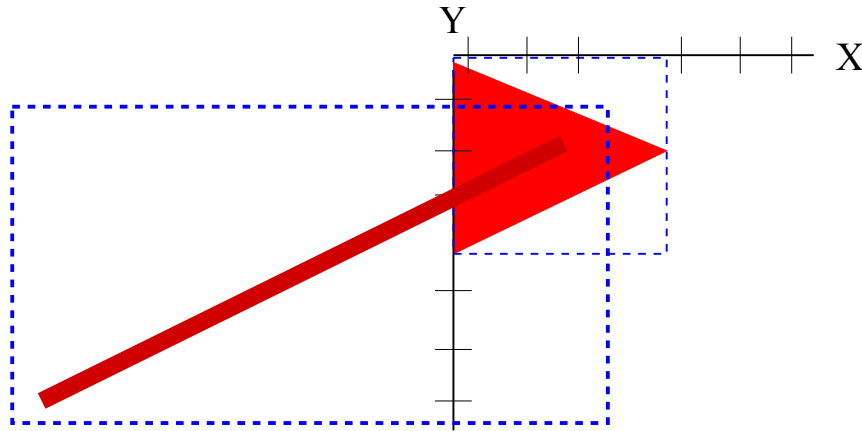


Figure 25: Curve with mapped arrow head and no rotational mapping

endpoint of the line ($E(ex, ey, ez)$). The second point depends on whether the last element of the line is a straight line or if it is a bezier element. If it is a bezier element, the second point is the second base point of the bezier element, if it is a straight line, it is either the preceding point or the endpoint of the preceding bezier element. We call this second point $S(sy, dy, sz)$.

The direction vector can be calculated as $v(vx, vy, vz) = (ex - sy, ey - sy, ez - sz)$. To normalize the vector we have to calculate the length $l = \sqrt{vx^2 + vy^2 + vz^2}$ of the direction vector and divide all elements of v by this length. $v_n(v_nx, v_ny, v_nz) = (vx/l, vy/l, vz/l)$

Step 1: Rotation

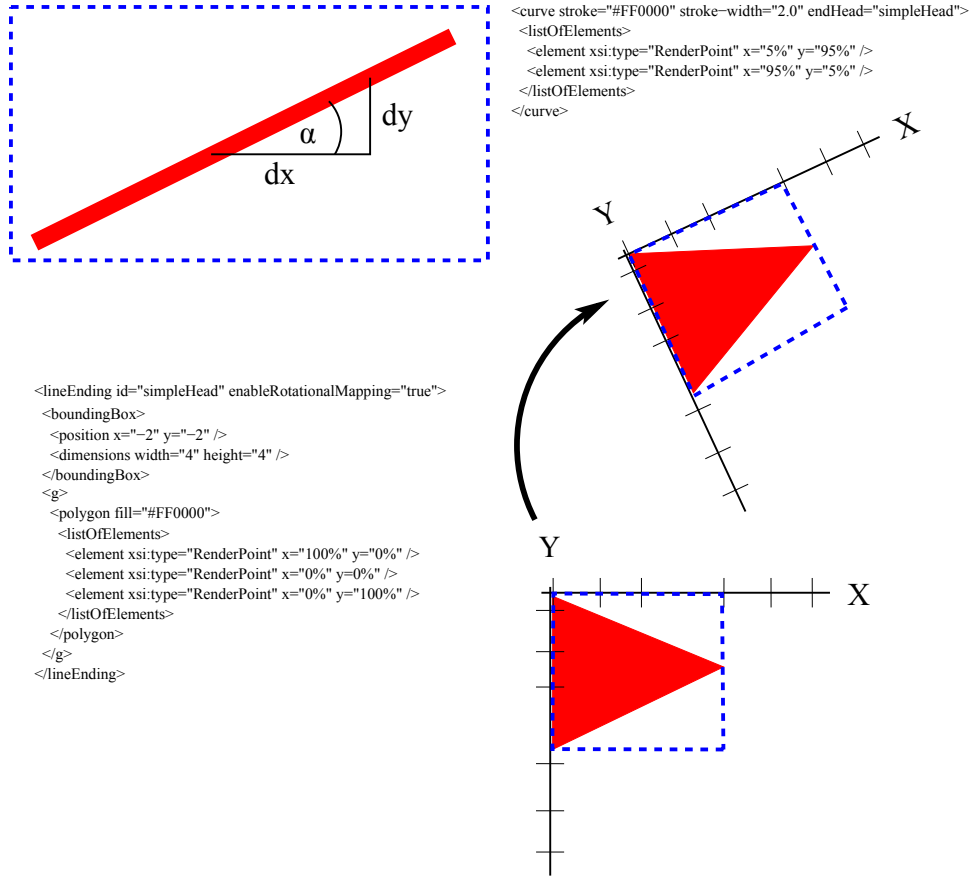


Figure 26: Step 1: Rotation

2. calculate the normalized vector that is

- (a) orthogonal to the direction vector of the line
- (b) located in the plane x- and y-axis

If the direction vector is parallel to the y-axis ($v_x = 0.0$), the orthogonal vector w is parallel to the x-axis ($w(v_y, 0, 0)$). For all other cases w is $w(w_x, w_y, w_z) = (-v_n y, 1 - v_n y^2, -v_n y * v_n z)$.

Again we have to normalize this vector by dividing through its length $n = \sqrt{w_x^2 + w_y^2 + w_z^2}$, which results in the normlized vector

$$w_n(w_n x, w_n y, w_n z) = (w_x / n, w_y / n, w_z / n).$$

3. create the transformation matrix that converts the original coordinate system into the coordinate system that is made up of the two calculated vectors:

The transformation matrix that results from the two normalized vector that we calculated in the steps above

$$\text{is } m = \begin{pmatrix} v_n x & w_n x & 0.0 & 0.0 \\ v_n y & w_n y & 0.0 & 0.0 \\ v_n z & w_n z & 0.0 & 1.0 \end{pmatrix}$$

The second step moves the origin of the arrow heads coordinate system to the endpoint of the line, which is exactly the same as we did in the first example.

Step 2: Translation

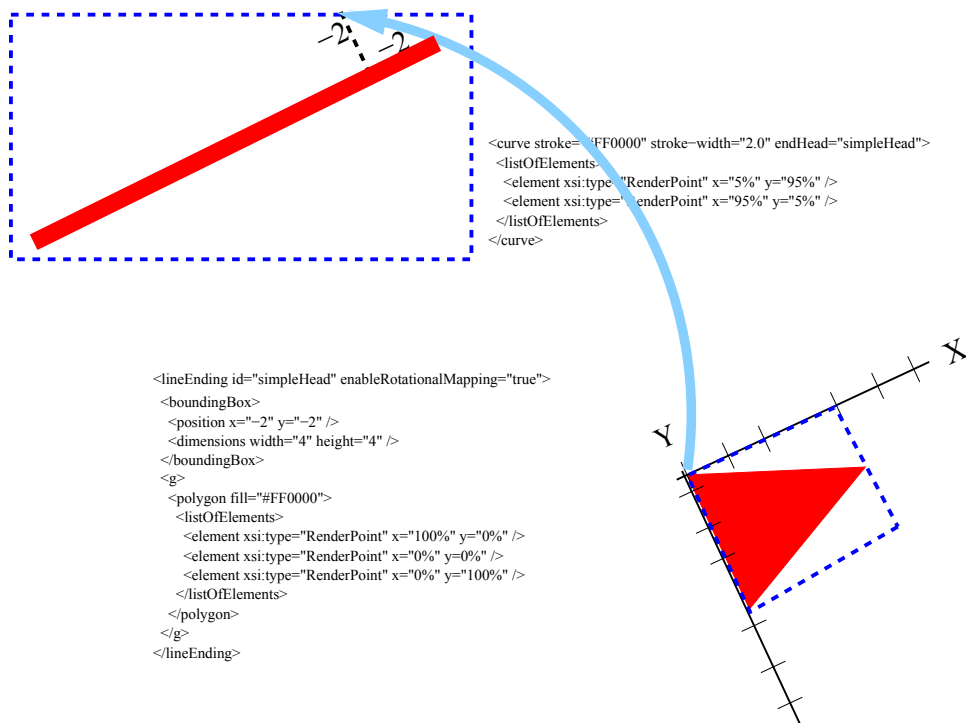


Figure 27: Step 2: Translation

Result

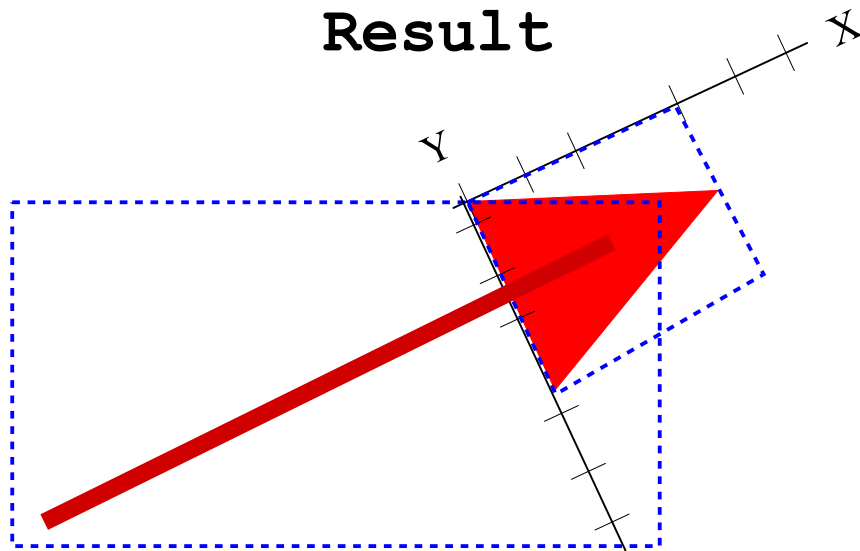


Figure 28: Curve with mapped arrow head and rotational mapping

Mapping of an arrow head to the beginning of a curve is exactly the same as for the end of a curve, only the direction of the line has to be reversed and in case of a cubic bezier, one has to use the first base point rather than the second base point.

1
2
3

C.2 Style resolution

To resolve which style applies to a certain object, one should follow the rule that more specific style definitions take precedence over less specific ones and that if there are several styles with the same specificity, the first one encountered in the file is to be used. In essence, this means that a program first has to search the local render information for a style that references the id of the object. If none is found, it searches for a style that mentions the role of the object. If it has one, see next section. If it does not find one, it searches for a style for the type of the object.

If a render information references another render information object via its **referenceRenderInformation** attribute, the program has to go through that one as well to see if a more specific render information is present there. If the chain of referenced **RenderInformation** objects has been searched and no style has been found that fits, it is up to the program how the object is rendered.

If several type based styles are found that would fit, a style that applies to only one type takes precedence over a style that applies to several types.

If a program explicitly wants to define render information that states that some objects are not to be rendered at all, it has to define a style that does nothing, i.e. has no render information but applies to the objects that should not be rendered.

C.3 Role resolution

This render extension explicitly provides means to write render information that renders layout objects based on certain roles those render objects or their corresponding model objects have. So far SBML models or layouts do not contain such role information or only for a limited number of objects if one would consider the role attribute of **SpeciesReferenceGlyph** objects to fall into this category. Although there is currently no means to specify these roles, there are already initiatives underway that try to complement SBML files with more biological information based on ontologies. One of these initiatives, the sboTerms, is about to be included into SBML Level 2 Version 2. This ontology or a similar one could provide this role information for layout objects in the future.

For the time being, we define an additional attribute called **objectRole** for all layout objects derived from **GraphicalObject** including **GraphicalObject** itself. The attribute specifies a user defined role string. render information including the same role string in its **roleList** attribute applies to the object. This is only true if no more specific render information takes precedence (see "Style resolution").

A specific style can reference one or more roles to which it applies. When a program tries to determine which style applies to a specific object it might have to determine the role of the object layout first. If the layout object itself has a role, this will be taken, otherwise if the layout object is associated with an object in the model, the program should get the role from the associated object. If none of them has a role, no role based style can be applied to the object.

C.4 Style information for reaction glyphs and species reference glyphs

When defining a style for a **ReactionGlyph** or **SpeciesReferenceGlyph** object, one has to distinguish between layout objects that only specify a bounding box for the object and those that specify a curve. In the case of a bounding box, you want to define complete render information, whereas in the case of a curve, you only want to set certain attributes that determine certain aspects of how the curve should be drawn, e.g. its color. To resolve this conflict, the style for such an object has to define render information for both cases. The render information for the case of a bounding box is specified just like render information for any other object within a group. Render information for the case of a curve is defined by the appropriate attributes that are in effect in the outermost **RenderGroup** object itself. Those attributes include **stroke**, **stroke-width** and **stroke-dasharray**. Additionally **startHead** and **endHead** can be specified to define line endings for layout curve objects. If the group does not define one or more of these attributes, the default value is used (see also [Appendix 3.3.2 on page 12](#)).

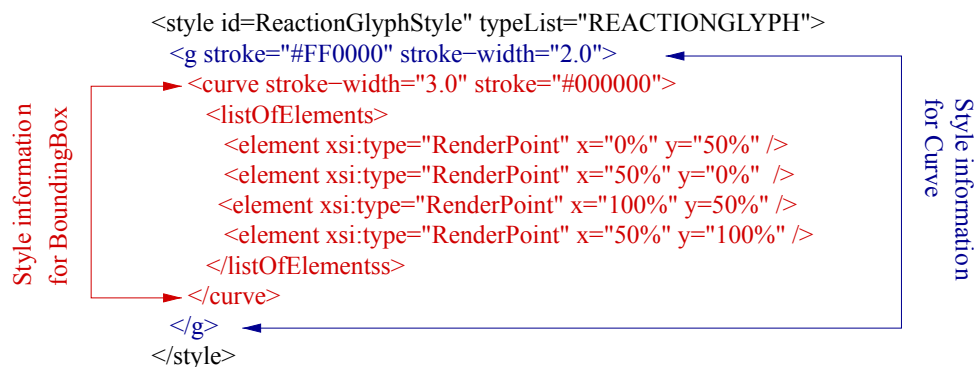


Figure 29: style with render information for objects with curve or bounding box

C.5 Style information for text glyphs

Just as in the case of curves in **ReactionGlyphs** and **SpeciesReferenceGlyphs**, **TextGlyphs** can be considered render information which is located in the layout. A **TextGlyph** specifies the text to be rendered, it therefore does not need additional render information in the form of a **text** element. On the other hand, it needs render information in the form of font properties. Just as for the **RenderCurve** object for **ReactionGlyphs** and **SpeciesReferenceGlyphs**, this render information is taken from the font related attributes of the outermost group element of the style that is used to render a **TextGlyph**. Any additional information within the group is ignored. If the group does not specify any of the **font-family**, **font-size**, **font-weight**, **font-style**, **text-anchor** or **vtext-anchor** attributes, the default values are to be used.

D Validation of SBML documents

D.1 Validation and consistency rules

This section summarizes all the conditions that must (or in some cases, at least *should*) be true of an SBML Level 3 Version 1 model that uses the Render package. We use the same conventions as are used in the SBML Level 3 Version 1 Core specification document. In particular, there are different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition *must* be true, or a rule states that it *should* be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules. To help highlight these differences, we use the following three symbols next to the rule numbers:

- ☑ A checked box indicates a *requirement* for SBML conformance. If a model does not follow this rule, it does not conform to the Render specification. (Mnemonic intention behind the choice of symbol: “This must be checked.”)
- ▲ A triangle indicates a *recommendation* for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the Render specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: “This is a cause for warning.”)
- ★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not strictly considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: “You’re a star if you heed this.”)

The validation rules listed in the following subsections are all stated or implied in the rest of this specification document. They are enumerated here for convenience. Unless explicitly stated, all validation rules concern objects and attributes specifically defined in the Render package.

- 🔍 For convenience and brevity, we use the shorthand “**render:x**” to stand for an attribute or element name **x** in the namespace for the Render package, using the namespace prefix **fb**. In reality, the prefix string may be different from the literal “**render**” used here (and indeed, it can be any valid XML namespace prefix that the modeler or software chooses). We use “**render:x**” because it is shorter than to write a full explanation everywhere we refer to an attribute or element in the Render namespace.

General rules about this package

- render-10101** ☑ To conform to the Render package specification for SBML Level 3 Version 1, an SBML document must declare the use of the following XML Namespace:
“<http://www.sbml.org/sbml/level3/version1/render/version1>”. (Reference: SBML Level 3 Package specification for Render, Version 1 [Section 3.1 on page 7.](#))
- render-10102** ☑ Wherever they appear in an SBML document, elements and attributes from the Render package must be declared either implicitly or explicitly to be in the XML namespace
“<http://www.sbml.org/sbml/level3/version1/render/version1>”.
(Reference: SBML Level 3 Package specification for Render, Version 1 [Section 3.1 on page 7.](#))

General rules about identifiers

- render-10301** ☑ (Extends validation rule #10301 in the SBML Level 3 Version 1 Core specification.) Within a **Model** the values of the **render:id** attributes of a [GlobalRenderInformation](#) object and a [LocalRenderInformation](#) object should not class with any other attribute **id** values from the global model scope Within the Render package the attribute **render:id** need only be unique for [LineEndings](#). (Reference: SBML Level 3 Package specification for Render, Version 1 [Ap](#)

[pendix 3.3 on page 10.](#))

- render-10302** ✓ The value of a **render:id** must conform to the syntax of the **SBML** data type **STid** (Reference: SBML Level 3 Package specification for Render, Version 1 [Section 3.2 on page 8.](#))

Rules for the extended **SBML** class

- render-20101** ✓ In all SBML documents using the Render package, the **SBML** object must have the **render:-required** attribute. (Reference: SBML Level 3 Version 1 Core, Section 4.1.2.)
- render-20102** ✓ The value of attribute **render:required** on the **SBML** object must be of data type **boolean**. (Reference: SBML Level 3 Version 1 Core, Section 4.1.2.)
- render-20103** ✓ The value of attribute **render:required** on the **SBML** object must be set to “**false**”. (Reference: SBML Level 3 Package specification for Render, Version 1 [Section 3.1 on page 7.](#))

Rules for extended **GraphicalObject** object

- render-20201** ✓ A **GraphicalObject** object may have the optional attribute **render:objectRole**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **GraphicalObject** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.1 on page 13.](#))
- render-20202** ✓ The attribute **render:objectRole** on a **GraphicalObject** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.1 on page 13.](#))

Rules for extended **Layout** object

- render-20301** ✓ A **Layout** object may contain one and only one instance of the **ListOfLocalRenderInformation** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **Layout** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))
- render-20302** ✓ The **ListOfLocalRenderInformation** subobject on a **Layout** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))
- render-20303** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfLocalRenderInformation** container object may only contain **LocalRenderInformation** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))
- render-20304** ✓ A **ListOfLocalRenderInformation** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfLocalRenderInformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))
- render-20305** ✓ A **ListOfLocalRenderInformation** object may have the optional attributes **render:versionMajor** and **render:versionMinor**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **ListOfLocalRenderInformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))
- render-20306** ✓ A **ListOfLocalRenderInformation** object may have the optional element **render:defaultValues**. No other elements from the SBML Level 3 Render namespaces are permitted on a **ListOfLocalRenderInformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14.](#))

- render-20307** ✓ The attribute `render:versionMajor` on a [ListOfLocalRenderInformation](#) must have a value of data type `unsigned integer`. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14](#).)
- render-20308** ✓ The attribute `render:versionMinor` on a [ListOfLocalRenderInformation](#) must have a value of data type `unsigned integer`. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.3 on page 14](#).)

Rules for extended *ListOfLayouts* object

- render-20401** ✓ A [ListOfLayouts](#) object may contain one and only one instance of the [ListOfGlobalRenderInformation](#) element. No other elements from the SBML Level 3 Render namespaces are permitted on a [ListOfLayouts](#) object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20402** ✓ The [ListOfGlobalRenderInformation](#) subobject on a [ListOfLayouts](#) object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20403** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a [ListOfGlobalRenderInformation](#) container object may only contain [GlobalRenderInformation](#) objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20404** ✓ A [ListOfGlobalRenderInformation](#) object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a [ListOfGlobalRenderInformation](#) object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20405** ✓ A [ListOfGlobalRenderInformation](#) object may have the optional attributes `render:versionMajor` and `render:versionMinor`. No other attributes from the SBML Level 3 Render namespaces are permitted on a [ListOfGlobalRenderInformation](#) object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20406** ✓ A [ListOfGlobalRenderInformation](#) object may have the optional element `render:defaultValues`. No other elements from the SBML Level 3 Render namespaces are permitted on a [ListOfGlobalRenderInformation](#) object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20407** ✓ The attribute `render:versionMajor` on a [ListOfGlobalRenderInformation](#) must have a value of data type `unsigned integer`. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)
- render-20408** ✓ The attribute `render:versionMinor` on a [ListOfGlobalRenderInformation](#) must have a value of data type `unsigned integer`. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.4.2 on page 14](#).)

Rules for *ColorDefinition* object

- render-20501** ✓ A [ColorDefinition](#) object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a [ColorDefinition](#). (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20502** ✓ A [ColorDefinition](#) object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a [ColorDefinition](#). (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

- render-20503** ✓ A **ColorDefinition** object may have the optional attributes **render:id** and **render:value**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **ColorDefinition** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.1 on page 19](#).)
- render-20504** ✓ The attribute **render:value** on a **ColorDefinition** must have a value of data type **string** where that string is restricted to either a 6 or 8 digit hex number or the value “none”. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.1 on page 19](#).)

Rules for Ellipse object

- render-20601** ✓ An **Ellipse** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on an **Ellipse**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20602** ✓ An **Ellipse** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on an **Ellipse**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20603** ✓ An **Ellipse** object must have the required attributes **render:cx**, **render:cy** and **render:rx**, and may have the optional attributes **render:cz**, **render:ry** and **render:ratio**. No other attributes from the SBML Level 3 Render namespaces are permitted on an **Ellipse** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20604** ✓ The value of the attribute **render:cx** of an **Ellipse** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20605** ✓ The value of the attribute **render:cy** of an **Ellipse** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20606** ✓ The value of the attribute **render:rx** of an **Ellipse** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20607** ✓ The value of the attribute **render:cz** of an **Ellipse** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20608** ✓ The value of the attribute **render:ry** of an **Ellipse** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)
- render-20609** ✓ The attribute **render:ratio** on an **Ellipse** must have a value of data type **double**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.3 on page 31](#).)

Rules for *GlobalRenderInformation* object

- render-20701** ✓ A **GlobalRenderInformation** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GlobalRenderInformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20702** ✓ A **GlobalRenderInformation** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GlobalRenderInformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20703** ✓ A **GlobalRenderInformation** object may contain one and only one instance of the **ListOfGlobalStyles** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **GlobalRenderInformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.3 on page 16](#).)
- render-20704** ✓ The **ListOfGlobalStyles** subobject on a **GlobalRenderInformation** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.3 on page 16](#).)
- render-20705** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfGlobalStyles** container object may only contain **GlobalStyle** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.3 on page 16](#).)
- render-20706** ✓ A **ListOfGlobalStyles** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfGlobalStyles** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.3 on page 16](#).)
- render-20707** ✓ The value of the attribute **render:referenceRenderInformation** of a **GlobalRenderInformation** object must be the identifier of an existing **GlobalRenderInformation** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15](#).)

Rules for *GlobalStyle* object

- render-20801** ✓ A **GlobalStyle** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GlobalStyle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20802** ✓ A **GlobalStyle** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GlobalStyle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

Rules for *GradientBase* object

- render-20901** ✓ A **GradientBase** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GradientBase**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20902** ✓ A **GradientBase** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GradientBase**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-20903** ✓ A **GradientBase** object must have the required attribute **render:id**, and may have the optional attribute **render:spreadMethod**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **GradientBase** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.2 on page 20](#).)

- render-20904** ✓ A **GradientBase** object must contain at least one instance of the **GradientStop** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **GradientBase** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.2 on page 20.](#))
- render-20905** ✓ The value of the attribute **render:spreadMethod** of a **GradientBase** object must conform to the syntax of SBML data type **GradientSpreadMethod** and may only take on the allowed values of **GradientSpreadMethod** defined in SBML; that is the value must be one of the following "pad", "reflect" or "repeat". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.2 on page 20.](#))

Rules for GradientStop object

- render-21001** ✓ A **GradientStop** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GradientStop**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21002** ✓ A **GradientStop** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GradientStop**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21003** ✓ A **GradientStop** object must have the required attributes **render:offset** and **render:stop-color**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **GradientStop** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.3 on page 20.](#))
- render-21004** ✓ The value of the attribute **render:offset** of a **GradientStop** object must conform to the syntax of SBML data type **RelAbsVector** but in this case can only encode a relative value i.e. a string encoding a number followed by a % sign. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.3 on page 20.](#))
- render-21005** ✓ The attribute **render:stop-color** on a **GradientStop** must have a value of data type **string** where that string is restricted to either a 6 or 8 digit hex number; the **id** of an existing **ColorDefinition** or the value "none". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.3 on page 20.](#))
- render-21006** ▲ The value of the attribute **render:offset** of a **GradientStop** object should be between "0%" and "100%". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.3 on page 20.](#))
- render-21007** ▲ The value of the attribute **render:offset** of a **GradientStop** object should be greater than or equal to the value of the **offset** attribute on the previous **GradientStop**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.3 on page 20.](#))

Rules for RenderGroup object

- render-21101** ✓ A **RenderGroup** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RenderGroup**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21102** ✓ A **RenderGroup** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RenderGroup**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21103** ✓ A **RenderGroup** object may have the optional attributes **render:startHead**, **render:endHead**, **render:font-family**, **render:font-weight**, **render:font-style**, **render:text-anchor**, **render:vtext-anchor** and **render:font-size**. No other attributes from the

SBML Level 3 Render namespaces are permitted on a **RenderGroup** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))

- render-21104** ✓ The value of the attribute **render:startHead** of a **RenderGroup** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21105** ✓ The value of the attribute **render:endHead** of a **RenderGroup** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21106** ✓ The attribute **render:font-family** on a **RenderGroup** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21107** ✓ The value of the attribute **render:font-weight** of a **RenderGroup** object must conform to the syntax of SBML data type **FontWeight** and may only take on the allowed values of **FontWeight** defined in SBML; that is the value must be one of the following "normal" or "bold". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21108** ✓ The value of the attribute **render:font-style** of a **RenderGroup** object must conform to the syntax of SBML data type **FontStyle** and may only take on the allowed values of **FontStyle** defined in SBML; that is the value must be one of the following "normal" or "italic". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21109** ✓ The value of the attribute **render:text-anchor** of a **RenderGroup** object must conform to the syntax of SBML data type **HTextAnchor** and may only take on the allowed values of **HTextAnchor** defined in SBML; that is the value must be one of the following "start", "middle" or "end". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21110** ✓ The value of the attribute **render:vtext-anchor** of a **RenderGroup** object must conform to the syntax of SBML data type **VTextAnchor** and may only take on the allowed values of **VTextAnchor** defined in SBML; that is the value must be one of the following "top", "middle", "bottom" or "baseline". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))
- render-21111** ✓ The value of the attribute **render:font-size** of a **RenderGroup** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.6 on page 33.](#))

Rules for Image object

- render-21201** ✓ An **Image** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on an **Image**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21202** ✓ An **Image** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on an **Image**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21203** ✓ An **Image** object must have the required attributes **render:x**, **render:y**, **render:width**, **render:height** and **render:href**, and may have the optional attributes **render:id** and

render:z. No other attributes from the SBML Level 3 Render namespaces are permitted on an **Image** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21204 ✓ The value of the attribute **render:x** of an **Image** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21205 ✓ The value of the attribute **render:y** of an **Image** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21206 ✓ The value of the attribute **render:width** of an **Image** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21207 ✓ The value of the attribute **render:height** of an **Image** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21208 ✓ The attribute **render:href** on an **Image** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21209 ✓ The attribute **render:href** on an **Image** must point to a local file of type “jpeg” or “png”. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

render-21210 ✓ The value of the attribute **render:z** of an **Image** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.10.5 on page 32.](#))

Rules for *LineEnding* object

render-21301 ✓ A **LineEnding** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **LineEnding**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

render-21302 ✓ A **LineEnding** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **LineEnding**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

render-21303 ✓ A **LineEnding** object must have the required attribute **render:id**, and may have the optional attribute **render:enableRotationalMapping**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **LineEnding** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.11 on page 33.](#))

- render-21304** ✓ A **LineEnding** object may contain one and only one instance of each of the **RenderGroup** and **BoundingBox** elements. No other elements from the SBML Level 3 **Render** namespaces are permitted on a **LineEnding** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.11 on page 33](#).)
- render-21305** ✓ The attribute **render:enableRotationalMapping** on a **LineEnding** must have a value of data type **boolean**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.11 on page 33](#).)

Rules for **LinearGradient** object

- render-21401** ✓ A **LinearGradient** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **LinearGradient**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21402** ✓ A **LinearGradient** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **LinearGradient**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21403** ✓ A **LinearGradient** object may have the optional attributes **render:x1**, **render:y1**, **render:z1**, **render:x2**, **render:y2** and **render:z2**. No other attributes from the SBML Level 3 **Render** namespaces are permitted on a **LinearGradient** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21404** ✓ The value of the attribute **render:x1** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21405** ✓ The value of the attribute **render:y1** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21406** ✓ The value of the attribute **render:z1** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21407** ✓ The value of the attribute **render:x2** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21408** ✓ The value of the attribute **render:y2** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22](#).)
- render-21409** ✓ The value of the attribute **render:z2** of a **LinearGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed

by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.4 on page 22.](#))

Rules for *LocalRenderInformation* object

- render-21501** ✓ A **LocalRenderInformation** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **LocalRenderInformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21502** ✓ A **LocalRenderInformation** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **LocalRenderInformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21503** ✓ A **LocalRenderInformation** object may contain one and only one instance of the **ListOfLocalStyles** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **LocalRenderInformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.2 on page 16.](#))
- render-21504** ✓ The **ListOfLocalStyles** subobject on a **LocalRenderInformation** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.2 on page 16.](#))
- render-21505** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfLocalStyles** container object may only contain **LocalStyle** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.2 on page 16.](#))
- render-21506** ✓ A **ListOfLocalStyles** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfLocalStyles** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.2 on page 16.](#))

Rules for *LocalStyle* object

- render-21601** ✓ A **LocalStyle** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **LocalStyle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21602** ✓ A **LocalStyle** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **LocalStyle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21603** ✓ A **LocalStyle** object may have the optional attribute **render:idList**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **LocalStyle** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.6.3 on page 19.](#))
- render-21604** ✓ The attribute **render:idList** on a **LocalStyle** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.6.3 on page 19.](#))

Rules for *Polygon* object

- render-21701** ✓ A **Polygon** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Polygon**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21702** ✓ A **Polygon** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Polygon**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

- render-21703** ✓ A **Polygon** object may contain one and only one instance of the **ListOfElements** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **Polygon** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.1 on page 30.](#))
- render-21704** ✓ A **Polygon** object may contain one and only one instance of the **ListOfCurveSegments** element from the Layout package. No other elements from the SBML Level 3 Layout namespaces are permitted on a **Polygon** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.1 on page 30.](#))

Rules for RadialGradient object

- render-21801** ✓ A **RadialGradient** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RadialGradient**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21802** ✓ A **RadialGradient** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RadialGradient**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21803** ✓ A **RadialGradient** object may have the optional attributes **render:cx**, **render:cy**, **render:cz**, **render:r**, **render:fx**, **render:fy** and **render:fz**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **RadialGradient** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21804** ✓ The value of the attribute **render:cx** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21805** ✓ The value of the attribute **render:cy** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21806** ✓ The value of the attribute **render:cz** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21807** ✓ The value of the attribute **render:r** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21808** ✓ The value of the attribute **render:fx** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))
- render-21809** ✓ The value of the attribute **render:fy** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed

by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))

- render-21810** ✓ The value of the attribute **render:fz** of a **RadialGradient** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.7.5 on page 22.](#))

Rules for Rectangle object

- render-21901** ✓ A **Rectangle** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Rectangle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21902** ✓ A **Rectangle** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Rectangle**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-21903** ✓ A **Rectangle** object must have the required attributes **render:x**, **render:width** and **render:-height**, and may have the optional attributes **render:y**, **render:z**, **render:rX**, **render:rY** and **render:ratio**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **Rectangle** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))
- render-21904** ✓ The value of the attribute **render:x** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))
- render-21905** ✓ The value of the attribute **render:width** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))
- render-21906** ✓ The value of the attribute **render:height** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))
- render-21907** ✓ The value of the attribute **render:y** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))
- render-21908** ✓ The value of the attribute **render:z** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))

render-21909 ✓ The value of the attribute **render:rX** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))

render-21910 ✓ The value of the attribute **render:rY** of a **Rectangle** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))

render-21911 ✓ The attribute **render:ratio** on a **Rectangle** must have a value of data type **double**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.2 on page 30.](#))

Rules for **RenderCubicBezier** object

render-22001 ✓ A **RenderCubicBezier** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RenderCubicBezier**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

render-22002 ✓ A **RenderCubicBezier** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RenderCubicBezier**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

render-22003 ✓ A **RenderCubicBezier** object must have the required attributes **render:basepoint1_x**, **render:basepoint1_y**, **render:basepoint2_x** and **render:basepoint2_y**, and may have the optional attributes **render:basepoint1_z** and **render:basepoint2_z**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **RenderCubicBezier** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

render-22004 ✓ The value of the attribute **render:basepoint1_x** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

render-22005 ✓ The value of the attribute **render:basepoint1_y** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

render-22006 ✓ The value of the attribute **render:basepoint2_x** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

render-22007 ✓ The value of the attribute **render:basepoint2_y** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

- render-22008** ✓ The value of the attribute **render:basepoint1_z** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))
- render-22009** ✓ The value of the attribute **render:basepoint2_z** of a **RenderCubicBezier** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.5 on page 27.](#))

Rules for **RenderCurve** object

- render-22101** ✓ A **RenderCurve** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RenderCurve**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22102** ✓ A **RenderCurve** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RenderCurve**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22103** ✓ A **RenderCurve** object may have the optional attributes **render:startHead** and **render:endHead**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **RenderCurve** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-22104** ✓ A **RenderCurve** object may contain one and only one instance of the **ListOfElements** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **RenderCurve** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-22104** ✓ A **RenderCurve** object may contain one and only one instance of the **ListOfCurveSegments** element from the Layout package. No other elements from the SBML Level 3 Layout namespaces are permitted on a **RenderCurve** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-22105** ✓ The value of the attribute **render:startHead** of a **RenderCurve** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-22106** ✓ The value of the attribute **render:endHead** of a **RenderCurve** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))

Rules for **RenderPoint** object

- render-22201** ✓ A **RenderPoint** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RenderPoint**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22202** ✓ A **RenderPoint** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RenderPoint**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22203** ✓ A **RenderPoint** object must have the required attributes **render:x** and **render:y**, and may have the optional attribute **render:z**. No other attributes from the SBML Level 3 Render

namespaces are permitted on a **RenderPoint** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.4 on page 27.](#))

- render-22204** ✓ The value of the attribute **render:x** of a **RenderPoint** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.4 on page 27.](#))
- render-22205** ✓ The value of the attribute **render:y** of a **RenderPoint** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.4 on page 27.](#))
- render-22206** ✓ The value of the attribute **render:z** of a **RenderPoint** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.9.4 on page 27.](#))

Rules for Text object

- render-22301** ✓ A **Text** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Text**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22302** ✓ A **Text** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Text**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22303** ✓ A **Text** object must have the required attributes **render:x** and **render:y**, and may have the optional attributes **render:z**, **render:font-family**, **render:font-weight**, **render:font-style**, **render:text-anchor**, **render:vtext-anchor** and **render:font-size**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **Text** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22304** ✓ The value of the attribute **render:x** of a **Text** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22305** ✓ The value of the attribute **render:y** of a **Text** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22306** ✓ The value of the attribute **render:z** of a **Text** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))

- render-22307** ✓ The value of the attribute **render:font-family** of a **Text** object must conform to the syntax of SBML data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22308** ✓ The value of the attribute **render:font-weight** of a **Text** object must conform to the syntax of SBML data type **FontWeight** and may only take on the allowed values of **FontWeight** defined in SBML; that is the value must be one of the following "normal" or "bold". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22309** ✓ The value of the attribute **render:font-style** of a **Text** object must conform to the syntax of SBML data type **FontStyle** and may only take on the allowed values of **FontStyle** defined in SBML; that is the value must be one of the following "normal" or "italic". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22310** ✓ The value of the attribute **render:text-anchor** of a **Text** object must conform to the syntax of SBML data type **HTextAnchor** and may only take on the allowed values of **HTextAnchor** defined in SBML; that is the value must be one of the following "start", "middle" or "end". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22311** ✓ The value of the attribute **render:vtext-anchor** of a **Text** object must conform to the syntax of SBML data type **VTextAnchor** and may only take on the allowed values of **VTextAnchor** defined in SBML; that is the value must be one of the following "top", "middle", "bottom" or "baseline". (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))
- render-22312** ✓ The value of the attribute **render:font-size** of a **Text** object must conform to the syntax of SBML data type **RelAbsVector** but in this case is restricted to being either an absolute or relative value i.e. a string encoding either an absolute number or a relative number followed by a % sign. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.10.4 on page 31.](#))

Rules for Transformation2D object

- render-22401** ✓ A **Transformation2D** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Transformation2D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22402** ✓ A **Transformation2D** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Transformation2D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22403** ✓ A **Transformation2D** object may have the optional attribute **render:transform**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **Transformation2D** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.8.2 on page 24.](#))
- render-22404** ✓ The value of the attribute **render:transform** of a **Transformation2D** object must be an array of values of type **double**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.8.2 on page 24.](#))

Rules for Transformation object

- render-22501** ✓ A **Transformation** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Transformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22502** ✓ A **Transformation** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Transformation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

- render-22503** ✓ A **Transformation** object must have the required attribute **render:transform**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **Transformation** object. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.8.1 on page 23.](#))
- render-22504** ✓ The value of the attribute **render:transform** of a **Transformation** object must be an array of values of type **double**. (Reference: SBML Level 3 Specification for Render Version 1, [Section 3.8.1 on page 23.](#))

Rules for GraphicalPrimitive1D object

- render-22601** ✓ A **GraphicalPrimitive1D** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GraphicalPrimitive1D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22602** ✓ A **GraphicalPrimitive1D** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GraphicalPrimitive1D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22603** ✓ A **GraphicalPrimitive1D** object may have the optional attributes **render:id**, **render:stroke**, **render:stroke-width** and **render:stroke-dasharray**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **GraphicalPrimitive1D** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.1 on page 25.](#))
- render-22604** ✓ The attribute **render:stroke** on a **GraphicalPrimitive1D** must have a value of data type **string** where that string is restricted to either a 6 or 8 digit hex number or the **id** of an existing **ColorDefinition**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.1 on page 25.](#))
- render-22605** ✓ The attribute **render:stroke-width** on a **GraphicalPrimitive1D** must have a value of data type **double**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.1 on page 25.](#))
- render-22606** ✓ The value of the attribute **render:stroke-dasharray** of a **GraphicalPrimitive1D** object must be an array of values of type **unsigned integer**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.1 on page 25.](#))

Rules for GraphicalPrimitive2D object

- render-22701** ✓ A **GraphicalPrimitive2D** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GraphicalPrimitive2D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22702** ✓ A **GraphicalPrimitive2D** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GraphicalPrimitive2D**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22703** ✓ A **GraphicalPrimitive2D** object may have the optional attributes **render:fill** and **render:-fill-rule**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **GraphicalPrimitive2D** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.2 on page 25.](#))
- render-22704** ✓ The attribute **render:fill** on a **GraphicalPrimitive2D** must have a value of data type **string** where that string is restricted to either a 6 or 8 digit hex number; the **id** of an existing **ColorDefinition** or the value “none”. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.2 on page 25.](#))

- render-22705** ✓ The value of the attribute `render:fill-rule` of a **GraphicalPrimitive2D** object must conform to the syntax of SBML data type **FillRule** and may only take on the allowed values of **FillRule** defined in SBML; that is the value must be one of the following "nonzero" or "evenodd". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.2 on page 25.](#))

Rules for Style object

- render-22801** ✓ A **Style** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Style**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22802** ✓ A **Style** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Style**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22803** ✓ A **Style** object may have the optional attributes `render:id`, `render:name`, `render:roleList` and `render:typeList`. No other attributes from the SBML Level 3 Render namespaces are permitted on a **Style** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.6.1 on page 18.](#))
- render-22804** ✓ A **Style** object may contain one and only one instance of the **RenderGroup** element. No other elements from the SBML Level 3 Render namespaces are permitted on a **Style** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.6.1 on page 18.](#))
- render-22805** ✓ The attribute `render:name` on a **Style** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.6.1 on page 18.](#))
- render-22806** ✓ The attribute `render:roleList` on a **Style** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.6.1 on page 18.](#))
- render-22807** ✓ The attribute `render:typeList` on a **Style** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.6.1 on page 18.](#))

Rules for RenderInformationBase object

- render-22901** ✓ A **RenderInformationBase** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **RenderInformationBase**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22902** ✓ A **RenderInformationBase** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **RenderInformationBase**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-22903** ✓ A **RenderInformationBase** object must have the required attribute `render:id`, and may have the optional attributes `render:name`, `render:programName`, `render:programVersion`, `render:referenceRenderInformation` and `render:backgroundColor`. No other attributes from the SBML Level 3 Render namespaces are permitted on a **RenderInformationBase** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22904** ✓ A **RenderInformationBase** object may contain one and only one instance of each of the **ListOfColorDefinitions**, **ListOfGradientDefinitions** and **ListOfLineEndings** elements. No other elements from the SBML Level 3 Render namespaces are permitted on a **RenderInformationBase** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))

- render-22905** ✓ The attribute **render:name** on a **RenderInformationBase** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22906** ✓ The attribute **render:programName** on a **RenderInformationBase** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22907** ✓ The attribute **render:programVersion** on a **RenderInformationBase** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22908** ✓ The value of the attribute **render:referenceRenderInformation** of a **RenderInformationBase** object must be the identifier of an existing **RenderInformation** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22909** ✓ The attribute **render:backgroundColor** on a **RenderInformationBase** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22910** ✓ The **ListOfColorDefinitions**, **ListOfGradientDefinitions** and **ListOfLineEndings** subobjects on a **RenderInformationBase** object is optional, but if present, these container objects must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22911** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfColorDefinitions** container object may only contain **ColorDefinition** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22912** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfGradientDefinitions** container object may only contain **GradientBase** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22913** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfLineEndings** container object may only contain **LineEnding** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22914** ✓ A **ListOfColorDefinitions** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfColorDefinitions** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22915** ✓ A **ListOfGradientDefinitions** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfGradientDefinitions** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))
- render-22916** ✓ A **ListOfLineEndings** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfLineEndings** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.5.1 on page 15.](#))

Rules for DefaultValues object

- render-23001** ✓ A **DefaultValues** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **DefaultValues**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-23002** ✓ A **DefaultValues** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **DefaultValues**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)
- render-23003** ✓ A **DefaultValues** object may have the optional attributes **render:backgroundColor**, **render:-spreadMethod**, **render:linearGradient_x1**, **render:linearGradient_y1**, **render:linearGradient_z1**, **render:linearGradient_x2**, **render:linearGradient_y2**, **render:linearGradient_z2**, **render:radialGradient_cx**, **render:radialGradient_cy**, **render:radialGradient_cz**, **render:radialGradient_r**, **render:radialGradient_fx**, **render:radialGradient_fy**, **render:radialGradient_fz**, **render:fill-rule**, **render:-default_z**, **render:stroke**, **render:stroke-width**, **render:font-family**, **render:font-size**, **render:font-weight**, **render:font-style**, **render:text-anchor**, **render:vtext-anchor**, **render:startHead** and **render:endHead**. No other attributes from the SBML Level 3 Render namespaces are permitted on a **DefaultValues** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23004** ✓ The attribute **render:backgroundColor** on a **DefaultValues** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23005** ✓ The value of the attribute **render:spreadMethod** of a **DefaultValues** object must conform to the syntax of SBML data type **GradientSpreadMethod** and may only take on the allowed values of **GradientSpreadMethod** defined in SBML; that is the value must be one of the following "pad", "reflect" or "repeat". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23006** ✓ The value of the attribute **render:linearGradient_x1** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23007** ✓ The value of the attribute **render:linearGradient_y1** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23008** ✓ The value of the attribute **render:linearGradient_z1** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23009** ✓ The value of the attribute **render:linearGradient_x2** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))

- render-23010** ✓ The value of the attribute `render:linearGradient_y2` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23011** ✓ The value of the attribute `render:linearGradient_z2` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23012** ✓ The value of the attribute `render:radialGradient_cx` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23013** ✓ The value of the attribute `render:radialGradient_cy` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23014** ✓ The value of the attribute `render:radialGradient_cz` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23015** ✓ The value of the attribute `render:radialGradient_r` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23016** ✓ The value of the attribute `render:radialGradient_fx` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23017** ✓ The value of the attribute `render:radialGradient_fy` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23018** ✓ The value of the attribute `render:radialGradient_fz` of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))

- render-23019** ✓ The value of the attribute **render:fill-rule** of a **DefaultValues** object must conform to the syntax of SBML data type **FillRule** and may only take on the allowed values of **FillRule** defined in SBML; that is the value must be one of the following "nonzero" or "evenodd". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23020** ✓ The value of the attribute **render:default_z** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23021** ✓ The attribute **render:stroke** on a **DefaultValues** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23022** ✓ The attribute **render:stroke-width** on a **DefaultValues** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23023** ✓ The attribute **render:font-family** on a **DefaultValues** must have a value of data type **string**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23024** ✓ The value of the attribute **render:font-size** of a **DefaultValues** object must conform to the syntax of SBML data type **RelAbsVector** i.e. a string encoding optionally an absolute number followed by an optional relative number followed by a % sign. Adding spaces between the coordinates is encouraged, but not required. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23025** ✓ The value of the attribute **render:font-weight** of a **DefaultValues** object must conform to the syntax of SBML data type **FontWeight** and may only take on the allowed values of **FontWeight** defined in SBML; that is the value must be one of the following "normal" or "bold". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23026** ✓ The value of the attribute **render:font-style** of a **DefaultValues** object must conform to the syntax of SBML data type **FontStyle** and may only take on the allowed values of **FontStyle** defined in SBML; that is the value must be one of the following "normal" or "italic". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23027** ✓ The value of the attribute **render:text-anchor** of a **DefaultValues** object must conform to the syntax of SBML data type **HTextAnchor** and may only take on the allowed values of **HTextAnchor** defined in SBML; that is the value must be one of the following "start", "middle" or "end". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23028** ✓ The value of the attribute **render:vtext-anchor** of a **DefaultValues** object must conform to the syntax of SBML data type **VTextAnchor** and may only take on the allowed values of **VTextAnchor** defined in SBML; that is the value must be one of the following "top", "middle", "bottom" or "baseline". (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23029** ✓ The value of the attribute **render:startHead** of a **DefaultValues** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))
- render-23030** ✓ The value of the attribute **render:endHead** of a **DefaultValues** object must be the identifier of an existing **LineEnding** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.3.2 on page 12.](#))

Rules for ListOfElements object

- render-23040** ✓ The **ListOfElements** subobject on a **RenderCurve** or a **Polygon** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-23041** ✓ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfElements** container object may only contain **RenderPoint** or the derived **RenderCubicBezier** objects. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-23042** ✓ A **ListOfElements** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfElements** object. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))
- render-23043** ✓ The first element within a **ListOfElements** container object must be of type **RenderPoint**. (Reference: SBML Level 3 Specification for Render Version 1, [Appendix 3.9.3 on page 26.](#))

References

Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at <http://www.w3.org/TR/xmlschema-2/>.

Eriksson, H.-E. and Penker, M. (1998). *UML Toolkit*. John Wiley & Sons, New York.

Fallside, D. C. (2000). XML Schema part 0: Primer (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at <http://www.w3.org/TR/xmlschema-0/>.

Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.

SBML Team (2010). The SBML Issue Tracker. Available via the World Wide Web at <http://sbml.org/issue-tracker>.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures (W3C candidate recommendation 24 October 2000). Available online via the World Wide Web at the address <http://www.w3.org/TR/xmlschema-1/>.