deegree Charts

deegree v2.5

lat/lon GmbH
Aennchenstr. 19
53177 Bonn
Germany
Tel ++49 - 228 - 184 96-0
Fax ++49 - 228 - 184 96-29
info@lat-lon.de
www.lat-lon.de

Dept. of Geography
Bonn University
Meckenheimer Allee 166
53115 Bonn

Tel. ++49 228 732098

Change log

Date	***Description***	***Author***
2008-05-02	Writing the documentation	Moataz Elmasry
2008-05-05	Corrections and reformatting	Andreas Poth
2008-05-07	Chapter 5 added	Andreas Poth
2008-05-08	Proof-reading	Markus Lupp
2008-06-05	Added documentation for the new parameter “style”	Moataz Elmasry
2008-08-14	Bug fix in Appendix A	Andreas Poth
2009-08-05	Check proper writing for name of deegree Charts	Judit Mays
2009-08-28	Adjust appendixes to demo release settings	Judit Mays
2009-10-05	Bugfix in chart configuration	Judit Mays

Table of Contents

Illustration Index

1 Introduction

deegree is a Java Framework offering the main building blocks for Spatial Data Infrastructure (SDIs). Its entire architecture is developed using standards of the Open Geospatial Consortium (OGC) and ISO Technical Committee 211 - Geographic information / Geoinformatics (ISO/TC 211). deegree encompasses OGC Web Services as well as clients. deegree is Free Software protected by the GNU Lesser General Public License (GNU LGPL) and is accessible at www.deegree.org

deegree Charts is a deegree web application designed for creating chart figures. The user is able to generate these charts by sending HTTP requests through the browser to the chart servlet and receives as a result images with the requested charts. This servlet can be used for displaying charts as symbols in a WMS using SLD or as a standalone application to display chart images for a given dataset. Chart type, appearance and data sources can all be determined using the given parameters of that request. In the next pages we will discuss how to install and configure the servlet and how to use it to generate different types and shapes of charts.

Besides deegree Charts, deegree comprises a number of additional services and clients. A complete list of deegree components can be found using the following URL and links:

`http://www.lat-lon.de` → Products

Downloads of packaged deegree components can be found via:

`http://www.deegree.org` → Download

2 Download/ Installation

2.1 Prerequisites

To run deegree Charts you need:

- Java (JRE or JSDK) version 1.5.x
- Tomcat 5.5.x

For installation of these components refer to the corresponding documentation at java.sun.com and tomcat.apache.org respectively.

deegree Charts is already a part of the deegree framework, so you need a recent version of deegree jar. You also need the other jars that the deegree framework is using, which can be checked out from the SVN[1] under http://wald.intevation.org/projects/deegree and should include jfreechart.jar.

[1] SVN is a subversion Framework (revision control system), that allows multiple programmers to work on the same project and the same code and synchronize their work together without running into code conflict. If you want to know more about SNVs try using the svn tool from http://subclipse.tigris.org/

3 Configuration

A servlet is an object that is hosted by a servlet container (like tomcat). It receives requests through HTTP protocol (e.g. send by a browser) and creates a proper response according to that request; just like any computer program that receives an input and produces an output. The main purpose of servlets as objects is that they are mainly designed to be accessed remotely.

3.1 Servlet Configuration

To use the deegree Charts servlet we will need to create a new web application. Create a folder with any name, this will act as the main project folder. In this folder create another folder called “WEB-INF”, which will hold the servlet information and inside it shall exist the xml file “web.xml” (deployment descriptor) and the “lib” folder. The lib folder contains all the needed libraries to run the servlet. In the “web.xml” create a `<servlet>` tag with the servlet class org.deegree.enterprise.servlet.ChartServlet. Also add two init-param elements. The first is “errorMsg” which shall hold the path of an error image, in case the chart could not be created. The second parameter is “configFile” which is the input xml configuration of the servlet. With it you can control how the output chart should look. The configuration file structure will be explained in section 3.2. Please consult Appendix A for a web.xml example that you can use directly in your web application (do not forget to update the paths of your init-param elements according to your configuration)

3.2 Tomcat Configuration

Apache Tomcat is a web container, or application server developed at the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Tomcat includes its own internal HTTP server.

After Downloading and unpacking tomcat, please read the tomcat documentation on the apache website and install tomcat properly. After installing tomcat you need to create our context in tomcat in order to run our servlet, you need to do the following.

1. In “tomcat/conf” folder you need to create a folder with the path “./Catalina/localhost”
2. under “tomcat/conf/Catalina/localhost” create an xml file with any name, which will act as the web context.
3. The content of this xml file is as follows:

```
<Context displayName="charts" docBase="my_webapp_path" ></Context>
```

Where the value of the attribute docBase is the absolute path to your web application folder as explained in section 3.1

4. To start tomcat use your command line tool and run the batch file "./bin/startup.bat" in tomcat directory

By this we are done with tomcat configuration.

3.3 Chart configuration

An example configuration xml file is as follows:

```
<dg:ChartConfiguration xmlns:dg="http://www.deegree.org/charts"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.deegree.org/charts
http://schemas.deegree.org/charts/1.0.0/charts_configuration.xsd">
   <dg:GeneralChartSettings>
      <dg:Antialiasing>true</dg:Antialiasing>
      <dg:TextAntialiasing>true</dg:TextAntialiasing>
      <dg:BorderVisibility>false</dg:BorderVisibility>
      <dg:RectangleInsets>0,0,0,0</dg:RectangleInsets>
      <dg:BackgroundColor>0xffffff</dg:BackgroundColor>
      <dg:BackgroundOpacity>0</dg:BackgroundOpacity>
      <dg:ForegroundOpacity>1</dg:ForegroundOpacity>
      <dg:FontFamily>ARIAL</dg:FontFamily>
      <dg:FontType>PLAIN</dg:FontType>
      <dg:FontSize>12</dg:FontSize>
   </dg:GeneralChartSettings>
   <dg:GeneralPlotSettings>
      <dg:ForegroundOpacity>0.8</dg:ForegroundOpacity>
      <dg:BackgroundColor>0xffffff</dg:BackgroundColor>
      <dg:BackgroundOpacity>0</dg:BackgroundOpacity>
      <dg:OutlineVisibility>false</dg:OutlineVisibility>
   </dg:GeneralPlotSettings>
   <dg:PiePlotSettings>
      <dg:InteriorGap>0.01</dg:InteriorGap>
      <dg:LabelGap>0.01</dg:LabelGap>
      <dg:Circular>true</dg:Circular>
      <dg:BaseSectionColor>0xFFFFFF</dg:BaseSectionColor>
      <dg:BaseSectionOpacity>0</dg:BaseSectionOpacity>
      <dg:ShadowColor>0xDDDDDD</dg:ShadowColor>
      <dg:ShadowOpacity>0</dg:ShadowOpacity>
   </dg:PiePlotSettings>
   <dg:LinePlotSettings>
      <dg:RenderLines>true</dg:RenderLines>
      <dg:RenderShapes>true</dg:RenderShapes>
   </dg:LinePlotSettings>
</dg:ChartConfiguration>
```

The configuration xml file consists of four main sections. The first section is GeneralChartSettings, which holds the general configurations for the output chart image. A chart image mainly contains two parts; outer and inner area (Figure 1). The inner area is the actual drawing area for data (plot), while the outer area is an extra space to render labels and legend.

Configurable elements of a chart are:

- Antialiasing: must be true if lines of a chart (not the plot!) shall be rendered using anti aliasing to create a softer visualization.

- TextAntialiasing: must be true if chart texts (e.g. title) shall be rendered using antialiasing to create a softer visualization.
- BorderVisibility: must be true if chart borders should be rendered
- RectangleInsets: This defines the spaces (west, north, east, south) between a chart border and the contained plot in pixel.
- BackgroundColor: background colour of a chart (not of a plot!).
- BackgroundOpacity: opacity of a chart background. Possible range of values is 0..1. A value of 0 will create a 100% transparent background.
- FontFamily: font family name like ARIAL, TIMES, SERIF, ... to be used for rendering chart text elements like title.
- FontType: type of font chart rendering. Possible values are PLAIN, BOLD and ITALIC.
- FontSize: size of font in pixel used for chart rendering.

The second section is GeneralPlotSettings, which holds the configuration for the actual chart(plot). It controls fore- and background colour and their respective opacity.

Configurable elements of a chart are:

- ForegroundOpacity: opacity of a plot elements like bars, lines an pie pieces. Possible range of values is 0..1.
- BackgroundColor: It is similar to char background color but instead of the chart the color of the background of the plot itself is defined.
- BackgroundOpacity: opacity of a chart background. Possible range of values is 0..1.
- OutlineVisibility: must be true if outlines of a plot should be rendered.

Third section is the PiePlotSettings, which holds configurations for the charts of type Pie and Pie3D (Figure 2 and Figure 3).

Configurable elements of a chart are:

- InteriorGap: is the percentual space between a pie and a plots border. Possible range of values is 0..1.
- LabelGap: is the percentual space between a pie piece label and the pie.
- Circular: must be true if a pie shall be a circle ignoring the charts width/height
- BaseSectionColor: colour of a pie background (not the plot background). This colour has just en effect if the plots ForegroundOpacity is < 1.

- BaseSectionOpacity: opacity of a pies base section. Possible range of values is 0..1.
- ShadowColor: a pie always has shadow, this element defines its colour.
- ShadowOpacity: opacity of a pies shadow. Possible range of values is 0..1. If you do not want a shadow set this element to 0.

Fourth section LinePlotSettings holds information to the chart of type Line.

- RenderLines: must be true if lines shall be rendered
- RenderShapes: must be true is shapes at points shall be rendered representing the data passed to a line plot (Figure 8).

The next figure shows the most important elements of the configuration xml file and what parts of the chart do they effect

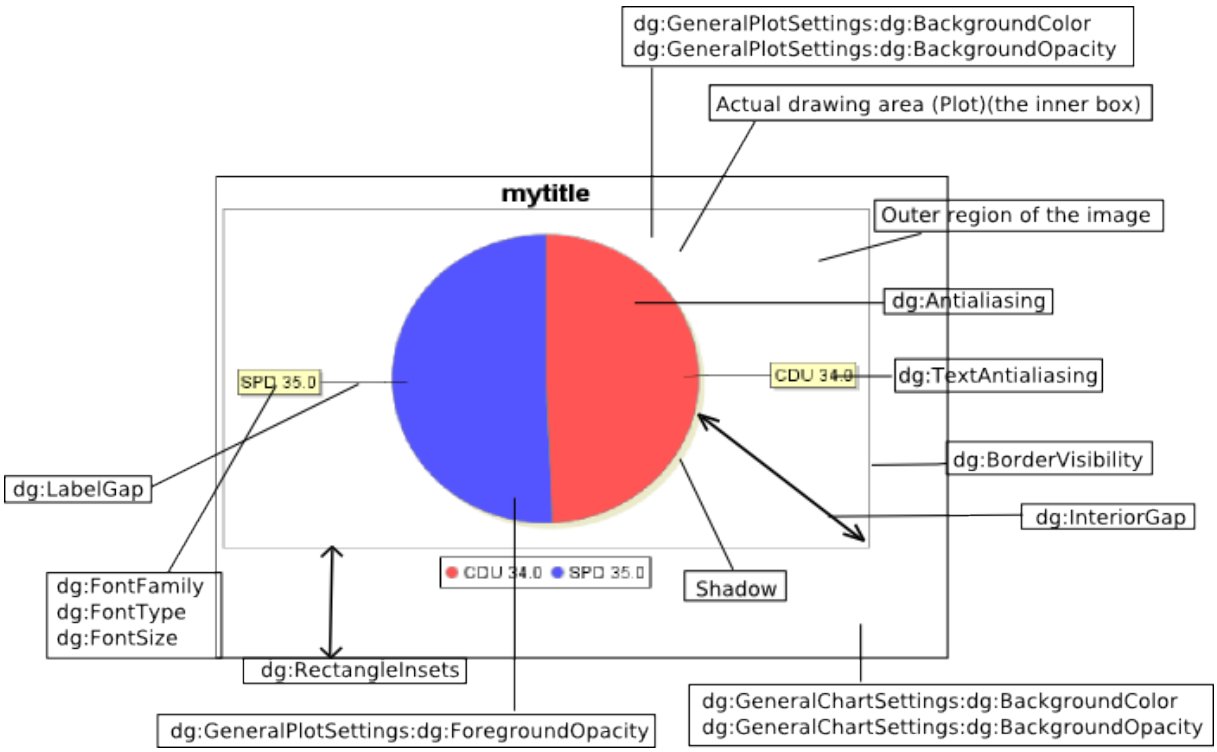


Figure 1: Main configuration elements and the regions they effect

4 Using deegree Charts

The deegree Charts servlet can be used via HTTP GET request through the browser.

```
The request should look as follows:
   http://www.someserver.com/servletname?
      chart=ChartType&
      title=string&
      legend=boolean&
      width=int&
      height=int&
      xAxis=string&
      yAxis=string&
      tooltips=boolean&
      lblType=label_type&
      orientation=[vertical | horizontal]&
      format=image_format&
       style=configurationFile
      $value$=$concrete values$
```

Make sure that the $value$ format is in accordance with the chart type (it can have different formats that depend on the chart type).

4.1 Chart Parameter

4.1.1 Available Chart Types

- Pie renders the given input on a pie, with each input taking a section (slice) of the pie. As values it accepts only keys that have one value. For example CDU=34&SPD=35

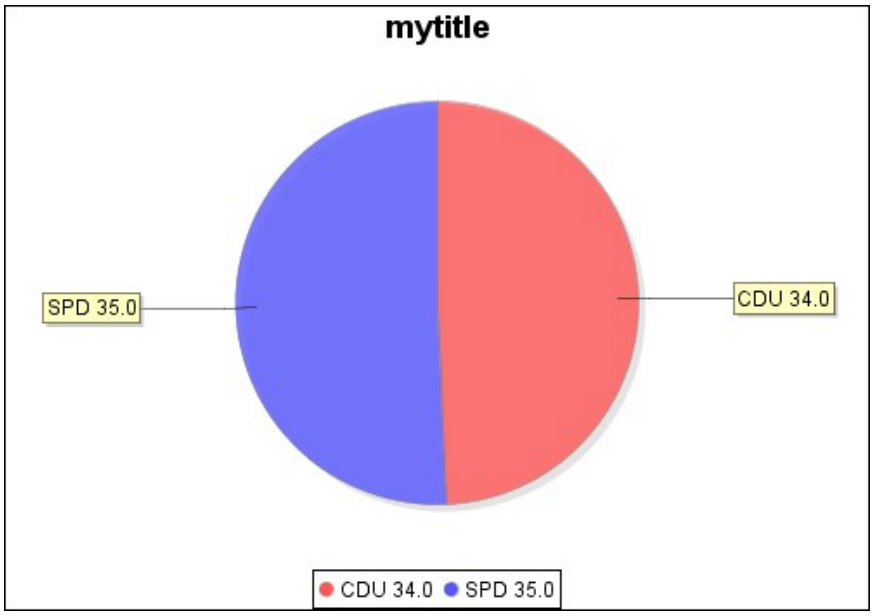


Figure 2: Pie chart

- Pie3D Same as Pie with a 3D Effect

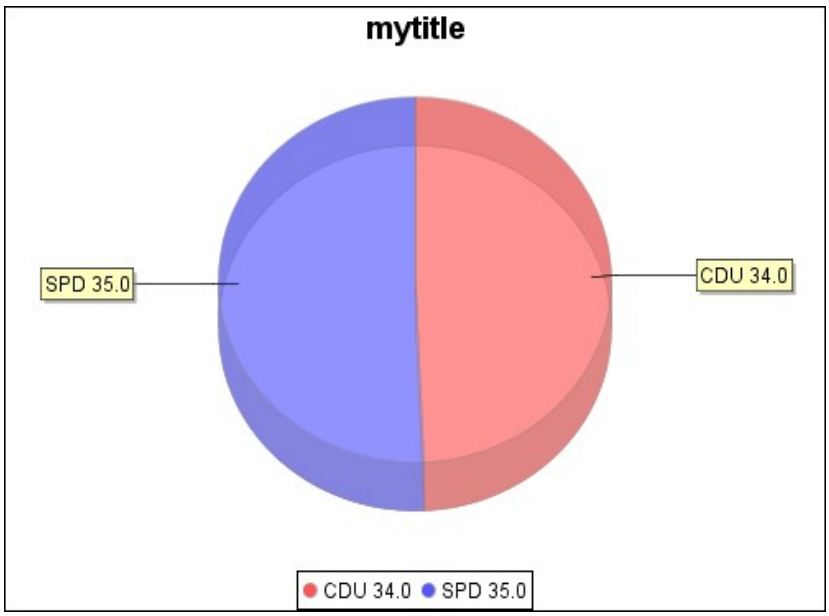


Figure 3: 3D Pie chart

- Bars renders the result as bars instead of a pie. It can accept either single values as in Pie Chart or multiple values separated by comma, so as to have multiple values for the same category, for example CDU=34,23&SPD=35,12. Each category (ex. CDU will be rendered with a different colour); but make sure that the number of parameters of each category is equal, i.e. If CDU has two parameters, then shall SPD have two parameters as well

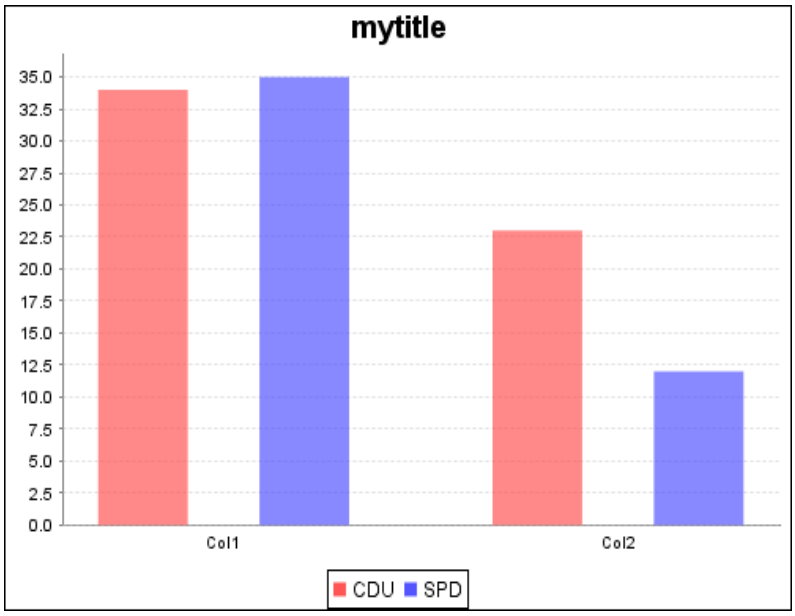
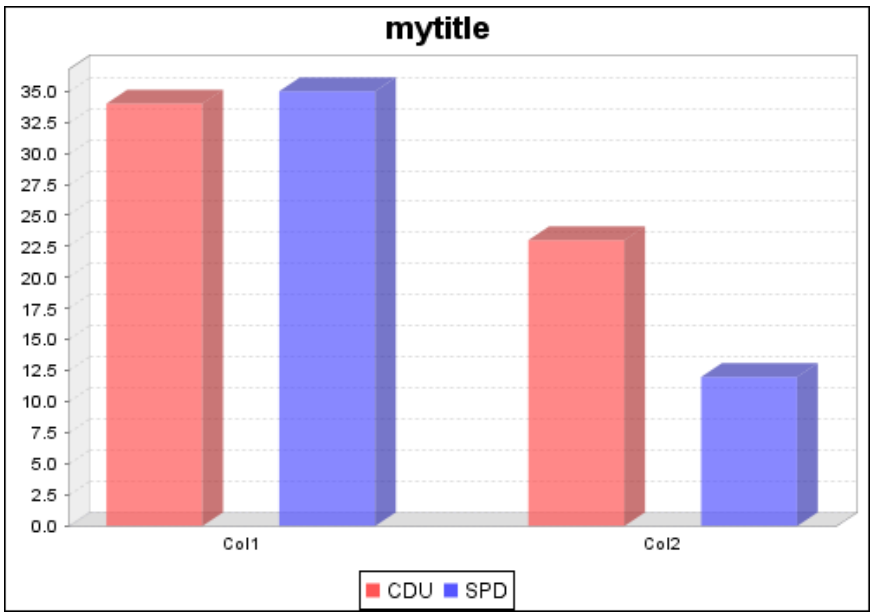
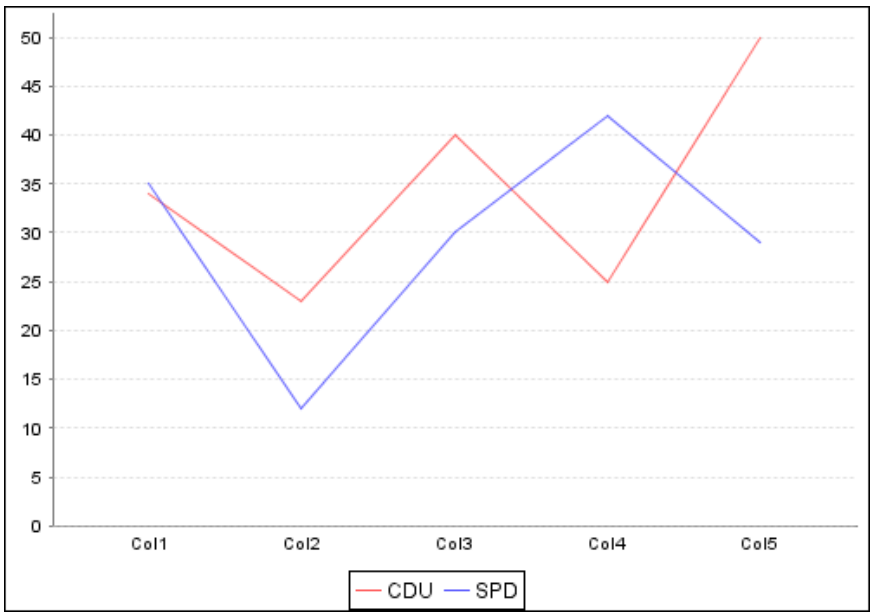


Figure 4: Bar chart

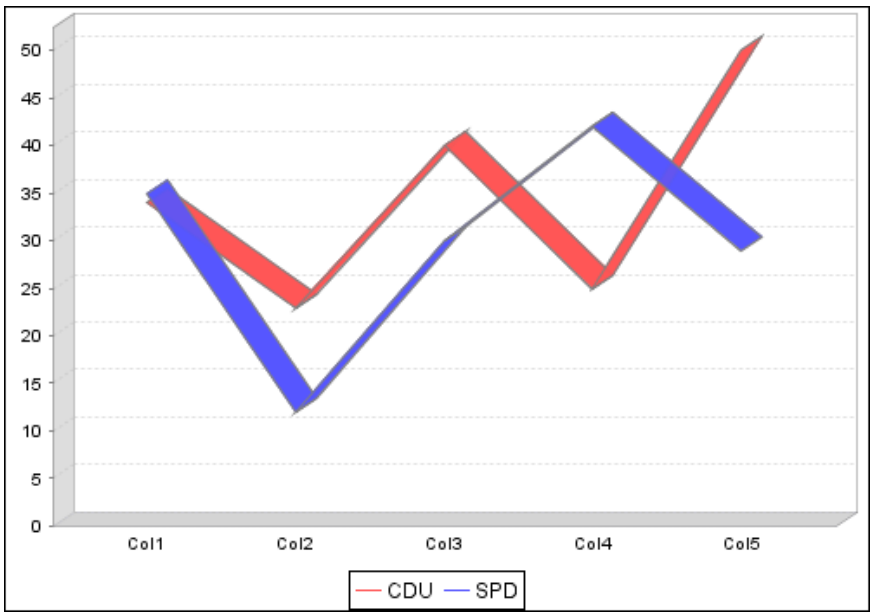
- Bar3D Same as Bar Chart but with 3D Effect



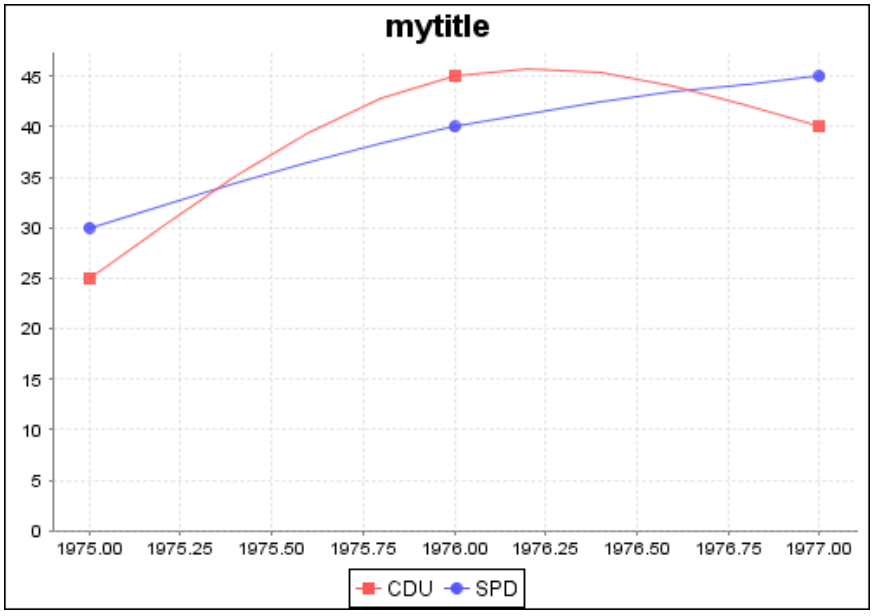
- Line This type is used to draw Line charts. Each input parameter(category) should contain multiple values separated by comma, ex: CDU=34,23,40,25,50&SPD=35,12,30,42,29. Each category (ex. CDU) will be rendered with a different colour; but make sure that the number of parameters of each category is equal, i.e. If CDU has two parameters, then shall SPD have two parameters as well.



- Line3D Same as Line but with 3D effect



- XYLine. A line chart that accepts multiple values separated by comma. But XY here means that each value (tupel) consists of two tokens (X and Y hence). That means in each category there are multiple points, and each point has x and y coordinates. Different formats are explained in detail under the parameter “type”. Each category (ex. CDU) will be rendered with a different color; make sure that the number of parameters of every category is the same, i.e. if CDU has two parameters, then SPD needs two parameters as well



4.1.2 Further Chart Types

The available chart types only represent a subset of possible chart types. The type of chart is only restricted by what is available through the JFreeChart library, as deegree Chart depends on it.

If you need to support for a new kind of chart, contact the deegree users list and ask for future implementation plans. If you are able to implement support for a new type of chart yourself, contact the deegree developers list, to let your code be integrated into deegree.

4.2 Other Parameters

1. title: Title of the generated image (optional)
2. legend: true/false to show/hide the legend of the categories/rows (optional)
3. width: Width of the generated image (mandatory)
4. height: Height of the generated image (mandatory)
5. xAxis: Title of the x Axis (optional)
6. yAxis: Title of the Y Axis (optional)
7. tooltips: true/false to Show/hide tool tips (optional).
8. lblType Type of the label. This parameter shall be used only in case the chart type equals Pie or Pie3D. The value of this parameter is either “Key”, “Value” or “KeyValue” which displays the key, value or key concatenated with value respectively.
9. orientation: possible values vertical/horizontal. The orientation of the chart (optional and default=vertical)
10. format: format of the output image, ex: image/png and image/jpeg
11. style: path to a configuration file to have specific configuration to the built chart. Typical structure of this file is the same as the default Configuration of the deegree Chart (section 3.3) with different values of course. If the address is a web address then it should either start with http:// or ftp://; if it's a file path to a local file then it should start with file:/. This parameter is optional. Ex style=http://www.myserver.com/configs.xml or style=file:/C:/configs.xml
12. $values$: This is a place holder for parameter(s) and parameter values that carry the actual values for the chart.

    $values$ can have three different formats. But only one format type shall be used per request. The values can be given as normal key value pairs as follows:

- Format 1:
  &CDU=34&SPD=36&Gruene=11

  In this format the key has only one value. This type can only be used together with the chart type: Pie, Pie3D, Bar, Line and Line3D
- Format 2:
  &CDU=23,25,21,26&SPD=42,23,33,36

  In this format multiple values per keys can be given Either comma or white spaces can be used as separators(white spaces have to be URL-encoded in case of a Http Get Request) This type can only be used together with the chart type: Bar, Line and Line3D
- Format 3:
  &CDU=1970,23;1974,44;1978,43&SPD=1970,23;1974,44;1978,43

  In this format the key has multiple value, in which each value is a tuple that contains in turn two tokens representing x and y of a Cartesian point. Commas and white spaces(encoded with %20) can be used as separators between tokens, i.e. between x and y, while semi colon shall be used as a separator between tuples. This type can only be used together with XYLine (and later with XYBar)

4.3 HTTP request examples

4.3.1 Format One

```
http://localhost:8080/contextName/charts?
chart=Pie&title=mytitle&legend=true&width=500&height=350&format=image/png&CDU=34&SP
D=35&lblType=KeyValue
```

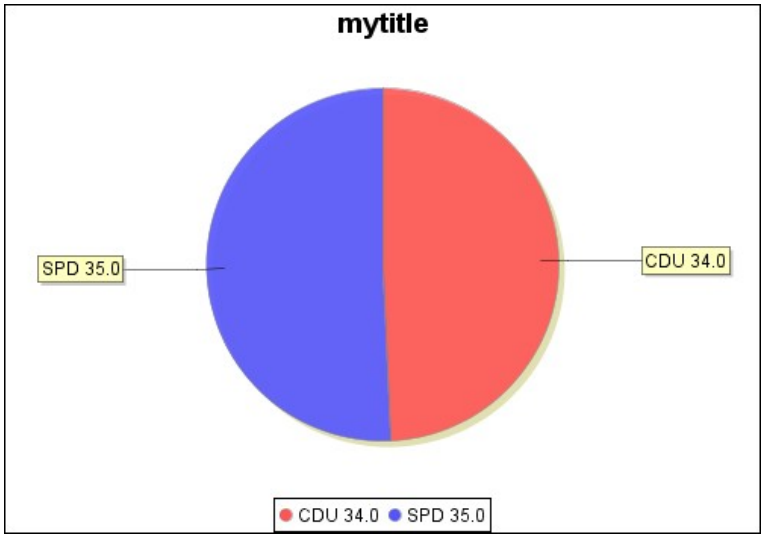


Figure 9: Result of Request Example 1

4.3.2 Format Two

```
http://localhost:8080/contextName/charts?
chart=Bar3D&title=mytitle&legend=true&width=500&height=350&format=image/png&CDU=34,
23&SPD=35,12
```

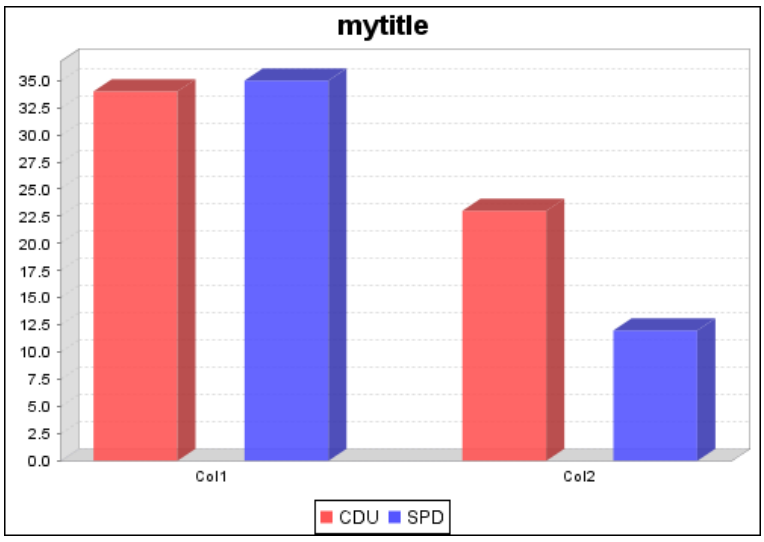


Figure 10: Result of Request Example 2

4.3.3 Format Three

```
http://localhost:8080/contextName/chart?
chart=XYLine&title=mytitle&legend=true&width=500&height=350&format=image/png&CDU=19
75,25;1976,45;1977,40&SPD=1975,30;1976,40;1977,45&xAxis=myXAXis&yAxis=myYAxis
```

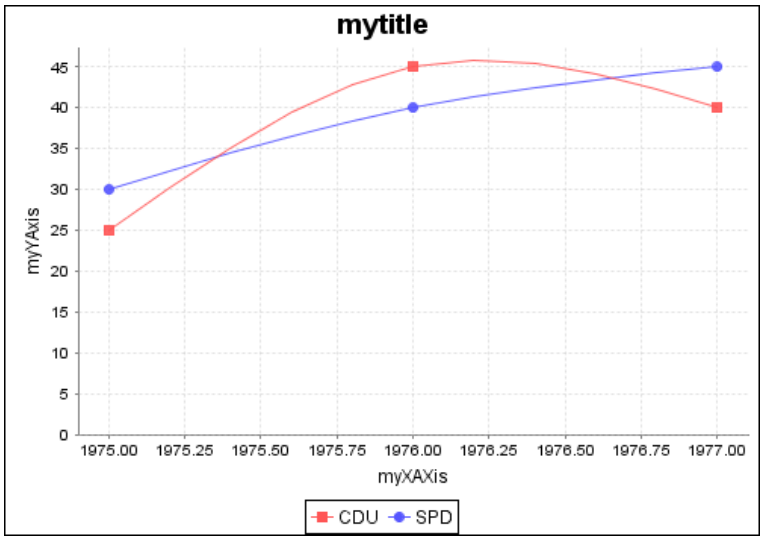


Figure 11: Result of Request Example 3

4.3.4 Other Format Examples

Note: Update the style address to the required web address

```
http://localhost:8080/contextName/charts?
chart=XYLine&title=mytitle&legend=true&width=500&height=350&format=image/png&CDU=19
75,25;1976,45;1977,40&SPD=1975,30;1976,40;1977,45&xAxis=myXAXis&yAxis=myYAxis&style
=http://www.myserver.com/config.xml
```

5 Usage of deegree Charts with deegree WMS and SLD

The major reason for having a special servlet for rendering charts in deegree and not to use already available solutions is that it should be possible to use charts as points symbols within a map rendered by deegree WMS. The deegree WMS uses SLD for defining a layer style and so it should be possible to embed dynamically created charts without introducing deegree specific elements into a SLD document. To enable this deegree introduces a mechanism that enables you to access the properties of a feature – or better: their values – to create a URL for referencing an external graphic within a SLD point symbolizer.

According to OGC's SLD 1.0 specification a point symbolizer using an external graphic looks like this:

```
<sld:PointSymbolizer>
   <sld:Graphic>
      <ExternalGraphic>
         <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:type="simple" xlink:href="file:///c:/graphics/symbol.png"/>
         <Format>image/png</Format>
      </ExternalGraphic>
      <sld:Size>20</sld:Size>
      <sld:Rotation>0</sld:Rotation>
   </sld:Graphic>
</sld:PointSymbolizer>
```

As you see an external graphic will be referenced by an URL defined through OnlineResource element. The can be a file URL as well as a HTTP URL.

Now, deegree allows you to define any part of such an URL using the properties of a feature to be rendered using the embeding sld:Style. To do this property values are referenced by a property's name bordered by two '$' chars. For example: if a feature stores the name of the symbol that shall be used for rendering it in a property named 'app:Symbol' but not the location on the servers filesystem the PointSymbolizer defined above can handle this by changing the OnlineResource@xlink:href value too:

```
<sld:PointSymbolizer>
   <sld:Graphic>
      <ExternalGraphic>
         <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:type="simple" xlink:href="file:///c:/graphics/$Symbol$"/>
         <Format>image/png</Format>
      </ExternalGraphic>
      <sld:Size>20</sld:Size>
      <sld:Rotation>0</sld:Rotation>
   </sld:Graphic>
</sld:PointSymbolizer>
```

Notice that just the un-qualified property name is used and that the name is case sensitive.

Coming back to using charts as point symbols the only thing you have to do is to use a HTTP Get call targeting the deegree Charts servlet using value parameters read from the features to be rendered:

```
<sld:PointSymbolizer>
   <sld:Graphic>
      <ExternalGraphic>
         <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                         xlink:type="simple"
xlink:href="http://localhost/chart/charts?
chart=Pie&amp;amp;title=&amp;amp;legend=false&amp;amp;width=200&amp;amp;height=200&amp;amp;xAxis=party&
amp;amp;yAxis=value&amp;amp;format=image/png&amp;amp;lblType=KeyValue&amp;amp;SPD=$spd1999$&amp;amp;CDU
=$cdu1999$"/>
         <Format>image/png</Format>
      </ExternalGraphic>
      <sld:Size>200</sld:Size>
      <sld:Rotation>0</sld:Rotation>
   </sld:Graphic>
</sld:PointSymbolizer>
```

Using this definition we will get a pie chart of 200 pixel size for each feature displaying the percentages of the German political parties SPD and CDU read from feature properties 'cdu1999' and 'spd1999' (Figure 9).

Appendix A Tomcat Deployment Descriptor

File: WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
   <display-name>deegree Charts</display-name>
   <servlet>
      <servlet-name>chart</servlet-name>
      <servlet-class>org.deegree.enterprise.servlet.ChartServlet</servlet-class>
      <init-param>
         <param-name>ConfigFile</param-name>
         <param-value>WEB-INF/conf/deegreeCharts/chart_config.xml</param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
      <servlet-name>chart</servlet-name>
      <url-pattern>/charts</url-pattern>
   </servlet-mapping>
</web-app>
```

Appendix B deegree Charts Configuration File

File: WEB-INF/conf/deegreeCharts/chart_config.xml

```
<dg:ChartConfiguration xmlns:dg="http://www.deegree.org/charts"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.deegree.org/charts
http://schemas.deegree.org/charts/1.0.0/charts_configuration.xsd">
   <dg:GeneralChartSettings>
      <dg:Antialiasing>true</dg:Antialiasing>
      <dg:TextAntialiasing>true</dg:TextAntialiasing>
      <dg:BorderVisibility>false</dg:BorderVisibility>
      <dg:RectangleInsets>0,0,0,0</dg:RectangleInsets>
      <dg:BackgroundColor>0xffffff</dg:BackgroundColor>
      <dg:BackgroundOpacity>0</dg:BackgroundOpacity>
      <dg:ForegroundOpacity>1</dg:ForegroundOpacity>
      <dg:FontFamily>ARIAL</dg:FontFamily>
      <dg:FontType>PLAIN</dg:FontType>
      <dg:FontSize>12</dg:FontSize>
   </dg:GeneralChartSettings>
   <dg:GeneralPlotSettings>
      <dg:ForegroundOpacity>0.8</dg:ForegroundOpacity>
      <dg:BackgroundColor>0xffffff</dg:BackgroundColor>
      <dg:BackgroundOpacity>0</dg:BackgroundOpacity>
      <dg:OutlineVisibility>false</dg:OutlineVisibility>
   </dg:GeneralPlotSettings>
   <dg:PiePlotSettings>
      <dg:InteriorGap>0.01</dg:InteriorGap>
      <dg:LabelGap>0.01</dg:LabelGap>
      <dg:Circular>true</dg:Circular>
      <dg:BaseSectionColor>0xFFFFFF</dg:BaseSectionColor>
      <dg:BaseSectionOpacity>0</dg:BaseSectionOpacity>
      <dg:ShadowColor>0xDDDDDD</dg:ShadowColor>
      <dg:ShadowOpacity>0</dg:ShadowOpacity>
   </dg:PiePlotSettings>
   <dg:LinePlotSettings>
      <dg:RenderLines>true</dg:RenderLines>
      <dg:RenderShapes>true</dg:RenderShapes>
   </dg:LinePlotSettings>
</dg:ChartConfiguration>
```

Appendix C XSD-Schema for Configuration File

File: http://schemas.deegree.org/charts/1.0.0/charts_configuration.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:dg="http://www.deegree.org/charts"
targetNamespace="http://www.deegree.org/charts" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="ChartConfiguration" type="dg:ChartConfigurationType">
        <xs:annotation>
            <xs:documentation>
                Root element of deegree chart description/configuration document
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="ChartConfigurationType">
        <xs:annotation>
            <xs:documentation>
                A chart configuration contains four sections. The first two
                elements contains general informations about chart and plot
                settings, the least two desribes settings for pie and line plots
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="GeneralChartSettings"
                        type="dg:GeneralChartSettingsType"/>
            <xs:element name="GeneralPlotSettings"
                        type="dg:GeneralPlotSettingsType"/>
            <xs:element name="PiePlotSettings" type="dg:PiePlotSettingsType"/>
            <xs:element name="LinePlotSettings" type="dg:LinePlotSettingsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GeneralChartSettingsType">
        <xs:annotation>
            <xs:documentation>
                general setting common to all charts. jFreeChart differs between a
                chart and a plot. A chart represent the complete graphic including
                title and legend. A plot is embedded into a chart and represent the
                pie-, bar-, line- etc. diagram. This type describes colors,
                opacities, font and visibilities
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="Antialiasing" type="xs:boolean">
                <xs:annotation>
                    <xs:documentation>
                        true if antialiasing shall be use for rendering a plot
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="TextAntialiasing" type="xs:boolean">
                <xs:annotation>
                    <xs:documentation>
                        true if antialiasing shall be used for rendering labels and
                        title
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="BorderVisibility" type="xs:boolean">
                <xs:annotation>
                    <xs:documentation>
                        true if chart border shall be visible
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="RectangleInsets" type="xs:string">
                <xs:annotation>
```

```
                <xs:documentation>
                    insets of plot too chart border.
                    Must be four comma separated integer values (e.g. 2,5,2,5)
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BackgroundColor" type="xs:string">
            <xs:annotation>
                <xs:documentation>
                    hexadecimal coded color of a chart background;
                    e.g. 0xFFFF00 -> will cause a yellow bachground
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BackgroundOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    opacity of a chart background; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ForegroundOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    opacity of a chart foreground; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="FontFamily" type="xs:string">
            <xs:annotation>
                <xs:documentation>
                    name of the font family that shall be used for rendering
                    labels (e.g. ARIAL, HELVETICA etc.)
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="FontType">
            <xs:annotation>
                <xs:documentation>
                    font type that shall be used for rendering labels
                </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="PLAIN"/>
                    <xs:enumeration value="BOLD"/>
                    <xs:enumeration value="ITALIC"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="FontSize" type="xs:float"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="GeneralPlotSettingsType">
    <xs:annotation>
        <xs:documentation>
            general setting common to all plots.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="ForegroundOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    opacity of a plot foreground; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BackgroundColor" type="xs:string">
            <xs:annotation>
```

```
                <xs:documentation>
                    hexadecimal coded color of a plot background;
                    e.g. 0xFFFF00 -> will cause a yellow background
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BackgroundOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    opacity of a plot background; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="OutlineVisibility" type="xs:boolean">
            <xs:annotation>
                <xs:documentation>
                    true if plot outline shall be visible
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PiePlotSettingsType">
    <xs:annotation>
        <xs:documentation>settings for Pie and Pie3D plots.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="InteriorGap" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    percentual interior gap between a plots diagram and a plots
                    border; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="LabelGap" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    percentual gap between label and label box; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Circular" type="xs:boolean">
            <xs:annotation>
                <xs:documentation>
                    true if plot shall be circular indepent of ratio of chart
                    width/height
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BaseSectionColor" type="xs:string">
            <xs:annotation>
                <xs:documentation>
                    hexadecimal encoded color of a pie plots base;
                    e.g. 0xFFFF00 -> will cause a yellow base
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="BaseSectionOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>opacity of a pie plots base seciton; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ShadowColor" type="xs:string">
            <xs:annotation>
                <xs:documentation>
                    hexadecimal encoded color of a pie plots shadow;
```

```
                    e.g. 0xFFFF00 -> will cause a yellow shadow
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ShadowOpacity" type="xs:float">
            <xs:annotation>
                <xs:documentation>
                    opacity of a pie plots shadow. If you do not want to have
                    a shadow set this value to 0; 0..1
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="LinePlotSettingsType">
    <xs:annotation>
        <xs:documentation>settings for Line and XYLine plots.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="RenderLines" type="xs:boolean">
            <xs:annotation>
                <xs:documentation>
                    true if line shall be rendered;
                    if false just shapes (points) will be rendered
                    (if RenderShapes is set to true)
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="RenderShapes" type="xs:boolean">
            <xs:annotation>
                <xs:documentation>
                    true if shapes (points) shall be rendered;
                    if false just lines will be rendered
                    (if RenderLines is set to true)
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```