GreenPoint, Inc.

WebCharts3D v5.0
Developer's Guide

July 3, 2004

Table of Contents

1.0 Introduction

Scope

This document provides WebCharts3D v5.0 developer's information and instructions for using WebCharts3D Bean within Swing framework and in the server environment. This process will require some experience with Java, HTML, XML and/or Swing programming. Anyone comfortable with these skills should be able to perform the procedures described in this document.

WebCharts3D API was changed in version 5.0. Old API is still available, but the new features can be accessed only by using new API functions.

2.0   Using Swing Chart component

WebCharts3D includes Swing component *MxComponent* located in *com.gp.api.swing* package. This class can be used in rich client applications to display, print, and save charts into the files in a variety of formats. After you create an instance of *MxComponent* you can add it to any Swing container and treat as an ordinary Swing component. WebCharts3D documentation included with the product contains full description of *MxComponent* APIs.

Prerequisites for using WebCharts3D v5.0 with Swing

Before beginning, you should ensure the following:

- You have Java JDK v1.4 (or later), and are familiar with Java and Swing programming.
- WebCharts3D v5.0 has been installed on your machine and is functioning properly.
- You are familiar with XML and understand WebCharts3D's concept of XML style and model.
- WebCharts3D jar file wc50.jar is added to your project's CLASSPATH or made available to JDK in some other way. You can use wcruntime.jar file located in WEB-INF/lib subfolder of the installation directory instead if having chart editing functionality is not important to you.

Displaying charts

To display a chart you need to create *MxComponent*, set it's style and then model, and add it to a Swing container:

```
MxComponent chart = new MxComponent();
MxFrameChartStyle style = new MxFrameChartStyle();
style.legend.placement = MxStandardConstants.Placement.RIGHT;
chart.setStyle(style);
chart.setModel(new MxSampleChartModel(2,5));
add(chart);
```

WebCharts3D Designer can automatically generate Java code required to create a chart displayed in it. You can switch to XML Style pane and from the right mouse button menu select View>>Java. Copy and paste the code into your program. You can also save chart's XML style into a file and then instantiate style in your code.

To completely change the chart's style and/or model use *setStyle* and *setModel* functions. For example, to switch between pie chart and bar chart you can use the following code:

```
if(switchToBar) {
    MxFrameChartStyle style = new MxFrameChartStyle();
    style.legend.placement = MxStandardConstants.Placement.RIGHT;
    chart.setStyle(style);
} else {
    MxPieChartStyle style = new MxPieChartStyle();
    style.legend.placement = MxStandardConstants.Placement.RIGHT;
    chart.setStyle(style);
}
```

Defining model

The example above uses MxSampleChartModel that is a useful tool for testing, but does not contain any meaningful data. To provide data for the chart you can either use *MxStandardChartModel* class or define your own model by subclassing *MxChartModel*.

Using standard chart model

Class MxStandardChartModel subclasses MxChartModel and provides API functions for inserting, removing and changing model's values and labels. For example, to create 2x2 model you can enter:

```
MxStandardChartModel model  = new MxStandardChartModel();
model.insertRow("First row", null,-1);
model.insertRow("Second row", null,-1);
model.insertCol("First col", new double[] {1,2}, -1)
model.insertCol("Second col", new double[] {1,2}, -1)
```

You can also create an instance of this class from model's XML, databases or separated files. For example, to create a new instance from XML you can type:

```
MxChartModel model = MxChartModel.fromXML("string", null)
```

Where string can be created in the Xml Model pane of the designer and/or taken from there or from code pane.

Using custom chart model

To create a custom chart's model you need to subclass MxChartModel class. For example, you can define MyChartModel as:

```
public class MyChartModel extends MxChartModel

    public int getColCount() { return ??? ; }
    public int getRowCount() { return ??? ; }

    public double valueAt(int rowIndex, int colIndex) { return ??? ; }
    public String getColLabelAt(int colIndex) { return ??? ; }
```

```
        public String getRowLabelAt(int rowIndex) { return ??? ; }

        public int getRowSelection() { return 0; }
        public void setRowSelection(int rowSelection) {}
        public int getColSelection() { return 0;}
        public void setColSelection(int colSelection) { }
        public void setRowLabelAt(int rowIndex, String s) {}
        public void setColLabelAt(int colIndex, String s) {}
        public void setValueAt(int rowIndex, int colIndex, double value){}
}
```

Updating model

To dynamically update chart's model without recreating the model object you need to use *MxStandardChartModel*. The instances of this model are returned by all XML, database and import functions. If you created your own model, you can convert it to *MxStandardChartModel* by calling asStandardModel() function:

```
        chart.setStyle(new MxFrameChartStyle());
        chart.setModel(new MxSampleChartModel(2,5).asStandardModel());
```

To update *MxStandardChartModel* you can use functions defined in the API section of Model chapter of the documentation included with the product. You will need to call *repaint*() method to refresh the chart after the model is modified. For example, the following code creates an illusion of moving bar or line chart by deleting the first column and inserting a new column of data at the end:

```
        MxStandardChartModel model = (MxStandardChartModel) chart.getModel();
        model.deleteCol(0);
        model.insertCol(Integer.toString(3000+count++), new double[] {.....},-1 );
        chart.repaint();
```

Updating style

To update the chart's style without recreating the style object you need to change the chart's styles and then call *repaint*() method to refresh the chart. For example, the following code creates an illusion of rotating pie by increasing the initial angle by 5 degrees:

```
        MxPieChartStyle style = (MxPieChartStyle) chart.getStyle();
        style.angle += 5;
        chart.repaint();
```

Processing events

To process a selection change event, add an instance of *MxComponentListener* to the chart and check the event id for *MxComponentEvent.STATE_CHANGED*. For example,

the following code can be used to update a label with the values of the currently selected element:

```
chart.addMxComponentListener(new MxComponentListener() {
  public void componentChanged(MxComponentEvent event) {
    if(event.getID() == MxComponentEvent.STATE_CHANGED) {
      MxStandardChartModel model = (MxStandardChartModel) chart.getModel();
      label.setText("Selected row: "+ model.getRowSelection()+
                    " column:"+model.getColSelection());
    }
  }
});
```

Using Applets

Class *com.gp.api.swing.applet.MxApplet* is a subclass of JApplet that contains one child - an instance of MxComponent class. To access this instance you can use *getChart()* function. WebCharts3D Designer will produce and show in the Code pane the JSP code that produces the applet. This code uses jsp:plugin tag to generate appropriate HTML and requires JSP. You can also code HTML manually (see example below).

In order to run the licensed version of the applet you need to create webcharts3D.xml file and place it in the same folder with the applet archive. Applet's webcharts3D.xml should at least contain the (client-side) license key:

<xml license='0000-0000-0000-0000'/>

You can use either wcapplet.jar or server-side run-time wcruntime.jar files as the applet's archive. Note that while wcapplet.jar is smaller it does not contain image-generation code and cannot be used to save images.

MxApplet supports the following parameters:

xmlStyle
       Defines initial chart's style in XML format.
xmlStyleURL
       Defines initial chart's style URL. This parameter will not be used if xmlStyle is
       specified.
xmlModel
       Defines initial chart's model in XML format.
xmlModelURL
       Defines initial chart's model URL. This parameter will not be used if xmlModel is
       specified

WebCharts3D applet API consists of the following functions:

public String getXmlStyle()
>   Returns XML representation of the chart's style.

public String getXmlModel()
>   Returns XML representation of the chart's model.

public void setXmlStyle(String xml)
>   Sets chart's style from XML and repaints the chart.

public void setXmlModel(String xml)
>   Sets chart's model from XML and repaints the chart.

public void setXmlModel(String xml)
>   Sets chart's model from XML and repaints the chart.

public MxComponent getChart()
>   Returns an instance of charting component. Note that the entire API available in the previous versions has been moved to MxComponent class and to dynamically access the chart properties and contents you need to use getChart().

The following HTML code demonstates usage of the applet:

```
<object classid="clsid:CAFEEFAC-0014-0002-0000-ABCDEFFEDCBA" WIDTH=700
HEIGHT=480
   codebase="http://java.sun.com/products/plugin/autodl/jinstall-1_4_2-windows-
   i586.cab#Version=1,4,2,0">
   <param NAME = CODE VALUE = com.gp.api.swing.applet.MxApplet.class >
   <param NAME=ARCHIVE value=wc50.jar>
   <param NAME="type" VALUE="application/x-java-applet;jpi-version=1.4">
   <param NAME="scriptable" VALUE="true">
   <param NAME="xmlStyle" VALUE="<pieChart depth='thick'
angle='320'><dataLabels style='Pattern' placement='Inside' decoration='Simple'
background='#FFFAF6'/></pieChart>">
   <param NAME="xmlModel" VALUE="<XML type='default'> <COL>2000</COL>
   <COL>2001</COL> <COL>2002</COL> <COL>2003</COL>
   <COL>2004</COL> <ROW col0='100.0' col1='200.0' col2='100.0' col3='180.0'
   col4='200.0'>Sample 0:</ROW> </XML>">
</object>
```

3.0 Using Swing Designer components

WebCharts3D v5.0 provides Swing chart designer components located in *com.gp.api.swing.designer* package that can be incorporated into any Swing application and allow the end-user to create and/or modify chart style objects - *MxDesignTable* and *MxDesignPanel*.

MxDesignTable

Class *MxDesignTable* can be used to edit arbitrary chart's styles. MxDesignTable is a subclass of JTable and is used by WebCharts3D Designer to display the properties of the edited chart. *MxDesignPanel* class includes *MxDesignTable* as its right component. Usually you would not use this class directly, but only as a part of the design panel. For the complete list of the API functions available in this class please refer to the documentation provided with the product.

MxDesignPanel

Class *MxDesignPanel* can be used to edit instances of *MxComponent*. *MxDesignPanel* is a subclass of JSplitPane with a number of additional API methods described in the documentation provided with the product. The design panel includes left pane that displays edited component and allows the end-user to resize it and the right pane that shows the chart's attributes in Expert or Design modes.

The following code demonstrates using *MxDesignPanel*:

```
// Create MxComponent or obtain an existing instance of it:
MxComponent chart = new MxComponent();
MxFrameChartStyle style = new MxFrameChartStyle();
style.legend.placement = MxStandardConstants.Placement.RIGHT;
chart.setStyle(style);
chart.setModel(new MxSampleChartModel(2,5));
chart.setSize(300,200);

// Create MxDesignPanel and set the component as its target:
panel = new MxDesignPanel();
panel.setTarget(chart);
add(panel);
panel.setDividerLocation(300);
```

To obtain or set target's styles as XML use corresponding API functions:

```
panel.getTarget().getStyle().toXML()
```
and/or
```
panel.getTarget().setStyle(MxWidgetStyle.read("…",null) )
```

4.0   Setting up WebCharts3D on the server

WebCharts3D provides a powerful server component that can be used with any JSP compliant web or application server to deliver real-time interactive server-generated charts to the browsers and mobile devices.

Prerequisites for using WebCharts3D v5.0 on the web server

Before beginning, you should ensure the following:

- You have a JSP-compliant web or application server, Java JDK v1.4 (or later), and are familiar with HTML and Java server-side programming.
- WebCharts3D v5.0 has been installed on either server or developer's machine and is functioning properly.
- You are familiar with XML and understand WebCharts3D's concept of XML style and model.
- If your server is running on a UNIX machine that does not have X-Windows server running, your Java Virtual Machine launch parameters in your server configuration file set property java.awt.headless to true:

        java.awt.headless=true

    For the specific details about setting JVM parameters on your web or application server please refer to the vendor's documentation.

Making WebCharts3D available for the server

WebCharts3D installation folder contains **WEB-INF** folder with the following files:

    lib/wcruntime.jar   - contains WebCharts3D run-time classes
    lib/webcharts3D.xml   - contains configuration parameters
    wc3d50.tld   - contains WebCharts3D tag library

In order to setup WebCharts3D on the server, you need to make WebCharts3D classes available for the web server. There are multiple ways to do so, but with most web servers you can simply copy **/WEB-INF/lib/wcruntime.jar** file to **WEB-INF/lib** folder of your web applications.

WebCharts3D server-side configuration file **webcharts3D.xml** should be copied along with **wcruntime.jar** file. Usually you will not need to modify this file with the exception of adding a valid WebCharts3D server-side license to it. To learn what parameters can be modified in this file please refer to "Configuring Server Component" part of this paper.

Tag library file **wc3d50.tld** is required only if you are planning to use custom tags inside your application. There are multiple ways to make tag library available for the web server, but with most servers this can be done by copying **wc3d50.tld** into WEB-INF folder of the application root. WebCharts3D v5.0 tag library uses taglib version 2.0 (J2EE 1.4). If your server does not support this version of tag library you might need to create a tag library descriptor for the supported version as it is described in the documentation supplied with the product.

The above files can be automatically deployed to your web or application server by using "Server" tab of the "Setup" dialog in WebCharts3D Designer. You can fill out application home location and Designer will automatically deploy required files into this directory. The additional benefit of using this method is the ability to modify configuration file by using GUI interface.

Testing WebCharts3D installation

WebCharts3D installation folder contains **sample** folder that can be used to verify WebCharts3D setup. You can copy this folder to the application root directory and then enter http://......./sample/sample.jsp into a web browser's address bar. If the setup was successful, you should be able to see sample page with charts.

5.0   Developing web pages

WebCharts3D can be used with any JSP enabled server to produce interactive animated images in a number of different raster and vector formats including PNG, JPEG, TIFF, Flash, SVG, and PDF. To use WebCharts3D on the server side, the first decision to make is what method of image production to employ.

There are two ways in which an image can be generated on the server and presented to an end user:

Indirect method
> The image is generated and stored on the server and the <IMG> tag's (or plugin's tag) *src* attribute contains the unique ID of the image. In case of images, the image map can be included into the page together with <IMG> tag that will provide interactivity. When the request for the actual bytes comes to the server, it retrieves the image using the unique ID, serves it to the browser, and reduces its request count.

Direct method
> The image is generated when the browser attempts to retrieve the image. The <IMG> tag's src attribute can contain the image parameters, or the server script can be customized to produce a desired chart.

The following table summarizes the benefits and the drawbacks of these methods:

	Indirect	Direct
Interactivity	The image map can be produced and embedded into the page together with the IMG tag. The image is linked to the image map via the image's unique ID.	The browser receives a byte stream and the image map cannot be integrated with the image. (Modern browsers do not support remote image maps). Only SWF and SVG formats can provide interactivity in this mode.
Updating	In general, the entire page should be refreshed when the chart has to be changed.	The image can be changed without refreshing the page. The request URL can contain the hints on how to generate the new image.

It is also possible to utilize WebCharts3D API to generate images on the server and save them into a file or embed inside emails. In this case you will be bypassing WebCharts3D chart delivery mechanism and will need to implement such procedure manually.

Server-side API

WebCharts3D server-side API adds the following three classes located in *com.gp.api.jsp* package to the set of standard API functions described in the product documentation: *MxServerComponent*, *MxServerProperties*, and *MxChartDescription*.

MxServerComponent

Class *MxServerComponent* is WebCharts3D server-side image generation signleton bean. Usually you should not create an instance of this class directly, but obtain it by calling static method of this class *getDefaultInstance* that takes *ServletContext* as a parameter:

MxServerComponent server = MxServerComponent.getDefaultInstance(application);

This function looks up an instance of MxServerComponent in the servlet context. If it exists, then this instance is returned, otherwise a new instance is created using default constructor, stored in the context and then returned.

The API for *MxServerComponent* class includes a large number of functions for producing, storing and delivering images to the clients. For the list of the API functions please refer to the product documentation.

MxServerProperties

Class *MxServerProperties* is used to initialize an instance of *MxServerComponent*. The XML version of this class is stored in webcharts3D.xml file and is used by the default constructor of *MxServerComponent*.

MxChartDescription

Class *MxChartDescription* holds all information about the chart the server needs to create an image. You can obtain an instance of this class by calling *newImageSpec*() function of MxServerComponent.

The most important fields of this class are style and model that contain correspondingly the chart's style and the chart's model. There are multiple ways to populate these fields:

Xml style

Xml style can be specified directly as a java string:

myChart.style = "…";

or loaded from a file using loadStyles API function:

myChart.loadStyles("filename");

Xml model

Xml model can be specified directly as a java string:

myChart.model = "…"

or loaded from a file containing XML or delimited data using one of the linkTo finctions:

myChart.linkTo("filename");
myChart.linkTo("c:\\test\\test.csv",',',",true,false,false,null);
myChart.linkTo(dbConn,"select * from Sample",false, false);

Generating chart's bytes

To generate the chart's bytes and/or to save the chart into a file:

1. Import *com.gp.api.jsp* package: <%@ page import = "com.gp.api.jsp.*" %>

2. Obtain an instance of *MxServerComponent*. If you do not have access to the servlet context to use with *getDefaultInstance* method, you will need to create a new instance of this class:

   MxServerComponent svr =
               new MxServerComponent(MxServerProperties.newInstance());

   Due to a number of the different checks that are performed during server component instantiation it is a relatively long operation and we recommend you to create a single instance of this class. This class supports multi-threading and can be called from different threads.

3. Create an instance of *MxChartDescription* and fill it.

   MxChartDescription chart = svr.newImageSpec();
   chart.width = 400 ;
   chart.height= 300 ;
   chart.type = "swf";
   chart.style = "<pieChart type='Pie'>…</pieChart>" ;
   chart.model = "<XML type='default'>…</XML>";

4. Obtain chart's bytes by using *getBytes()* function:

   byte bytes [] = svr.getBytes(chart);

   or save the chart into a file:

   svr.saveBytesTo(chart, "C:\\test.swf");

Using direct method for chart delivery

When you use direct method of generating charts, the image tag (or plugin tag) url points to some jsp page that returns chart's bytes instead of pointing to getimage.jsp that retrieves the bytes from the image cache. This method guarantees that the image will never expire, but cannot be used to produce interactive images since image maps has to be embedded into the page. This method can be used to produce interactive Flash and SVG charts.

1. Import *com.gp.api.jsp* package: <%@ page import = "com.gp.api.jsp.*" %>

2. Obtain an instance of *MxServerComponent*. As opposed to indirect method, you do not have to use *getDefaultInstance* method and can create an instance directly. However, we recommend that you use *getDefaultInstance* method to facilitate caching and take advantage of built-in resource management. (See also comments in "Generating chart's bytes" section).

   MxServerComponent svr =
           MxServerComponent.getDefaultInstance(application);

3. Create an instance of *MxChartDescription* and fill it:

   MxChartDescription chart = svr.newImageSpec();
   chart.width = 400 ;
   chart.height= 300 ;
   chart.type = "swf";
   chart.style = "<pieChart type='Pie'/>" ;
   chart.model = "<XML type='default'>…</XML>";

4. Return image bytes to the server. You can obtain image bytes and then write them to response, or call *writeBytesTo* function which will set content length and type and write the bytes:

   svr.writeBytesTo(chart,response);

Note that a page that returns actual image bytes cannot have any text in it including spaces and newlines. Make sure that there are no spaces after the last %> and that your file starts with:

<%@ page import = "com.gp.api.jsp.*" %><%
so that there are no spaces or newlines between %> and <%. After you test the page in the browser you can include a reference to it into your html page.

Using indirect method for chart delivery

When you use indirect method of generating charts (method used by the code produced by Designer), the image tag (or plugin tag) url points to a special jsp page and contains a unique image id that is used to retrieve the image from the cache. This page can be located anywhere and named in any way, as long as the same name is provided as an argument to *getImageTag* function. The main advantages of using indirect method are:

- Ability to generate interactive images by embedding image maps inside web pages
- Code is stored inside the target jsp page instead of creating a separate jsp page for every chart.

To produce charts using indirect method:

1. Import *com.gp.api.jsp* package: <%@ page import = "com.gp.api.jsp.*" %>

2. Obtain an instance of *MxServerComponent*. As opposed to direct method, you should always use *getDefaultInstance* method unless you modify getimage.jsp file.

   MxServerComponent svr =
           MxServerComponent.getDefaultInstance(application);

3. Create an instance of MxChartDescription and fill it:

   MxChartDescription chart = svr.newImageSpec();
   chart.width = 400 ;
   chart.height= 300 ;
   chart.type = "swf";
   chart.style = "<pieChart type='Pie'/>" ;
   chart.model = "<XML type='default'>…</XML>";

4. Produce image or plugin tag using *getImageTag* function and embed this tag into your page:

   out.write(svr.getImageTag(chart,"getImage.jsp?image="));

   getImageTag will generate html tags based on the type of the image and append the unique id of the image to the provided url.

6.0   Understanding caching policies

WebCharts 3D image generation engine implements resource pooling and image caching algorithms to provide the best possible performance. In some cases adjusting caching policies by modifying configuration file can improve the overall performance of your system. This section describes WebCharts3D caching algorithm:

1.   When the image request is received, the component looks up the image in the image cache using the request parameters as the key. If the image is not found, it is generated and is assigned a unique ID. The image (existing or new) is stamped with the current date and time. The requests that return a pointer to the image (for example, as an image tag) increment the reference count of the image by one. This reference count tells the Image Server how many pages requested the image's url, but never actually retrieved it. Note, that the requests that dispatch the image immediately do not change the reference count.

2.   When the request for the stored image is received, it looks up the image by it's unique ID and returns its bytes. If the reference count is positive, it is decreased by one.

3.   When gc() function is called, WebCharts3D garbage collector is invoked. (This is not Java VM garbage collector). Garbage collector checks the images in the image cache. An image is deleted from the image cache when:

   a)   Its reference count is zero and the time since the last request for this image exceeds minTimeout. minTimeout allows image server to avoid regenerating images that are often requested.

   b)   The time since the last request for this image exceeds maxTimeout. maxTimeout allows image server to remove the "dead" images. For example, if the browser requested the page and the image was generated, but the client opted to close the browser and the image was never actually dispatched, then this image will have a positive reference count and will be purged after maxTimeout expires.

   c)   A new image should be placed into the full cache and the image in question has the earliest stamp date.

   NOTE. Server side tags and *MxServerComponent's getDefaultInstance* function automatically call garbage collection. If you use other methods to generate the image you should call *gc()* function manually.

   NOTE. When a component is destroyed as a result of the system garbage collection it destroys all cached images both in the memory and on the disk.

7.0 Configuring server component

WebCharts3D server-side charting engine is a complex component that supports multi-threading and provides its own memory and resource management. Depending upon the environment you can find it useful to change some of the engine's configuration settings.

To configure the server-side component you can either instantiate it using the constructor that takes and instance of *MxServerProperties* or modify webcharts3d.xml file and use either default constructor or *getDefaultInstance* method. Webcharts3d.xml file can be edited manually or by using WebCharts3D's Designer Setup dialog. After this file is modified the web server should be restarted for the new settings to take effect.

Server component properties

This section describes the attributes of webcharts3d.xml configuration file and *MxServerProperties* class that can be modified. Modifying other attributes can prevent the engine from working properly.

Cache	Type of the cache to use for the generated charts, can be File or Memory.
minTimeout	Minimum time interval to keep the images in the cache.
maxTimeout	Maximum time interval to keep the images that were produced but never retrieved by the browser in cache.
cacheSize	Maximum number of cached charts or -1 for the unlimited cache. The value of 0 will disable the cache and therefore disable the indirect method of chart delivery.
maxThreads	Maximum number of the requests that will be processed simultaneously. All other requests will be waiting until one of the executing requests is complete. Chart production can be memory and CPU intensive process and maxThreads parameter can be used to fine-tune the performance of a particular server.
gcInterval	WebCharts3D garbage collection interval – only the first call to gc() function during this interval will be actually executed.
compression	This is a hint to the engine whether to compress the produced images (if this image type supports compression)

	or not. Compressed images are usually much smaller, but will require more processing time.
useBuiltinXmlParser	True or false
xmlParser	SAX compliant XML parser to be used when useBuiltinXmlParser is set to true.
license	WebCharts3D server-side license key.
folders	Contains information about server's virtual directories. Currently only one entry in this table is used - Images that defines a directory to store images when File cache type is used.

Editing configuration file

Webcharts3D server configuration file has the following structure:

```
<server image="<default image type>"
      cache="Memory|File"
      minTimeout="milliseconds"
      maxTimeout="milliseconds"
      cacheSize="integer"
      maxThreads="integer"
      gcInterval="milliseconds"
      compression="Default| Uncompressed"
      useBuiltinXmlParser="true|false"
      xmlParser="org.apache.xerces.parsers.SAXParser"
      license="server-side license"
      >
      <folders>
            <map key="images" value="/webcharts3d/tmp"/>
      </folders>
</server>
```

Using MxServerProperties

MxServerProperties can be instantiated by using *newInstance*() static function that returns a new instance of this class, or by using *read() or read(String filename)* static functions that return an instance created from webcharts3d.xml or specified configuration file.

The following are the fields of *MxServerProperties* that can be modified:

```
public String image       = "PNG"    ;
public Cache  cache       = Cache.MEMORY ; //  Or Cache.FILE
public long minTimeout  =   5000    ;
public long maxTimeout  =  30000    ;
public int  cacheSize      = 100  ;
public int maxThreads    = 20;
public long   gcInterval    = 3000      ;
public Compression compression = Compression.DEFAULT;  // Or Compression.UNCOMPRESSED
public boolean useBuiltinXmlParser = true;
public String xmlParser = XmlReader.Xerces;
public String license = null;
public IcVDTable folders = new IcVDTable();
```

To define the temporary folders for the file based cache you need to define the mapping between "Images" virtual directory and the physical path:

```
folders.define("Images","C:\\Temp");
```

Using Setup Dialog

WebCharts3D Designer's Setup dialog can be used to create and edit webcharts3d.xml file in GUI mode. To edit the file you need to either define application home or to define 'Jar and configuration file location' and click on the 'Deploy/Edit Configuration' button. The following dialog is displayed:
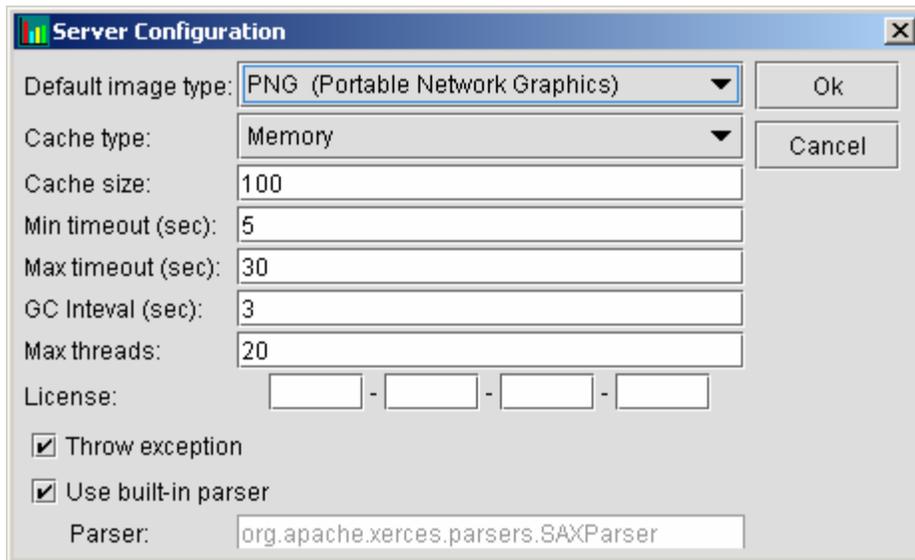
Note that this dialog uses seconds instead of milliseconds as time intervals.

8.0   Where to get additional help

WebCharts3D v5.0 Designer includes additional built-in documentation for more information on the various API calls available. The distribution also includes sample code showing various API calls and can run these samples. WebCharts3D Designer's "Xml Style Pane" can be used to automatically produce the code required to create appropriate chart style instance.

For more information about WebCharts3D please contact support@gpoint.com.

APPENDIX A – MxComponent API

The following are some of the com.gp.api.swing.MxComponent API methods. To view all available methods please refer to the internal Designer's documentation.

public *MxComponent*()
 Creates a new instance of *MxComponent*.

public *MxWidgetStyle* getStyle()
 Returns the chart's style object.

public void setStyle(*MxWidgetStyle* style)
 Sets the chart's style object.

public *MxWidgetModel* getModel()
 Retruns the model that the chart uses to obtain the data. For charts, this model will
 always be an instance of *MxChartStyle*.

public void setModel(*MxWidgetModel* model)
 Sets the model that the chart uses to obtain the data.

public boolean isRightMouseButtonMenuEnabled()
 Returns true if the rightMouseButton menu is enabled.This menu allows the end-user to
 customize some of the current chart's styles.

public void setRightMouseButtonMenuEnabled(boolean value)
 Enables/disables the rightMouseButton menu.

public void saveTo(java.io.File file)
 Saves the object into the file using file's extension as image type.

public void export()
 Opens a file dialog allowing an end-user to select the file type and location, and then
 saves the object into the selected file using saveTo function.

public void print(String jobName)
 Prints the object

public byte [] getBytes(String type)
 Creates an image of specified type (PNG, JPG, SWF, SVG, PDF, TIF, or another
 supported type) and returns its bytes.

public void addMxComponentListener(*MxComponentListener* l)
 Adds the specified listener to receive events from this component, where
 MxComponentListener is declared as:

 public interface MxComponentListener extends java.util.EventListener {
   public void componentChanged(MxComponentEvent widget) ;
 }

public void removeMxComponentListener(*MxComponentListener* l)
 Removes the specified listener so that it no longer receives events from this component.

APPENDIX B – MxDesignTable and MxDesignPanel API

Class *com.gp.api.swing.designer.MxDesignPanel* can be used to edit instances of MxComponent. MxDesignPanel is a subclass of JSplitPane with the following additional methods:

public void setTarget(MxComponent target)
>   Sets the component to be edited

public void setTarget(MxComponent target)
>   Returns the current component

public MxDesignTable getDesignTable()
>   Returns an instance of MxDesignTable class used by this panel

public static void initGallery()
>   Starts an asynchronous thread that builds charts gallery from the contents of "gallery" folder of the current directory. Gallery folder and its subfolders are scanned for WebCharts3D project files (*.wcp) and the images of these charts are built.

>   If you are planning to use gallery, you might want to call this method on application startup to avoid delays when opening gallery dialog for the first time.

public void showGallery()
>   Displays gallery dialog with charts templates and sets current chart to the selected one.

Class *com.gp.api.swing.designer.MxDesignTablecan* be used to edit arbitrary chart's styles. MxDesignTable is a subclass of JTable with the following additional methods:

public void setExpertMode(boolean value)
>   Sets expert mode that displays the object structure instead of user-friendly dialogs

public boolean getExpertMode()
>   Returns true if expertMode is set, false otherwise

public void setUseColorSpaces(boolean value)
>   When set to true, all color fields upon click display color space popups with standard and custom colors. Otherwise color dialog is displayed.

public boolean isUseColorSpaces()
>   Returns true if colors spaces are used, false otherwise

public void expandAll()
>   Expands all expandable nodes of the table

public void collapseAll()
>   Collapses all expandable nodes of the table

public void setTarget(MxStructure style)
>   Set's the target style to be edited. This style can be either chart style or any other chart substyle, including frame, axis, paint and other styles. When the style is null, the table will not display anything. SetTarget method automatically sets up table's model. You should not call setModel method of this class directly.

public MxStructure getTarget()

> Returns the edited style. MxDesignTable neither guarantees to return the original instance that was passed for editing, nor guarantees to return a true deep copy of this style (i.e. some subobjects from the original style can be reused). If you want to ensure that the entire object is either accepted or discarded, pass a copy of the styles as an argument to setTarget method. For example, it the table is shown in a dialog with "Accept" and "Discard" buttons, use setTarget(style.copy()) instead of setTarget(style), and assign the new styles to the chart only if accept button was pressed. Note that MxDesignTable is not aware of any chart component and if you need to update the actual chart in responce to the table modifications you will need to add a listener to its model.

APPENDIX C – MODEL API

WebCharts3D software uses classes derived from *com.gp.api.model.MxChartModel* class to represent chart's model. To create a model you can either derive a new class from *MxChartModel* and implement required methods or use API functions to create an instance of *MxStandardChartModel* from XML, files or databases, or to dynamically build model.

To create your own chart model, derive a class from *MxChartModel* and implement the following methods:

public abstract int getColCount()
> Returns number of the columns in the model

public abstract int getRowCount()
> Returns number of the rows in the model

public abstract double valueAt(int rowIndex, int colIndex)
> Returns value at a particular row and column. This value of Double.NaN to represents a gap in the data.

public abstract String getColLabelAt(int colIndex)
> Returns column's label

public abstract String getRowLabelAt(int rowIndex)
> Returns row's label

public abstract int getColSelection()
> Returns currently selected column. This method can return -1 or 0 for the models that do not support selection.

public abstract void setColSelection(int colSelection)
> Sets currently selected column

public abstract int getRowSelection()
> Returns currently selected row. This method can return -1 or 0 for the models that do not support selection.

public abstract void setRowSelection(int rowSelection)
> Sets currently selected row

Using Standard Model

MxStandardChartModel class provides the default implementation of MxChartModel and adds a number of API functions that can be used to load, persist and dynamically update models.

The following functions can be used to dynamically update chart's model:

public void insertRow(java.lang.String rowLabel, double[] seq, int rowIndex)
> Insets a new row with specified label at the specified location. When -1 is specified as rowIndex, the new row is appended to the list of the rows. If seq is null or its length is less than the current number of columns, then the rest of the row is filled with Double.NaN values.

public void insertCol(java.lang.String colLabel, double[] seq, int colIndex)
> Insets a new column with specified label at the specified location. When -1 is specified as colIndex, the new column is appended to the list of the columns. If seq is null or its length is less than the current number of rows, then the rest of the column is filled with Double.NaN values.

public void replaceRow(double [] seq, int index)
> Replaces row at specified location with a new one. If seq is null or its length is less than the current number of columns, then the rest of the row is filled with Double.NaN values.

public void replaceCol(double [] seq, int index)
> Replaces column at specified location. If seq is null or its length is less than the current number of rows, then the rest of the column is filled with Double.NaN values.

public void deleteRow(int rowIndex)
> Deletes row at specified location.

public void deleteCol(int colIndex)
> Deletes column at specified location.

public void setRowLabelAt(int rowIndex, java.lang.String rowLabel)
> Changes row's label to provided rowLabel.

public void setColLabelAt(int colIndex, java.lang.String colLabel)
> Changes columns's label to provided colLabel.

public void setValueAt(int rowIndex, int colIndex, double value)
> Updates specified cell with the provided value.

NOTE. MxStandardChartModel does not fire any change events and the chart will not be updated until it is repainted.

WebCharts3D can read data from XML sources, separated files and databases. In all cases, the data can be either in row-by-row or crosstab (three values per row specifying row, column and value) formats. In some cases, database queries or files will contain data in the transposed format, when you would want to plot returned rows as columns and columns as rows. To achieve it, it is often easier to set *isTransposed* attribute on the chart style than to change your data source. The following functions can be used with MxChartModel class to retrieve and persist chart's model.

public static MxStandardChartModel load(
> java.sql.Connection conn, String sql, boolean isCrosstab, boolean hasRowLabels)
>> Creates a new chart's model from data returned by a database query.
>> conn   - Opened database connection
>> sql   - SQL command
>> isCrosstab   - true when the query returns data in crosstab format, false otherwise
>> hasRowLabels - true when the query returns rowLabels in the first column

public static MxStandardChartModel load(
> java.io.Reader rdr, char sep, char quote, boolean isCrosstab, boolean hasColLabels,
> boolean hasRowLabels)
>> Creates a new chart's model from data located in a delimited file. This function can import data from any delimited (comma, tab, semicolon, etc.) file that might have quoted values. For example, to parse file that contains rows like *"Year 2001","5","10"* you should specify

---

',' as a separator and '"' as a quote. To parse file that contains rows like *Year 2001;5;10* you should specify ';' as a separator and '\0' as a quote character.

rdr	- Reader
sep	- Delimiter character
quote	- Quote character or '\0' for the unquoted files
isCrosstab	- true when the query returns data in crosstab format, false otherwise
hasColLabels	- true when the first row contains column labels, false otherwise
hasRowLabels	- true when the first column contains row labels, false otherwise

public String toXML(boolean isCrossTab, String encoding)
> Returns XML represetation of the chart's model in default or crosstab format with XML encoding set to specified encoding.

public String toXML()
> Returns XML represetation of the chart's model in default format.

public static MxStandardChartModel fromXML(String xml, String parser)
> Creates new chart model from the xml representation using a specified parser or built-in parser if the parser is not specified.

APPENDIX D – MODEL XML FORMAT

WebCharts3D software supports two alternative formats for XML representation of chart's model - default and crosstab. Both formats describe a two-dimensional table that is used to create chart's model. Any model can be written as XML, but when a model is recreated from XML WebCharts3D API will return an instance of *MxStandardChartModel*. The examples in this section describe the following table:

	Year2000	Year2001
Assets	100	200
Liabilities	150	300

Default Format

In the default format the table is defined row-by-row:

```
<XML [type="default"]>
  <COL>Column label1</COL>
  <COL>Column label2</COL>
  ...
  <ROW attr1="value" attr2="value" ... >Row label</ROW>
  <ROW attr1="value" attr2="value" ... >Row label</ROW>
  ...
</XML>
```

The number of the <COL> elements in this format should match the number of the attributes in each row. You can specify empty strings as the attributes' values to create gaps in the chart. Optionally, you can omit <COL> elements completely. In this case, the labels are automatically produced from the first row. The shorthand notation is convenient when your column labels do not contain special characters.

Example:

```
<XML>
  <COL>Year2000</COL>
  <COL>Year2001</COL>
  <ROW col0="100.0" col1="200.0">Assets</ROW>
  <ROW col0="150.0" col1="300.0">Liabilities</ROW>
</XML>
```
defines the same table as
```
<XML>
  <ROW Year2000="100.0" Year20001="200.0">Assets</ROW>
  <ROW Year2000="150.0" Year2001="300.0">Liabilities</ROW>
</XML>
```

Crosstab Format

In the crosstab format each ROW element of the xml specifies value for the particular row and column.

```
<XML type="crosstab">
  <ROW rowLabel="rowLabel" colLabel="colLabel" value="value"/>
  <ROW rowLabel="rowLabel" colLabel="colLabel" value="value"/>
  ...
</XML>
```

Example:

```
<XML type="crosstab">
  <ROW rowLabel="Assets" colLabel="Year2000" value="100.0"/>
  <ROW rowLabel="Assets" colLabel="Year2001" value="200.0"/>
  <ROW rowLabel="Liabilities" colLabel="Year2000" value="150.0"/>
  <ROW rowLabel="Liabilities" colLabel="Year2001" value="300.0"/>
</XML>
```

APPENDIX E – SAMPLE PAGE USING INDIRECT METHOD

```
<%@ page import = "com.gp.api.jsp.MxServerComponent" %>
<%@ page import = "com.gp.api.jsp.MxChartDescription"%>
<%
   MxServerComponent  svr = MxServerComponent.getDefaultInstance(application);

   MxChartDescription myChart = svr.newImageSpec();
   myChart.width = 385  ;
   myChart.height= 309 ;
   myChart.type = "PNG"  ;
   myChart.style = " <frameChart is3D=\"false\"> <frame xDepth=\"3\" yDepth=\"1\"
leftAxisPlacement=\"Back\" isHStripVisible=\"true\"> <background minColor=\"#FDFEF6\"/>
</frame> <xAxis> <labelStyle isMultilevel=\"true\"/> </xAxis> <elements place=\"Stacked\"
shape=\"Line\" drawShadow=\"true\"> <morph morph=\"Grow\"/> </elements> </frameChart>" ;
   myChart.linkTo("c:\\test\\test.csv",',',",true,false,false,null);
   out.write(svr.getImageTag(myChart,"/getImage.jsp?image="));
%>
```

APPENDIX F – SAMPLE PAGE USING DIRECT METHOD

```
<%@ page import = "com.gp.api.jsp.*" %><%
   MxServerComponent  svr = MxServerComponent.getDefaultInstance(application);

   MxChartDescription myChart = svr.newImageSpec();
   myChart.width = 385  ;
   myChart.height= 309 ;
   myChart.type = "png"  ;
   myChart.style = " <frameChart is3D=\"false\"> <frame xDepth=\"3\" yDepth=\"1\"
leftAxisPlacement=\"Back\" isHStripVisible=\"true\"> <background
minColor=\"#FDFEF6\"/> </frame> <xAxis> <labelStyle isMultilevel=\"true\"/>
</xAxis> <elements place=\"Stacked\" shape=\"Line\" drawShadow=\"true\"> <morph
morph=\"Grow\"/> </elements> </frameChart>" ;
   myChart.linkTo("c:\\test\\test.csv",',',",true,false,false,null);
   svr.writeBytesTo(myChart,response);
%>
```

APPENDIX G – SAMPLE SERVER CONFIGURATION FILE

```
<server
      license="0000-0000-0000-0000"
      image="PNG"
      cache="Memory"
      minTimeout="5000"
      maxTimeout="30000"
      cacheSize="100" throwException="true" maxThreads="20"
      gcInterval="3000"
      compression="Default"
      useBuiltinXmlParser="true"
      xmlParser="org.apache.xerces.parsers.SAXParser"
      useFlashFonts="false"
      >
      <folders>
            <map key="images" value="Images"/>
      </folders>
      <mime>
            <map key="application/pdf" value=".pdf"/>
            <map key="image/png" value=".png"/>
            <map key="image/jpeg" value=".jpg;.jpeg"/>
            <map key="text/vnd.wap.wml" value=".wml"/>
            <map key="application/x-shockwave-flash" value=".swf"/>
            <map key="image/svg-xml" value=".svg"/>
            <map key="text/html" value=".htm;.html"/>
            <map key="image/tiff" value=".tif;.tiff"/>
            <map key="text/text" value=".txt;.ini"/>
            <map key="image/vnd.wap.wbmp" value=".wbmp"/>
            <map key="image/gif" value=".gif"/>
      </mime>
      <plugins>
            <map key="tif"><!-- TIFF plugin code --></map>
            <map key="swf"><!-- FLASH plugin code --></map>
            <map key="pdf"><!-- PDF plugin code --></map>
            <map key="map"><!-- Image map code --></map>
            <map key="svg"><!-- SVG plugin code --></map>
      </plugins>
</server>
```

APPENDIX H – STYLE API

WebCharts3D software uses classes derived from com.gp.api.styles.MxWidgetStyle class to represent chart's styles. The particular list of the available styles depends upon your configuration.

To create a style you can either use a default constructor and modify the attributes of the style or create a style from XML representation. All style classes can be persisted as XML with an optional DTD or XSD Schema using the following methods:

public static MxWidgetStyle read(InputStream is, String parser)
>   Creates new widget style from the stream using provided parser (or null for the built-in parser). This function will attempt to auto-detect the stream encoding by using its first bytes.

public static MxWidgetStyle read(String xml, String parser)
>   Creates new widget style from xml using provided parser (or null for the built-in parser)

public String toXML(String encoding, boolean expanded)
>   Returns a string representation of this style with XML's encoding attribute set to encoding. Note that when saving this string into a file, you will need to use the same encoding as was used to produce this XML

public String toXML()
>   Returns a string representation of this style with UTF-8 encoding

Generally it is much easier to create styles from XML produced by the designer than to code styles manually.

Example:

```
MxPieChartStyle style = new MxPieChartStyle();
style.type = MxPieChartStyle.Type.PIE;
```

is the same as:

```
MxChartStyle style = MxChartStyle.read("<pieChart type="Pie"/>",null);
```

WebCharts3D uses a large number of additional classes to define chart's constants and element properties. To see the list of these parameters, please refer to Style XSD Schema.

Parameters

Some of the WebCharts3D text elements can contain special parameters in the format ${name} or $(name). (The latter notation was introduced to avoid conflicts with other software that might use the former syntax for its own needs). These parameters are expanded during run-time and replaced with their corresponding values. For example, you can specify:

> $(colLabel) $(value) is $(colPercent) of $(colTotal)

to display the name of the column, the value of the element, percent and column's total value.

The following table summarizes supported parameters:

```
colIndex    - Index of the column
rowIndex    - Index of the row
colLabel    - Label of the column
rowLabel    - Label of the row
value       - Value formatted using appropriate yAxis labelFormat
nextvalue   - Value in the next series formatted using appropriate yAxis labelFormat
colTotal    - Sum of all absolute values in this column
rowTotal    - Sum of all absolute values in this row
colPercent  - Percent relative to the column total
rowPercent  - Percent relative to the row total
colTitle    - column title (XAxis title)
rowTitle    - row title (seriesTitle attribute)
valueTitle  - value title (YAxis title)
```

APPENDIX I – Style XSD Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
        <xsd:documentation xml:lang="en">
                WebCharts3D v5.0 Schema
                Copyright (c) GreeenPoint, Inc. 2000-2004. All rights reserved.
        </xsd:documentation>
</xsd:annotation>

<xsd:simpleType name="font">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[^-]*-[0-9]*(-bold)?(-italic)?"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="color">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="#?([0-9a-fA-F]{2})?[0-9a-fA-
F]{6}|red|cyan|white|pink|yellow|light_gray|gray|black|green|dark_gray|orange|lightgray|blue|mage
nta|darkgray"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="FrameStyle_FrameDecoration">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Simple"/>
                <xsd:enumeration value="Tray"/>
                <xsd:enumeration value="Thick"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="FrameStyle_AxisPlacement">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Default"/>
                <xsd:enumeration value="Front"/>
                <xsd:enumeration value="Back"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="BackgroundStyle_BackgroundPaint">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="PlainColor"/>
                <xsd:enumeration value="Transparent"/>
                <xsd:enumeration value="VerticalGradient"/>
                <xsd:enumeration value="HorizontalGradient"/>
                <xsd:enumeration value="DiagonalLtRbGradient"/>
                <xsd:enumeration value="DiagonalRtLbGradient"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="BackgroundStyle_BackgroundImage">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Centered"/>
                <xsd:enumeration value="Scaled"/>
                <xsd:enumeration value="LeftTop"/>
                <xsd:enumeration value="LeftBottom"/>
```

```
                <xsd:enumeration value="RightTop"/>
                <xsd:enumeration value="RightBottom"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="BackgroundStyle">
        <xsd:attribute name="type" type="BackgroundStyle_BackgroundPaint"/>
        <xsd:attribute name="minColor" type="color"/>
        <xsd:attribute name="maxColor" type="color"/>
        <xsd:attribute name="imagePlacement" type="BackgroundStyle_BackgroundImage"/>
        <xsd:attribute name="imageLocation" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="FrameStyle">
        <xsd:sequence>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="topRatio" type="xsd:integer"/>
        <xsd:attribute name="topSeries" type="xsd:integer"/>
        <xsd:attribute name="xDepth" type="xsd:integer"/>
        <xsd:attribute name="yDepth" type="xsd:integer"/>
        <xsd:attribute name="xTilt" type="xsd:integer"/>
        <xsd:attribute name="type" type="FrameStyle_FrameDecoration"/>
        <xsd:attribute name="outline" type="color"/>
        <xsd:attribute name="lightColor" type="color"/>
        <xsd:attribute name="darkColor" type="color"/>
        <xsd:attribute name="frameSize" type="xsd:integer"/>
        <xsd:attribute name="dashSize" type="xsd:integer"/>
        <xsd:attribute name="xAxisInside" type="xsd:boolean"/>
        <xsd:attribute name="leftAxisPlacement" type="FrameStyle_AxisPlacement"/>
        <xsd:attribute name="rightAxisPlacement" type="FrameStyle_AxisPlacement"/>
        <xsd:attribute name="isHGridVisible" type="xsd:boolean"/>
        <xsd:attribute name="isVGridVisible" type="xsd:boolean"/>
        <xsd:attribute name="isHStripVisible" type="xsd:boolean"/>
        <xsd:attribute name="isVStripVisible" type="xsd:boolean"/>
        <xsd:attribute name="gridColor" type="color"/>
        <xsd:attribute name="stripColor" type="color"/>
</xsd:complexType>
<xsd:simpleType name="AxisStyle_ScaleType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Category"/>
                <xsd:enumeration value="Scale"/>
                <xsd:enumeration value="Logarithmic"/>
                <xsd:enumeration value="DateTime"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="FormatStyle_Style">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Number"/>
                <xsd:enumeration value="Integer"/>
                <xsd:enumeration value="Percent"/>
                <xsd:enumeration value="Currency"/>
                <xsd:enumeration value="Pattern"/>
                <xsd:enumeration value="Choice"/>
                <xsd:enumeration value="DateTimePattern"/>
                <xsd:enumeration value="ShortDate"/>
                <xsd:enumeration value="MediumDate"/>
```

```
                    <xsd:enumeration value="LongDate"/>
                    <xsd:enumeration value="FullDate"/>
                    <xsd:enumeration value="ShortTime"/>
                    <xsd:enumeration value="MediumTime"/>
                    <xsd:enumeration value="LongTime"/>
                    <xsd:enumeration value="FullTime"/>
                    <xsd:enumeration value="ShortDateTime"/>
                    <xsd:enumeration value="MediumDateTime"/>
            </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="LocaleStyle">
            <xsd:attribute name="lang" type="xsd:string"/>
            <xsd:attribute name="country" type="xsd:string"/>
            <xsd:attribute name="variant" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="FormatStyle">
            <xsd:annotation>
                    <xsd:documentation>
Defines an input or output data format. This object can represent both
numeric and date/time format. For the detailed description of the pattern
format and other attributes please refer to Java documentation for
DecimalFormat and SimpleDateFormat classes.
                    </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                    <xsd:element name="locale" type="LocaleStyle" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="style" type="FormatStyle_Style"/>
            <xsd:attribute name="pattern" type="xsd:string"/>
            <xsd:attribute name="timezone" type="xsd:string"/>
            <xsd:attribute name="lenient" type="xsd:boolean"/>
            <xsd:attribute name="currency" type="xsd:string"/>
            <xsd:attribute name="groupingSize" type="xsd:integer"/>
            <xsd:attribute name="minIntegerDigits" type="xsd:integer"/>
            <xsd:attribute name="maxIntegerDigits" type="xsd:integer"/>
            <xsd:attribute name="minFractionDigits" type="xsd:integer"/>
            <xsd:attribute name="maxFractionDigits" type="xsd:integer"/>
            <xsd:attribute name="multiplier" type="xsd:integer"/>
</xsd:complexType>
<xsd:simpleType name="DateTimeScaleStyle_CalendarUnit">
            <xsd:annotation>
                    <xsd:documentation>
Describes the units of increment for date/time scales.
                    </xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Year"/>
                    <xsd:enumeration value="Month"/>
                    <xsd:enumeration value="Week"/>
                    <xsd:enumeration value="Day"/>
                    <xsd:enumeration value="Hour"/>
                    <xsd:enumeration value="Minute"/>
                    <xsd:enumeration value="Second"/>
                    <xsd:enumeration value="Millisecond"/>
            </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="DateTimeScaleStyle">
        <xsd:attribute name="majorUnit" type="DateTimeScaleStyle_CalendarUnit"/>
        <xsd:attribute name="majorStep" type="xsd:integer"/>
        <xsd:attribute name="minorUnit" type="DateTimeScaleStyle_CalendarUnit"/>
        <xsd:attribute name="minorStep" type="xsd:integer"/>
</xsd:complexType>
<xsd:simpleType name="LogScaleStyle_LogScaleType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="IncludeEven"/>
                <xsd:enumeration value="IncludeMinor"/>
                <xsd:enumeration value="ExcludeMinor"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="LogScaleStyle">
        <xsd:attribute name="type" type="LogScaleStyle_LogScaleType"/>
        <xsd:attribute name="base" type="xsd:integer"/>
</xsd:complexType>
<xsd:simpleType name="StandardConstants_TextOrientation">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Horizontal"/>
                <xsd:enumeration value="Vertical"/>
                <xsd:enumeration value="Slanted"/>
                <xsd:enumeration value="Parallel"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="AxisLabelStyle">
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
        <xsd:attribute name="isMultilevel" type="xsd:boolean"/>
        <xsd:attribute name="orientation" type="StandardConstants_TextOrientation"/>
</xsd:complexType>
<xsd:complexType name="AxisTitleStyle" mixed="true">
        <xsd:annotation>
                <xsd:documentation>
This style describes the axis's title. When font or/and color are
not specified, the default values for the entire chart are used.
                </xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="isVisible" type="xsd:boolean"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:complexType name="AxisStyle">
        <xsd:sequence>
                <xsd:element name="labelFormat" type="FormatStyle" minOccurs="0"/>
                <xsd:element name="parseFormat" type="FormatStyle" minOccurs="0"/>
                <xsd:element name="dateTimeStyle" type="DateTimeScaleStyle"
minOccurs="0"/>
                <xsd:element name="logStyle" type="LogScaleStyle" minOccurs="0"/>
                <xsd:element name="labelStyle" type="AxisLabelStyle" minOccurs="0"/>
                <xsd:element name="titleStyle" type="AxisTitleStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="type" type="AxisStyle_ScaleType"/>
        <xsd:attribute name="isVisible" type="xsd:boolean"/>
        <xsd:attribute name="isAbsolute" type="xsd:boolean"/>
        <xsd:attribute name="scaleMin" type="xsd:string"/>
```

```
                <xsd:attribute name="scaleMax" type="xsd:string"/>
                <xsd:attribute name="labelCount" type="xsd:integer"/>
                <xsd:attribute name="isReversed" type="xsd:boolean"/>
                <xsd:attribute name="isBucketed" type="xsd:boolean"/>
</xsd:complexType>
<xsd:simpleType name="DataLabelStyle_DataLabels">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Value"/>
                <xsd:enumeration value="RowLabel"/>
                <xsd:enumeration value="ColumnLabel"/>
                <xsd:enumeration value="Pattern"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="DataLabelStyle_Placement">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Inside"/>
                <xsd:enumeration value="Outside"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StandardConstants_Decoration">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Simple"/>
                <xsd:enumeration value="Double"/>
                <xsd:enumeration value="Bevel"/>
                <xsd:enumeration value="Round"/>
                <xsd:enumeration value="Shadow"/>
                <xsd:enumeration value="RoundShadow"/>
                <xsd:enumeration value="FrameOpened"/>
                <xsd:enumeration value="FrameClosed"/>
                <xsd:enumeration value="FrameTopBottom"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="DecorationStyle">
        <xsd:attribute name="style" type="StandardConstants_Decoration"/>
        <xsd:attribute name="foreColor" type="color"/>
        <xsd:attribute name="backColor" type="color"/>
</xsd:complexType>
<xsd:complexType name="DataLabelStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="style" type="DataLabelStyle_DataLabels"/>
        <xsd:attribute name="placement" type="DataLabelStyle_Placement"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:simpleType name="StandardConstants_Placement">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Top"/>
                <xsd:enumeration value="Bottom"/>
                <xsd:enumeration value="Left"/>
                <xsd:enumeration value="Right"/>
```

```
                    </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="StandardConstants_VAlignment">
                    <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="Top"/>
                                <xsd:enumeration value="Middle"/>
                                <xsd:enumeration value="Bottom"/>
                    </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="StandardConstants_HAlignment">
                    <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="Left"/>
                                <xsd:enumeration value="Right"/>
                                <xsd:enumeration value="Center"/>
                    </xsd:restriction>
        </xsd:simpleType>
        <xsd:complexType name="LegendStyle">
                    <xsd:sequence>
                                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                    </xsd:sequence>
                    <xsd:attribute name="spacing" type="xsd:integer"/>
                    <xsd:attribute name="allowSpan" type="xsd:boolean"/>
                    <xsd:attribute name="equalCols" type="xsd:boolean"/>
                    <xsd:attribute name="isVisible" type="xsd:boolean"/>
                    <xsd:attribute name="placement" type="StandardConstants_Placement"/>
                    <xsd:attribute name="valign" type="StandardConstants_VAlignment"/>
                    <xsd:attribute name="halign" type="StandardConstants_HAlignment"/>
                    <xsd:attribute name="background" type="color"/>
                    <xsd:attribute name="font" type="font"/>
                    <xsd:attribute name="foreground" type="color"/>
                    <xsd:attribute name="isMultiline" type="xsd:boolean"/>
        </xsd:complexType>
        <xsd:simpleType name="ElementStyle_Place">
                    <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="Default"/>
                                <xsd:enumeration value="Stacked"/>
                                <xsd:enumeration value="Percent"/>
                                <xsd:enumeration value="Clustered"/>
                    </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="ElementStyle_Shape">
                    <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="Bar"/>
                                <xsd:enumeration value="Line"/>
                                <xsd:enumeration value="Pyramid"/>
                                <xsd:enumeration value="Area"/>
                                <xsd:enumeration value="Curve"/>
                                <xsd:enumeration value="Step"/>
                                <xsd:enumeration value="PyramidCut"/>
                                <xsd:enumeration value="Scatter"/>
                                <xsd:enumeration value="Prism"/>
                                <xsd:enumeration value="Column"/>
                                <xsd:enumeration value="FloatingBars"/>
                                <xsd:enumeration value="Band"/>
                                <xsd:enumeration value="FloatingColumn"/>
                                <xsd:enumeration value="HighLowClose"/>
```

```
                <xsd:enumeration value="HighLowOpenClose"/>
                <xsd:enumeration value="HighLowOpenCloseExcel"/>
                <xsd:enumeration value="Box-Whisker"/>
                <xsd:enumeration value="Bubble"/>
                <xsd:enumeration value="Ring"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="MovieStyle">
        <xsd:attribute name="framesPerSecond" type="xsd:integer"/>
        <xsd:attribute name="frameCount" type="xsd:integer"/>
        <xsd:attribute name="stageDelay" type="xsd:integer"/>
        <xsd:attribute name="replayDelay" type="xsd:integer"/>
</xsd:complexType>
<xsd:simpleType name="MorphStyle_MorphStyle">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Blur"/>
                <xsd:enumeration value="Grow"/>
                <xsd:enumeration value="Both"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="MorphStyle_MorphStage">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Rows"/>
                <xsd:enumeration value="Cols"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="MorphStyle">
        <xsd:attribute name="morph" type="MorphStyle_MorphStyle"/>
        <xsd:attribute name="stage" type="MorphStyle_MorphStage"/>
</xsd:complexType>
<xsd:simpleType name="MarkerStyle_Marker">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Circle"/>
                <xsd:enumeration value="Rectangle"/>
                <xsd:enumeration value="Triangle"/>
                <xsd:enumeration value="Diamond"/>
                <xsd:enumeration value="LetterX"/>
                <xsd:enumeration value="Snow"/>
                <xsd:enumeration value="RCROSS"/>
                <xsd:enumeration value="MCROSS"/>
                <xsd:enumeration value="Bitmap"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="MarkerStyle">
        <xsd:attribute name="type" type="MarkerStyle_Marker"/>
        <xsd:attribute name="bitmap" type="xsd:string"/>
</xsd:complexType>
<xsd:simpleType name="TextureStyle">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Squares"/>
                <xsd:enumeration value="Lines"/>
                <xsd:enumeration value="Beans"/>
                <xsd:enumeration value="Pyramids"/>
```

```xml
                <xsd:enumeration value="Diamonds"/>
                <xsd:enumeration value="Dots"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ColumnStyle_PaintStyle">
        <xsd:attribute name="color" type="color"/>
        <xsd:attribute name="texture" type="TextureStyle"/>
        <xsd:attribute name="image" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="ColumnStyle_DataLabelStyle" mixed="true">
</xsd:complexType>
<xsd:complexType name="SeriesStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="morph" type="MorphStyle" minOccurs="0"/>
                <xsd:element name="marker" type="MarkerStyle" minOccurs="0"/>
                <xsd:element name="paint" type="ColumnStyle_PaintStyle" minOccurs="0"/>
                <xsd:element name="dataLabel" type="ColumnStyle_DataLabelStyle"
minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="index" type="xsd:int"/>
        <xsd:attribute name="place" type="ElementStyle_Place"/>
        <xsd:attribute name="shape" type="ElementStyle_Shape"/>
        <xsd:attribute name="isSecondAxis" type="xsd:boolean"/>
        <xsd:attribute name="forceBreak" type="xsd:boolean"/>
        <xsd:attribute name="lineWidth" type="xsd:integer"/>
        <xsd:attribute name="action" type="xsd:string"/>
        <xsd:attribute name="target" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="ColumnStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="paint" type="ColumnStyle_PaintStyle" minOccurs="0"/>
                <xsd:element name="dataLabel" type="ColumnStyle_DataLabelStyle"
minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="index" type="xsd:int"/>
        <xsd:attribute name="action" type="xsd:string"/>
        <xsd:attribute name="target" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="ElementStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="movie" type="MovieStyle" minOccurs="0"/>
                <xsd:element name="morph" type="MorphStyle" minOccurs="0"/>
                <xsd:element name="series" type="SeriesStyle" maxOccurs="unbounded"
minOccurs="0"/>
                <xsd:element name="column" type="ColumnStyle" maxOccurs="unbounded"
minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="action" type="xsd:string"/>
        <xsd:attribute name="target" type="xsd:string"/>
        <xsd:attribute name="place" type="ElementStyle_Place"/>
        <xsd:attribute name="shape" type="ElementStyle_Shape"/>
        <xsd:attribute name="outline" type="color"/>
        <xsd:attribute name="markerSize" type="xsd:integer"/>
        <xsd:attribute name="shapeSize" type="xsd:integer"/>
        <xsd:attribute name="lineWidth" type="xsd:integer"/>
        <xsd:attribute name="drawOutline" type="xsd:boolean"/>
```

```
        <xsd:attribute name="drawShadow" type="xsd:boolean"/>
        <xsd:attribute name="showMarkers" type="xsd:boolean"/>
        <xsd:attribute name="fixedWidth" type="xsd:integer"/>
</xsd:complexType>
<xsd:simpleType name="TableStyle_TableBorder">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Financial"/>
                <xsd:enumeration value="Outer"/>
                <xsd:enumeration value="Inner"/>
                <xsd:enumeration value="Grid"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="TableStyle_LabelTextStyle">
        <xsd:attribute name="index" type="xsd:int"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:complexType name="TableStyle_LabelDecoration">
        <xsd:sequence>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="elements" type="TableStyle_LabelTextStyle"
maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="hAlign" type="StandardConstants_HAlignment"/>
        <xsd:attribute name="vAlign" type="StandardConstants_VAlignment"/>
        <xsd:attribute name="isVisible" type="xsd:boolean"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:complexType name="TableStyle_CellDecoration">
        <xsd:sequence>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="hAlign" type="StandardConstants_HAlignment"/>
        <xsd:attribute name="vAlign" type="StandardConstants_VAlignment"/>
        <xsd:attribute name="isVisible" type="xsd:boolean"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:complexType name="TableStyle_Heatmap">
        <xsd:attribute name="isEnabled" type="xsd:boolean"/>
        <xsd:attribute name="isAbsolute" type="xsd:boolean"/>
        <xsd:attribute name="minColor" type="color"/>
        <xsd:attribute name="maxColor" type="color"/>
        <xsd:attribute name="minLevel" type="xsd:float"/>
        <xsd:attribute name="maxLevel" type="xsd:float"/>
</xsd:complexType>
<xsd:complexType name="ChartStyle_TableStyle">
        <xsd:sequence>
```

```xml
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="colLabels" type="TableStyle_LabelDecoration"
minOccurs="0"/>
                <xsd:element name="rowLabels" type="TableStyle_LabelDecoration"
minOccurs="0"/>
                <xsd:element name="cells" type="TableStyle_CellDecoration" minOccurs="0"/>
                <xsd:element name="format" type="FormatStyle" minOccurs="0"/>
                <xsd:element name="heatmap" type="TableStyle_Heatmap" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="isVisible" type="xsd:boolean"/>
        <xsd:attribute name="placement" type="StandardConstants_Placement"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="cellSpacing" type="xsd:integer"/>
        <xsd:attribute name="cellPadding" type="xsd:integer"/>
        <xsd:attribute name="rowZebraBackground" type="color"/>
        <xsd:attribute name="colZebraBackground" type="color"/>
        <xsd:attribute name="negativeForeground" type="color"/>
        <xsd:attribute name="border" type="TableStyle_TableBorder"/>
</xsd:complexType>
<xsd:complexType name="TitleStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="placement" type="StandardConstants_Placement"/>
        <xsd:attribute name="valign" type="StandardConstants_VAlignment"/>
        <xsd:attribute name="halign" type="StandardConstants_HAlignment"/>
        <xsd:attribute name="spacing" type="xsd:integer"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:simpleType name="PopupStyle_Popup">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Disabled"/>
                <xsd:enumeration value="MouseOver"/>
                <xsd:enumeration value="MouseDown"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PopupStyle">
        <xsd:attribute name="showOn" type="PopupStyle_Popup"/>
        <xsd:attribute name="decoration" type="StandardConstants_Decoration"/>
        <xsd:attribute name="background" type="color"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:simpleType name="PaletteStyle">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Standard"/>
                <xsd:enumeration value="Pastel"/>
                <xsd:enumeration value="Sunburn"/>
                <xsd:enumeration value="Dawn"/>
                <xsd:enumeration value="Modern"/>
                <xsd:enumeration value="Steel"/>
                <xsd:enumeration value="Transluent"/>
```

```
                <xsd:enumeration value="PastelTransluent"/>
                <xsd:enumeration value="SunburnTransluent"/>
                <xsd:enumeration value="DawnTransluent"/>
                <xsd:enumeration value="ModernTransluent"/>
                <xsd:enumeration value="SteelTransluent"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="ElementStyle_Paint">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Plain"/>
                <xsd:enumeration value="Shade"/>
                <xsd:enumeration value="Light"/>
                <xsd:enumeration value="Pattern"/>
                <xsd:enumeration value="Image"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PaintStyle">
        <xsd:attribute name="palette" type="PaletteStyle"/>
        <xsd:attribute name="paint" type="ElementStyle_Paint"/>
        <xsd:attribute name="isVertical" type="xsd:boolean"/>
        <xsd:attribute name="min" type="xsd:integer"/>
        <xsd:attribute name="max" type="xsd:integer"/>
        <xsd:attribute name="image" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="InsetStyle">
        <xsd:attribute name="left" type="xsd:integer"/>
        <xsd:attribute name="top" type="xsd:integer"/>
        <xsd:attribute name="right" type="xsd:integer"/>
        <xsd:attribute name="bottom" type="xsd:integer"/>
</xsd:complexType>
<xsd:complexType name="FrameChartStyle">
        <xsd:sequence>
                <xsd:element name="frame" type="FrameStyle" minOccurs="0"/>
                <xsd:element name="xAxis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="yAxis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="yAxis2" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="topYAxis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="topYAxis2" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
```

```
<xsd:simpleType name="PieChartStyle_Depth">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Thick"/>
                <xsd:enumeration value="Double"/>
                <xsd:enumeration value="Normal"/>
                <xsd:enumeration value="Thin"/>
                <xsd:enumeration value="Plain"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="PieChartStyle_Style">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Default"/>
                <xsd:enumeration value="Solid"/>
                <xsd:enumeration value="Sliced"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="PieChartStyle_Type">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Pie"/>
                <xsd:enumeration value="SmallNut"/>
                <xsd:enumeration value="MediumNut"/>
                <xsd:enumeration value="LargeNut"/>
                <xsd:enumeration value="Doughnut"/>
                <xsd:enumeration value="RaisedDoughnut"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PieChartStyle">
        <xsd:sequence>
                <xsd:element name="axis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="depth" type="PieChartStyle_Depth"/>
        <xsd:attribute name="style" type="PieChartStyle_Style"/>
        <xsd:attribute name="type" type="PieChartStyle_Type"/>
        <xsd:attribute name="angle" type="xsd:integer"/>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="RadarChartStyle">
        <xsd:sequence>
                <xsd:element name="axis" type="AxisStyle" minOccurs="0"/>
```

```
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="dashSize" type="xsd:integer"/>
        <xsd:attribute name="fill" type="xsd:boolean"/>
        <xsd:attribute name="gridColor" type="color"/>
        <xsd:attribute name="isGridVisible" type="xsd:boolean"/>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>

<xsd:simpleType name="DialChartStyle_DialType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Dial180"/>
                <xsd:enumeration value="Dial240"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="DialChartStyle">
        <xsd:sequence>
                <xsd:element name="axis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="dial" type="DialChartStyle_DialType"/>
        <xsd:attribute name="minorTicksVisible" type="xsd:boolean"/>
        <xsd:attribute name="inside" type="color"/>
        <xsd:attribute name="hand" type="color"/>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="SlideStyle">
```

```xsd
        <xsd:sequence>
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>

<xsd:simpleType name="StatisticalChartStyle_GridStyle">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Unit"/>
                <xsd:enumeration value="Stddev"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StatisticalChartStyle_ChartType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Histogram"/>
                <xsd:enumeration value="Timeline"/>
                <xsd:enumeration value="Profile"/>
                <xsd:enumeration value="Regression"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StatisticalChartStyle_VarType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Draw"/>
                <xsd:enumeration value="Fill"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="StatisticalChartStyle_ValueAtRisk">
        <xsd:attribute name="type" type="StatisticalChartStyle_VarType"/>
        <xsd:attribute name="level" type="xsd:float"/>
</xsd:complexType>
<xsd:complexType name="StatisticalChartStyle">
        <xsd:sequence>
                <xsd:element name="xAxis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="xAxis2" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="yAxis" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="yAxis2" type="AxisStyle" minOccurs="0"/>
                <xsd:element name="valueAtRisk" type="StatisticalChartStyle_ValueAtRisk"
minOccurs="0"/>
                <xsd:element name="dataLabels" type="DataLabelStyle" minOccurs="0"/>
                <xsd:element name="legend" type="LegendStyle" minOccurs="0"/>
```

```
                <xsd:element name="elements" type="ElementStyle" minOccurs="0"/>
                <xsd:element name="table" type="ChartStyle_TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="dashSize" type="xsd:integer"/>
        <xsd:attribute name="gridStyle" type="StatisticalChartStyle_GridStyle"/>
        <xsd:attribute name="type" type="StatisticalChartStyle_ChartType"/>
        <xsd:attribute name="is3D" type="xsd:boolean"/>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="seriesTitle" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>


<xsd:complexType name="TableStyle">
        <xsd:sequence>
                <xsd:element name="colLabels" type="TableStyle_LabelDecoration"
minOccurs="0"/>
                <xsd:element name="rowLabels" type="TableStyle_LabelDecoration"
minOccurs="0"/>
                <xsd:element name="cells" type="TableStyle_CellDecoration" minOccurs="0"/>
                <xsd:element name="format" type="FormatStyle" minOccurs="0"/>
                <xsd:element name="heatmap" type="TableStyle_Heatmap" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="isTransposed" type="xsd:boolean"/>
        <xsd:attribute name="cellSpacing" type="xsd:integer"/>
        <xsd:attribute name="cellPadding" type="xsd:integer"/>
        <xsd:attribute name="rowZebraBackground" type="color"/>
        <xsd:attribute name="colZebraBackground" type="color"/>
        <xsd:attribute name="negativeForeground" type="color"/>
        <xsd:attribute name="border" type="TableStyle_TableBorder"/>
</xsd:complexType>
<xsd:complexType name="TableChartStyle" mixed="true">
        <xsd:sequence>
                <xsd:element name="table" type="TableStyle" minOccurs="0"/>
                <xsd:element name="title" type="TitleStyle" minOccurs="0"/>
                <xsd:element name="background" type="BackgroundStyle" minOccurs="0"/>
                <xsd:element name="decoration" type="DecorationStyle" minOccurs="0"/>
                <xsd:element name="popup" type="PopupStyle" minOccurs="0"/>
                <xsd:element name="paint" type="PaintStyle" minOccurs="0"/>
                <xsd:element name="insets" type="InsetStyle" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="action" type="xsd:string"/>
        <xsd:attribute name="target" type="xsd:string"/>
        <xsd:attribute name="font" type="font"/>
        <xsd:attribute name="foreground" type="color"/>
        <xsd:attribute name="isMultiline" type="xsd:boolean"/>
</xsd:complexType>
<xsd:element name="frameChart" type="FrameChartStyle"/>
<xsd:element name="pieChart" type="PieChartStyle"/>
```

```
<xsd:element name="radarChart" type="RadarChartStyle"/>
<xsd:element name="dialChart" type="DialChartStyle"/>
<xsd:element name="slide" type="SlideStyle"/>
<xsd:element name="statistical" type="StatisticalChartStyle"/>
<xsd:element name="table" type="TableChartStyle"/>
</xsd:schema>
```