

Visual Mining, a division of Tervela, Inc.

# **CDL Reference Guide**

**A Guide to the Chart Definition Language  
Used in Visual Mining Products  
Version 7.2  
Summer 2015**

# Table of Contents

<b>TABLE OF CONTENTS .....</b>	<b>2</b>
<b>1. AN OVERVIEW OF CHART DEFINITION LANGUAGE (CDL) .....</b>	<b>4</b>
1.1 CDL STATEMENTS .....	4
<i>Where Attributes Fit In .....</i>	<i>5</i>
<i>CDL Statement Types.....</i>	<i>6</i>
1.2 VISUAL MINING'S SUITE OF CHART GENERATION SOLUTIONS .....	7
<i>NetCharts Pro.....</i>	<i>7</i>
<i>NetCharts Designer .....</i>	<i>7</i>
<i>NetCharts Server.....</i>	<i>8</i>
<i>NetCharts Performance Dashboards.....</i>	<i>8</i>
<b>2. NETCHARTS PRO APPLETS AND CDL .....</b>	<b>9</b>
CONFIGURING NETCHARTS PRO APPLETS .....	9
STANDARD HTML-STYLE PARAMETER USAGE.....	9
USING A PARAMETER SCRIPT TO SPECIFY CDL PARAMETERS.....	10
POINTING TO PARAMETERS WITH A URL .....	10
PUTTING PARAMETERS IN AN INCLUDE FILE.....	11
USING A PARAMETER SERVER .....	12
<i>A Technical Note about Server Connection Processing .....</i>	<i>13</i>
DYNAMIC UPDATES FROM ANOTHER APPLET .....	13
<i>Loading Data Parameters.....</i>	<i>14</i>
DYNAMIC UPDATES FROM JAVASCRIPT .....	14
<b>3. ANATOMY OF A CHART .....</b>	<b>15</b>
<b>4. TYPES OF AXES .....</b>	<b>17</b>
STANDARD AXES .....	17
MULTI-SCALED SINGLE AXES.....	17
MULTIPLE AXES IN A SINGLE DIRECTION .....	17
VARIABLE AXIS LABELING .....	17
LOGARITHMIC AXES .....	18
<b>5. CDL PARAMETERS ARRANGED BY CHART TYPE .....</b>	<b>19</b>
BAR CHART AND 3DBAR CHART .....	19
BOX CHART .....	21
BUBBLE CHART .....	22
COMBO CHART .....	23
DIAGRAM/MAP CHART .....	25
DIAL CHART .....	26
HEAT MAP CHART .....	28
HISTOGRAM CHART .....	29
LINE CHART.....	30
MULTIPIE CHART.....	32
PARETO CHART.....	33
PIE CHART .....	35
POLAR CHART.....	36
RADAR CHART.....	37
STOCK CHART.....	39
STRIP CHART .....	40
TIME CHART .....	42

**Table of Contents**

---

X-Y CHART ..... 42

**6. CDL PARAMETERS ARRANGED BY FUNCTION ..... 44**

    ACTIVE LABELS AND DRILLDOWN ..... 44

*Parameters Involved, Alphabetically* ..... 44

    AXIS MODIFICATIONS ..... 45

*Parameters Involved, Alphabetically* ..... 46

    GRIDS ..... 48

*Parameters Involved, Alphabetically* ..... 49

    LABELS ..... 49

*Font Names and Attributes* ..... 50

*Parameters Involved, Alphabetically* ..... 50

    LEGENDS ..... 51

*Parameters Involved, Alphabetically* ..... 51

    NOTES, OR ANNOTATIONS ..... 51

*Parameters Involved, Alphabetically* ..... 52

    REGIONS, OR BOXES ..... 52

*Parameters Involved, Alphabetically* ..... 53

**7. CDL PARAMETER DEFINITIONS ..... 54**

**8. COMMON CDL ATTRIBUTES ..... 312**

**APPENDIX A: DATE AND TIME VALUES ..... 323**

    DATE/TIME DATA INPUT ..... 323

*Mapping Date/Time Information* ..... 323

*Absolute Date Expressions* ..... 323

*Relative Time Units* ..... 324

*Numeric Time Units* ..... 325

    DATE/TIME DATA FORMAT AND TIMEBASE ..... 325

*SIMPLEDATE Format Expression Attributes* ..... 326

*DATE Format Expression Attributes* ..... 326

*TimeBase* ..... 327

*TimeUnit* ..... 328

**GENERAL INFORMATION ..... 336**

# 1. An Overview of Chart Definition Language (CDL)

All of Visual Mining's charting solutions use the Chart Definition Language (CDL) to create and manipulate charts. This common use of CDL makes it easier to recognize and preserve chart definitions when moving from one Visual Mining product to another. CDL is a simple ASCII scripting language that is easy to read and understand. CDL parameters live in a file with a file extension of .cdl or .cdx.

Visual Mining's Charting solutions have many chart rendering details designed into the code. Each chart can be generated using very minimal CDL parameters without concern for details such as tic marks, font characteristics, grid lines, etc. As users become more proficient they can use CDL to create complex charts with a wide range of features and an informative and animated appearance.

## 1.1 CDL Statements

In the most general form, CDL parameter strings have the following form:

```
parameter1 = value1;  
parameter2 = value2;  
...
```

*Note:* parameter strings can occur in any order and quotes can be single or double as long as they are matched.

*Note:* a semi-colon is required at the end of the parameter definition.

The parameter names come from a defined set of CDL names, such as **Background**, **Header**, **DataSet1**, etc. The value consists of one or more *attributes*.

Attributes are single primitive values that can be combined to form a complete value for a CDL parameter.

*Example 1:*

```
Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
```

In the above Example 1, the parameter **Background** has 6 attributes. Those attributes are: *Color*, *BorderType*, *BorderWidth*, *ImageURL*, *ImageFormat* and *BorderColor*. Note that attributes are separated by a comma.

In the below Example 2, we have assigned values to 4 of the attributes. The resulting chart will display with a white background, a black border which is raised using a width of 3.

*Note:* The trailing semicolon is required!

*Example 2:*

```
Background = (white, RAISED, 3, , , black);
```

CDL uses hundreds of attributes to describe colors, borders, width, depth, etc. For example, many parameters will use a color attribute to describe a color to be used in a chart. The below attributes are Color Attributes.

<b>Bgcolor</b>	<b>LineColor</b>	<b>SymColor</b>
<b>Fgcolor</b>	<b>BorderColor</b>	<b>TipColor</b>
<b>Color</b>	<b>FillColor</b>	<b>SliceColor</b>

### Where Attributes Fit In

The term “attributes” is somewhat context dependent, when we are speaking of *CDL attributes*. Generally, we mean the qualities or values assigned to some aspect or attribute of a chart, such as the names of the days of the week given as bar labels, or the string, “Weekday Network Load” given as the title of a chart.

CDL has a rich set of attributes—from 800 to 1000 altogether—which may be used in defining charts. The emphasis is on “may,” because all of the chart types possess defaults that produce a simple graph that is easily modified, and it’s not necessary to specify every little thing in order to get a decent result. The example below shows the CDL required to define a simple bar chart.

#### *Example:*

```
ChartType = Barchart;           # type of chart
ChartWidth = 400;               # width of chart
ChartHeight = 250;              # height of chart
Header = ("Weekday Network Load", black, "Arial", 14); # title of chart
BottomTics = ("ON", gray, "Arial Plain", 12); # use tic marks on bottom
LeftTics = ("ON", gray, "Arial Plain", 12); # use tic marks on left
BarLabels = "Mon", "Tue", "Wed", "Thu", "Fri"; # the tic labels
Bar3DDepth = 0;                 # flat bars
LeftTitle = ("Bytes Per Sec", gray, "Arial Plain", 12, 90); # add left axis title
DataSets = ("Server #1", slateblue); # specify first data set
DataSet1 = 100, 125, 245.78, 147, 67; # static data
```

Produced this chart, shown somewhat reduced in size:

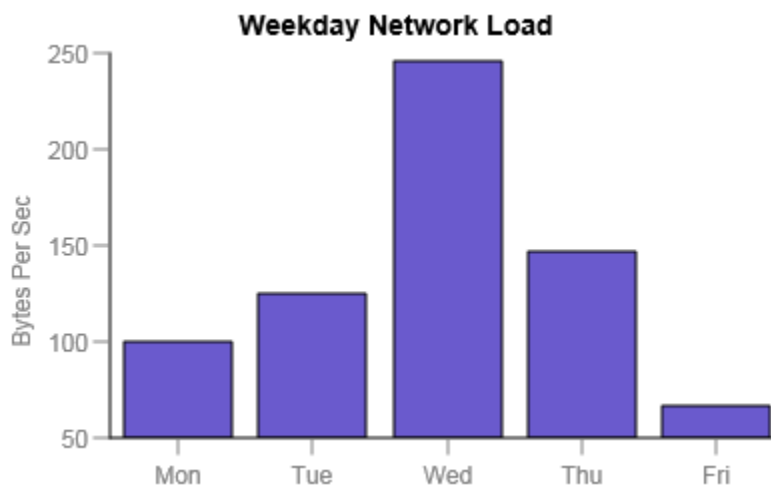


Figure 1: A Very Basic Bar Chart

No actual programming—meaning, constructing logical algorithms in a programming language such as C++ or Java—was required to do this. The defaults construct a reasonable chart, so you don’t have to delve into the depths of CDL to make simple charts. You just need to know what parameters you want to define, and describe them. However, the depths of CDL are there, and very useful indeed for adding interactivity and expressive detail to your charts.

This CDL can also be used in a variety of applications of Visual Mining products. It can be provided as parameters to a NetCharts Pro Applet. It can be loaded into NetCharts Server along with data connections to automatically generate charts from database. It can be used to override automatic styling in NetCharts

## 1. An Overview of Chart Definition Language (CDL)

---

Performance Dashboards KPIs. And this CDL can also be modified and made even more robust by using NetCharts Designer to define the CDL templates without needing to learn the CDL language.

### CDL Statement Types

CDL has 5 basic statement types, differing primarily in the number and type of attributes that make up the parameter value.

#### **Single Value**

A Single parameter value is one attribute.

```
Example 1: GraphType = STACKED;
```

In the above example, the value for parameter named GraphType has one attribute. In this case, the attribute is Type and the value is STACKED.

#### **List Values**

A List parameter value is a list of attribute values.

```
Example 1: DataSet1 = 100,202,340,500;  
Example 2: BarLabels = "Qtr1","Qtr2","Qtr3","Qtr4";
```

#### **Tuple Values**

A tuple can be easily identified because tuples start and end with parentheses. A Tuple parameter value is a group of attribute values.

```
Example 1: Header = ("My Header", black, Arial, 12, 90);
```

#### **Tuple List Values**

A Tuple List parameter value is a list of groups of attribute values.

```
Example 1: StockSet1 = (100,200,300), (102,234,490), (102,234,490);  
Example 2: DataSets = ("cherries",red), ("plums",purple), ("apples",red);
```

#### **VTUPLE Values**

A VTUPLE value is a tuple value that can contain a variable number of attributes.

```
DialTicLabels = (name, label1, label2,..., labelN);
```

The value for DialTicLabels is a VTuple. The first attribute is the name of a dial, the rest of the attributes are the tic labels for that dial. If the chart does not need all the attributes in the VTuple, it ignores them. If the chart needs more attributes than provided, it re-uses the last attribute.

### 1.2 Visual Mining's Suite of Chart Generation Solutions

All of the products are written entirely in Java and use some form of Visual Mining's *Chart Definition Language* (CDL) as a basis for defining and generating charts. Because of this common base rendering engine, all NetCharts solutions are capable of generating the same charts. The products differ primarily in the way in which each is integrated into the software infrastructure of a user's application. The following section describes Visual Mining products and how CDL is referenced in each product.

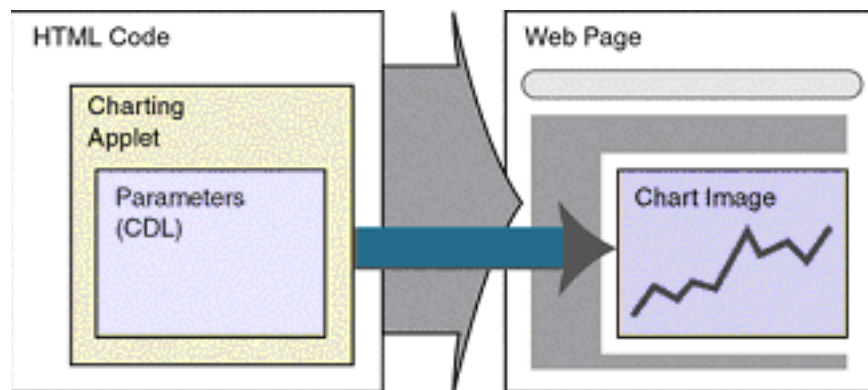


Figure 2. The relationship of CDL to its chart

#### NetCharts Pro

*NetCharts Pro* (<http://www.visualmining.com/nc-pro/>) is a Java programmer-friendly chart generation solution. The NetCharts Pro API allows it to be used in Integrated Development Environments (IDE) such as Eclipse and IBM's WebSphere Studio. NetCharts Pro can create images of charts in standard web formats such as PNG, SVG and JPG, making it ideally suited for server-side use to chart enable applications using EJBs, servlets or JSP pages.

#### NetCharts Designer

*NetCharts Designer* (<http://www.visualmining.com/nc-designer/>) provides a comprehensive desktop Integrated Development Environments (IDE) for creating and managing charts, graphs, tables, interactive dashboards, and scorecards that can then be used in web-based applications. NetCharts Designer streamlines data analytics and development with one simple interactive dashboard solution. NetCharts Designer can be used to create chart templates for use by NetCharts Pro and complete dashboard applications for deployment within NetCharts Server.

## NetCharts Server

*NetCharts Server* (<http://www.visualmining.com/nc-server/>) is a platform that can present complete dashboards or charts that developers define and publish using NetCharts Designer, the solution IDE. Together they are a traditional dashboard development solution allowing users complete control over the content, styling and interactivity included in the dashboards to implement very complex and rich dashboards. The NetCharts Server platform can also be used in conjunction with an entire range of web infrastructures from the simplest CGI scripts, to the most sophisticated Enterprise Application Servers. Its simple HTTP based interface and pre-built toolkits can be used by nearly any server-side web programming language (e.g. .NET, JSP, Java, CFML, PERL and C) to dynamically create chart enabled web pages.

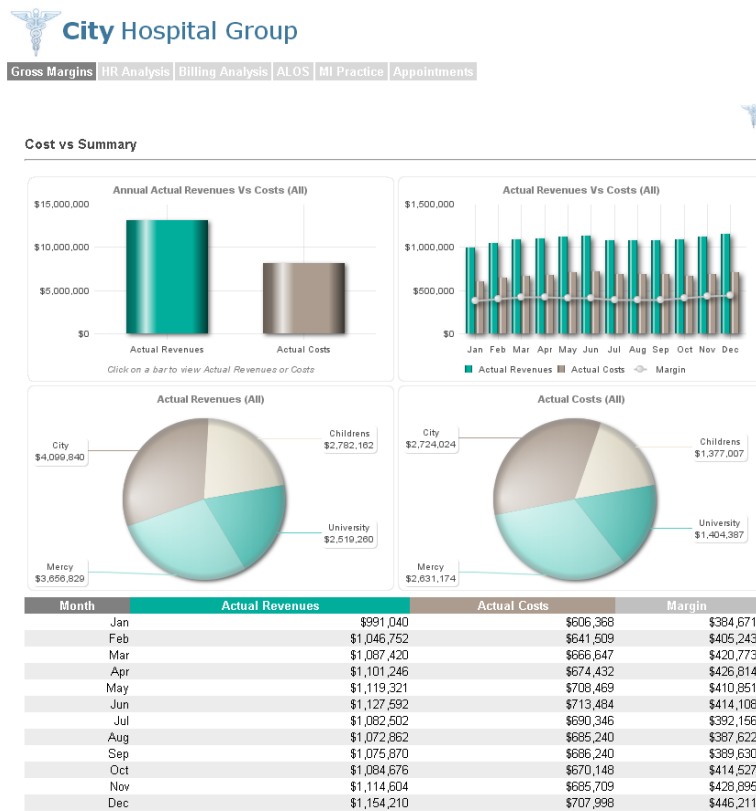


Figure 3. NetCharts Server combines charts and tables into dynamic formats.

## NetCharts Performance Dashboards

*NetCharts Performance Dashboards* (<http://www.visualmining.com/ncpd/>) is a complete end-user enabling, agile, dashboarding solution. NetCharts Performance Dashboards allows users or end-users to view, create and customize dashboards. No coding or programming is required and there are no languages to learn to connect to your data and produce and present dashboards. It's all done automatically based on selections, allowing for rapid, agile dashboard development with rich client-side interactivity.



## 2. NetCharts Pro Applets and CDL

Java applications can be embedded within an HTML document by means of an applet. Applets are Java code intended to be run within the context of an *applet viewer* or by a *web browser* using a Java runtime plugin or control. An applet is placed within an HTML document by using the `<APPLET>` tag. The `<APPLET>` tag is a container for the Applet that allows for the definition and configuration of the applet. Space to display the Applet is reserved by using the `WIDTH` and `HEIGHT` attributes of the `<APPLET>` tag.

Just like regular Java applications, additional parameters can be passed into an applet. The `<PARAM>` tags are child elements of the `<APPLET>` tag and must be contained between the start and end tags of an `<APPLET>` tag. The primary means of configuring the properties of NetCharts is via parameter passing into the applet via `<PARAM>` elements.

### Example:

```
<APPLET CODE="com.applets.simpleapplet" NAME="myapp" WIDTH=100 HEIGHT=100>
  <PARAM NAME="bgcolor" VALUE="black">
  <PARAM NAME="fgcolor" VALUE="yellow">
</APPLET>
```

## Configuring NetCharts Pro Applets

NetCharts Pro Applets are controlled via Visual Mining's Chart Definition Language (CDL). From the tic marks on an axis to the width of the lines in a line chart to the background to the text in the legend—all aspects of the applet chart are managed via CDL.

There is more than one way to arrange parameters to create a chart with NetCharts Pro Applets. The first two methods using individual `<PARAM>` tag or the NetCharts specific `NFParamScript` parameter will suffice for most applet charting with static data. Beyond inclusion of the parameters within the `<APPLET>` tag it is possible to have the chart definition retrieved from an external source or dynamically generated and applied to the applet chart.

**NOTE:** With recent browser updates, it is recommended to use the additional *permissions* parameter and run the applet in *sandbox* mode. Running your applets in *sandbox* mode, it is possible to limit the warning dialogs to a single dialog when the first NetCharts Pro Applet is presented.

## Standard HTML-Style Parameter Usage

The standard method for passing parameters to any Java applet is through the use of one or more `<PARAM>` tags, which are defined within the confines of the `<APPLET>` tag. Each `<PARAM>` tag is assigned a unique name and a value, which may be quoted. For example, the following HTML segment defines a pie chart that has a white background with a black shadow, a red header title, and three pie slices with specific values and labels:

### Example:

```
<applet name='mychart' code='netcharts.apps.NFBarchartApp'
  codebase='/netcharts' archive='ncp-core.jar'
  width=400 height=250>
  <param name='permissions' value='sandbox' />
  <param name='DataSets' value='("BarSet1",null,BAR,4,FILLED)'>
  <param name='DataSet1' value='100,125,245.78,147,167'>
  <param name='BarLabels' value='"Mon","Tue","Wed","Thu","Fri"'>
</applet>
```

Notice that each parameter is defined within a separate `<PARAM>` tag and that the values *can* span multiple lines, provided that the value is a quoted string. Also note how the strings defined within the parameter value use double quotes to differentiate themselves from the entire value string.

In this case, multiple parameters are passed in via the `<param>` tag. This can, however, get tedious, especially when many parameters are being passed in.

### **Using a Parameter Script to Specify CDL Parameters**

NetCharts has a parameter called `NFParamScript`. This parameter can be used to combine the CDL parameters into a single 'script'. While the use of `<PARAM>` tags, as above, is common for a small number of parameters, for the sake of convenience and readability, the `NFParamScript` parameter may be used, where the assigned value can hold any number of CDL statements. For example, the above applet code could be rewritten as:

The `NFParamScript` parameter requires less redundant typing, and is easier to generate from within a CGI script. For example, the above applet code could be rewritten as:

#### **Example:**

```
<applet name='mychart' code='netcharts.apps.NFBarchartApp'  
        codebase='/netcharts' archive='ncp-core.jar'  
        width=400 height=250>  
  <param name='permissions' value='sandbox' />  
  <param name='NFParamScript' value='  
    DataSets=("BarSet1", null, BAR, 4, FILLED);  
    DataSet1=100,125,245.78,147,167;  
    BarLabels="Mon", "Tue", "Wed", "Thu", "Fri";  
  '>  
</applet>
```

Notice how only a single `<PARAM>` tag is used, with a quoted, multi-line value definition, using tabs or spaces for readability. Also note how the single quote is used to delimit the `<PARAM>` tag value and the double quote is used to denote a string value as a parameter attribute.

It is important to note that each parameter definition in a `NFParamScript` is terminated within a semicolon.

If both individual `<PARAM>` tags and an `NFParamScript` are used to define parameters in the same applet, then the individual `<PARAM>` tags will be processed first. That is, the `NFParamScript` values will take precedence over the individual `<PARAM>` tag values, if the same parameter name is being defined.

### **Pointing to Parameters with a URL**

NetCharts Pro Applets has a parameter called `NFParamURL`. This parameter can be used to combine the CDL parameters into a single 'script' accessed via a URL. So instead of placing all of the parameter definitions within an HTML file, you can use a URL access to retrieve the parameter definitions.

Generally, this URL would refer to a data file, but any URL can be used, including a CGI script or application that generates the parameters dynamically. For instance, you may have multiple HTML files that reference the same chart. In that case, maintenance is reduced if the chart definition is stored in a single file or generated dynamically by a single CGI script. In the latter case, the CGI URL could even be customized to generate a custom chart for a given HTML file.

For example, the above applet code could be rewritten as:

#### **Example:**

## 2. NetCharts Pro Applets and CDL

---

```
<applet name='mychart' code='netcharts.apps.NFBarchartApp'  
        codebase='/netcharts' archive='ncp-core.jar'  
        width=400 height=250>  
    <param name='permissions' value='all-permissions' />  
    <param name='NFParamURL' value='barchart.dat`>  
</applet>
```

**NOTE:** With recent browser updates, the additional *permissions* parameter set to *all-permissions* will need to be set to allow for cross-domain access to the remote CDL. Running your applets with *all-permissions* will limit the warning dialogs to a single dialog when the first NetCharts Pro Applet is presented. You must also use the signed JAR file within the **/applets/all-permissions** folder of the NetCharts Pro distribution.

Now, `barchart.dat` can be a static file located within the codebase of the applet, or it could actually be pointing to a file generated dynamically by a server-side script, or it could even be a script itself, generating data on-the-fly. Regardless, the resulting content-type needs to be `text/plain` or `application/x-cdl`. Note, if the URL references a file outside of the codebase folder, the default applet security features in most web browsers will prevent the applet from accessing the file.

The `NFParamURL` parameter is processed after any individual `<PARAM>` tags, but before any `NFParamScript`. This allows the URL to contain standard attributes that may be overridden by the local parameter definitions defined in the `NFParamScript`.

**NOTE:** When a relative URL is given, the URL is interpreted relative to the Document Base of the HTML file containing the applet.

The `NFParamURL` parameter is processed after any individual `<PARAM>` tags, but before any `NFParamScript`. This allows the URL to contain standard attributes that may be overridden by the local parameter definitions defined in the `NFParamScript`.

### ***Putting Parameters in an Include File***

NetCharts Pro Applets has a parameter called *IncludeFile*. This parameter can be used to insert any CDL file into another CDL definition, combining the CDL parameters into a single file to access. The *IncludeFile* parameter has the following syntax:

```
IncludeFile = "urlpathname";
```

#### ***Example:***

```
IncludeFile = "http://www.visualmining.com/demo/background.cdl";  
IncludeFile = "../demo/background.cdl";
```

The filename given can be any URL that is valid for the environment in which the chart is being executed. For example, in a browser, if the chart is downloaded from a Web server, then the URL can specify a relative pathname or a full HTTP pathname.

An example of the *IncludeFile* parameter's use is a company trying to enforce a standard **Background** style for all of its charts. You could use an include file to accomplish this. Replace the **Background** definitions in the CDL with an *IncludeFile* reference to the CDL file containing the company standard Background definitions.

#### ***Example:***

**File: Background.cdl**

```
Background = (mintcream, BOX, 1, , TILE, sandybrown);
```

**File: MyChart.html**

```
<applet name='mychart' code='netcharts.apps.NFBarchartApp'  
        codebase='/netcharts' archive='ncp-core.jar'  
        width=400 height=250>  
<param name='permissions' value='all-permissions' />  
<param name='NFParamScript' value='  
        IncludeFile = "Background.cdl";  
        DataSet1 = ...  
        ...  
>  
</applet>
```

Note that if a relative pathname is specified, as in the example above, then the `DocumentBase` will be used as the start of the relative location.

### Using a Parameter Server

NetCharts Pro Applets has a parameter called *NFParamServer* that is similar to *NFParamURL*. In the same way that you specify a URL from which parameter definitions are read, you may specify an arbitrary TCP server from which definitions will be processed. With a parameter server however, the definitions can be processed throughout the applet's lifetime and not just at initial load. That is, a parameter server can continuously update any or all of the chart parameters, providing for dynamic charting.

The value of the *NFParamServer* parameter has the following format:

```
<param name='NFParamServer' value='hostname:port/arguments' />
```

If *NFParamServer* is defined, it will be processed after all other parameter definitions have been processed, including those sent into the applet via individual `<PARAM>` tags, a *NFParamURL* parameter or a *NFParamScript* parameter. This allows the initial parameters to contain standard attributes that may be overridden by the parameter server. At that time, a connection is made to the given host and port, which is assumed to be a TCP server capable of generating parameter statements. The TCP server can be written using any language or utility desired. It need only generate a stream of text data that is equivalent to a *NFParamScript*.

Everything following the "/" in the parameter string will be passed to the parameter server upon connection, terminated with a new line character. This allows the server to determine the specific data required for this connection.

For example, the following HTML segment specifies a parameter server located at `www.netcharts.com` using port 2000. An initial line containing `DataSet=Monday, User=Fred` is sent to the parameter server on startup, telling it which chart to generate.

**Example:**

```
<applet name='mychart' code='netcharts.apps.NFPiechartApp'  
        codebase='/netcharts' archive='ncp-core.jar'  
        width=400 height=400>  
<param name='permissions' value='all-permissions' />  
<param name='NFParamServer'  
        value="www.netcharts.com:2000/DataSet=Monday, User=Fred">  
</applet>
```

**NOTE:** With recent browser updates, the additional *permissions* parameter set to *all-permissions* will need to be set to allow for cross-domain access to the remote CDL. Running your applets with *all-*

*permissions* will limit the warning dialogs to a single dialog when the first NetCharts Pro Applet is presented. You must also use the signed JAR file within the `/applets/all-permissions` folder of the NetCharts Pro distribution.

### A Technical Note about Server Connection Processing

A background thread processes all parameter statements generated by the `NFParamServer` while the chart is being displayed. An *Update* command can be sent at any time within the data stream to cause the chart display to be updated. That is, parameter definitions received from the parameter server are batched together and the chart is refreshed whenever an *Update* command is received.

While server connection commands are being processed a status message is displayed whenever a parsing error occurs in the input stream. This aids developers in determining when a server bug exists. After displaying the status message, the parser will flush the input stream to the next semicolon. and attempt to continue processing.

A status message is also displayed if the server connection is broken prematurely, notifying the user of the broken connection. To properly close down a connection without displaying such a message, the Parameter Server should send the following command in the input stream:

```
Close;
```

**NOTE:** The trailing semicolon is required!

Parameter servers may display arbitrary messages to the user at any time by using the `STATUS` command in the input stream, as follows:

```
STATUS "This is a status message";
```

When the `STATUS` command is processed, the message window will be displayed immediately.

### *Dynamic Updates from another Applet*

Parameter definitions can be programmatically updated at any time throughout the life of a chart applet through the use of the `loadParams()` method. The `loadParams()` method accepts a string value, which consists of one or more parameter statements. As with the `NFParamServer`, the `loadParams()` definitions are batched together and the chart is refreshed whenever an *Update* command is given.

In the following Java example, the current applet accesses a NetCharts Pro pie chart applet via the `AppletContext()` (a standard Java capability) and then executes the `loadParams()` method of the pie chart applet to update the slice values and labels, as well as the pie chart background color.

#### **Example:**

```
AppletContext ac = getAppletContext();
NFPiechartApp pie = (NFPiechartApp) ac.getApplet("piechart");

if (pie == null) {
    System.out.println ("Unable to access piechart");
} else {
    pie.loadParams ("Background = (blue);"
        + "Slices = "
        + "(12,, 'Fred'),"
        + "(23,, 'Sally'),"
        + "(15,, 'Jim');"
        + "Update;"
    );
}
```

### Loading Data Parameters

Java programmers can define parameter data using raw values, instead of `String` or `StringBuffer` expressions, in order to gain some additional performance (for large data sets) or to streamline data management within the application or applet (small or large data sets.)

For example, the following code can be used to define a bar chart data set:

**Example:**

```
Vector data = new Vector();

data.addElement (new Integer(27));
data.addElement (new Float(45.3));
data.addElement ("34");

bar.set ("BarSet1", data);
```

This eliminates the need to convert data vectors to comma separated strings in order to pass them to the `loadParams()` method.

**NOTE:** The `Vector` items can be defined using different object types, depending on the attribute type. In this case, the **BarSet1** parameter expects a list of numbers, which can be defined using many different object types, including a `String`. The parser will automatically convert, if possible, items as needed when loading the data.

The specific object type used for the data depends on the type of the parameter definition. In the example above, the **BarSet1** parameter accepts a vector of numbers, so the object type used is a `Vector`. For parameters that accept a single value, a raw value type is passed.

### Dynamic Updates from JavaScript

The `loadParamsJS()` method of each NetCharts Pro Applet can be used to update chart parameters programmatically from JavaScript. The applet context is determined using the `document` object within JavaScript, as shown in the following example:

**Example:**

```
var app = document.piechart;

app.loadParamsJS ("Background = (blue);");
app.loadParamsJS ("Slices = (12, 'Fred'), (23, 'Sally'), (15, 'Jim');");
app.loadParamsJS ("Update;");
```

See *Web Scripting with NetCharts Pro Applets* for more details on NetCharts Pro Applets.

## 3. Anatomy of a Chart

While every element of charts produced by NetCharts can be customized and controlled, with countless combinations allowing you to build a chart exactly the way you want, the chart itself consists of many parts that are shared and used across the different chart types. The stacked bar chart represents a rectangular chart and shares many features with other rectangular charts like line charts, combo (bar and line), x-y/quadrant charts, etc. Other chart types like pie charts, dial charts, etc. also share some features but have components unique to their design.

Let's review the anatomy of a chart in a bit more detail by using a stacked bar chart as an example. To assist identifying the different parts that make up a chart, the chart example below includes gray labels with blue text pointing at each item. Below the chart image you'll find a description of what each part does.

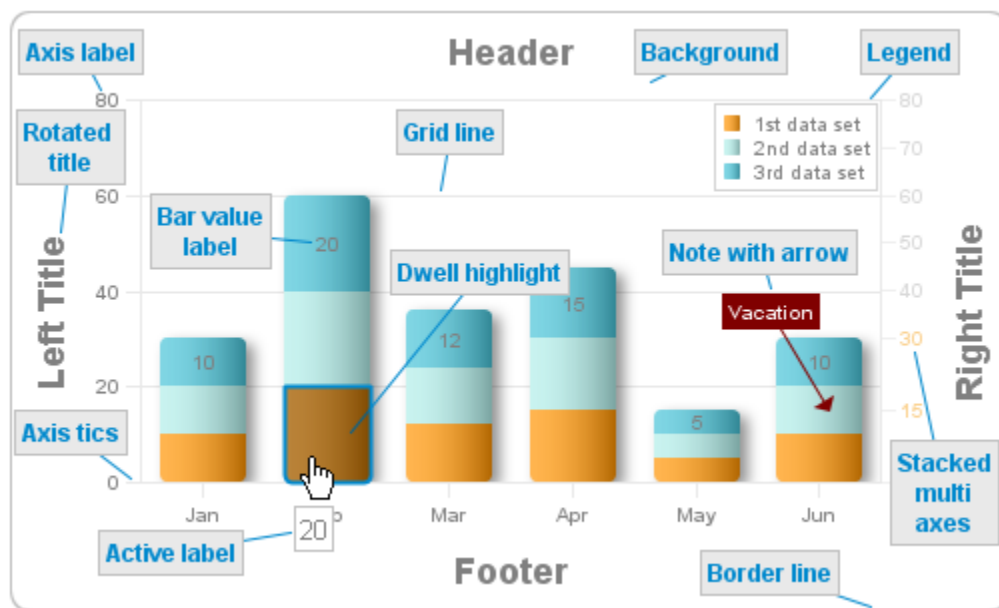


Figure 5: Chart anatomy, using a stacked bar chart as an example.

*Note that the axis labels and axis ticks, can appear on the top, bottom, left, or right sides of a chart (though we have omitted them at the top axis for the sake of clarity in the figure above) and they may also be combined. So you could label and show tick marks on all four axes, with multiple grids and backgrounds, if you so desired.*

**Legend** – While the Legend can be placed anywhere in the background area, it can also be placed inside of the chart area as demonstrated in this example where it can be positioned through further precision using X and Y coordinates.

### 3. Anatomy of a Chart

---

**Dwell highlight** – This bar and pie specific feature can be customized to use a particular color for its background and border when a pointer is placed over the area. In this example a transparent gray and a blue border line was used for the dwell highlight.

**Note (with an arrow)** – This feature allows you to place a text note anywhere on the chart with options that include mapping to pixel, percent or axis (Bottom, Top, Left, Right). In this example the *Vacation* note also uses a line with an arrow style pointer.

**Bar value label** – This feature allows you to place a label (External, Top, Middle, Bottom) on a bar to displays its value. In a stacked bar, you can also choose to display the value label on a specific dataset stack as demonstrated in this example.

**Grid line** – This feature allows you to specify a grid type and set the lines to be horizontal, vertical or both. You can also customize the grid further to use different colors, line thickness, and also set the background color.

**Background** – This feature allows you to set the background to be a different color than what's used in the center of the chart area. You can also customize further to use a background image, a single color or also combine two color gradients.

**Border line** – This feature allows you to customize the line thickness, color, of the border and also setting the corner type to be either square or rounded. This example demonstrates using a rounded gray border line.

**Axis label** – This feature allows the labels to display a value or a text label. The font type, size, angle and color can also be customized.

**Axis ticks** – This feature specifies the lines extending out from the axis line. They can be customized to be a certain color or length as well.

**Rotated title** – The titles (left, right, header and footer) are placed outside of the chart area where the font type, size, angle and color can also be customized independently.

**Stacked multi axes** – This feature allows using multiple axes where each axes uses its own specific scale, font type, size, angle and color. This example demonstrates using a multiple right axis where the lower right axis uses an orange font color and a different scale than the upper right axis.

You can customize our charts further through countless combinations by using what we call Chart Definition Language or CDL, which is a collection of plain text parameters that describe a chart. These CDL parameters can be edited manually or by using our design studio NetCharts Designer which enables you to create and manage charts, graphs, tables, and interactive dashboards.



## 4. Types of Axes

### **Standard Axes**

An axis can have many functions. An axis provides a reference for measuring coordinates. An axis also provides a way for displaying tic marks and scales. NetCharts allows for the zero base line of an axis to be located at the Top, Bottom, Left and Right of the chart. Moreover, you can have a series of axes and a series of scale sets on one line.

### **Multi-Scaled Single Axes**

CDL parameters for axes allow a single axis to have multiple scales. For example, the chart displayed below has a bottom axis with two scales. The number of scales an axis can have is theoretically unlimited although it will be practically bounded by the size of the chart.

#### **CDL Example:**

```
BottomTics = ("ON",black,"SansSerif",10,90,null);
LeftTics = ("ON",null,"SansSerif",10,0,null);
Footer = ("[Sundays shown in gray]",black,"SansSerif",10,0);
BottomFormat = (DATE,"%n %d","1/1/2001 12:01:00","1d");
# BottomTicLocations is an example of variable axis labeling.
# Tics are placed at the explicit locations specified.
BottomTicLocation = "1/1/2001 12:01:00","1/8/2001
12:01:00","1/15/2001 12:01:00","1/22/2001 12:01:00","1/29/2001
12:01:00","2/1/2001 12:01:00","2/2/2001 12:01:00","2/3/2001
12:01:00","2/4/2001 12:01:00","2/5/2001 12:01:00","2/6/2001 12:01:00";

BottomScaleSet = (-.5,31,7,80),(32,36.5,1,20);
```

### **Multiple Axes in a Single Direction**

This feature allows multiple axes to be defined in the same physical space that is usually occupied by a single axis. In other words, one can define multiple left, top, bottom or right axes. Each axis can be independently defined and controlled. Up to 10 axes can be defined in each direction. The parameters that define each additional axis are the same as the standard axis parameters with the exception of a number at the end.

#### **CDL example:**

```
LeftFormat = (FLOAT,"$%dK",,);
LeftFormat2 = (FLOAT,"$%dK",,);
LeftFormat3 = (FLOAT,"$%dK",,);
```

The parameters LeftFormat and LeftFormat1 are considered the same for backward compatibility.

### **Variable Axis Labeling**

This feature allows additional control over axis tic mark drawing and labeling. The drawing of tic marks on each axis can be specifically enabled or disabled.

There are 8 CDL parameters for variable tic labeling, one for each axis location.

TopTicLocations	Top Axis Tic Mark Locations
BottomTicLocations	Bottom Axis Tic Mark Locations
LeftTicLocations	Left Axis Tic Mark Locations
RightTicLocations	Right Axis Tic Mark Locations
TopTicLength	Top Axis Tic Mark Length
BottomTicLength	Bottom Axis Tic Mark Length
LeftTicLength	Left Axis Tic Mark Length
RightTicLength	Right Axis Tic Mark Length

## Logarithmic Axes

NetCharts (starting with version 4.0) supports logarithmic x and y axes.

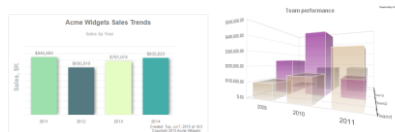
```
ChartName = "Logarithmic Y Scale";
ChartType = XYCHART; ChartWidth = 400;
ChartHeight = 300;
Background = (white,NONE,0,null,TILE,black);
Header = ("Base 16 Logarithmic Y-Scale",black,"SansSerif",14,0);
HeaderBox = (null,NONE,1,null,TILE,black);
BottomTics = (ON,black,"Courier New",10,null);
BottomScale = ("1200","2600",);
LeftTics = (ON,black,"Courier New",10,0);
LeftScale = ("300","6000","1000");
LeftScaleMode = (LOG,16);
```

Using LOG requires that the scale minimum be non-zero;

## 5. CDL Parameters Arranged by Chart Type

Use this index as a way of identifying parameter definitions to use when you have a specific chart to assemble. Chart types are arranged alphabetically.

### Bar Chart and 3D Bar Chart



#### Generally Supported Parameter Types

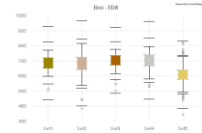
ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	AxisThickness = 15;
Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
Note Sets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

#### Specifically Supported Parameters

BarAnimationStyle	BarAnimationStyle = GROW   FADE   NONE;
Bar3DDepth	Bar3DDepth = Number;
BarActiveLabels	BarActiveLabels = ("Label", "URL", "Target"), ...;
BarBorder	BarBorder = (LineStyle, LineWidth, Color);
BarColorTable[n]	BarColorTable[1-50] = color1,color2...;
BarColorTable[n]P[m]	BarColorTable[1-50]P[1-50] = color1,color2...;
BarCorners	BarCorners = (topleft, topright, bottomright, bottomleft);
BarDropShadow	BarDropShadow = (color, offsetx, offset, size);
BarFillPattern	BarFillPattern = (type, Color1, Color2, imageURL), ...;
BarFillPattern[n]P[m]	BarFillPattern [1-50]P[1-50] = (type, Color1, Color2, imageURL), ...; (for <i>STACKEDGROUPED BarChart</i> only)

BarHighlights	Barhighlights = (type,start,stop,top,right,bottom,left,width,height,topLeft,topRight,bottomRight,bottomLeft), ...;
BarRightFillPattern	BarRightFillPattern = (type, color1, color2, imageURL), ...;
BarRightFillPattern[n]P[m]	BarRightFillPattern[1-50]P[1-50] = (type, color1, color2, imageURL), ...; (for <i>STACKEDGROUPED BarChart</i> only)
BarSpotlights	BarSpotlights = (start,stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;
BarSymbol	BarSymbol = (BarSymbolType, BarColor);
BarTopFillPattern	BarTopFillPattern = (type, Color1, Color2, imageURL), ...;
BarTopFillPattern[n]P[m]	BarTopFillPattern[1-50]P[1-50] = (type, color1, color2, imageURL), ...; (for <i>STACKEDGROUPED BarChart</i> only)
BarValueLabel	BarValueLabel = (mode, color, font name, width);
BarValueLabelBox	BarValueLabelBox = (color, mode, depth);
BarValueLabelStyle	BarValueLabelStyle = labelposition1, labelposition2, ...;
BarWidth	BarWidth = Percent1,Percent2,...;
DataAxis	DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
DataSets	DataSets = (Label1, Color1, Type1), (Label2, Color2, Type2), ...;
DataSet[n]	DataSet[1-50] = a, b, c, ...;
DataSet[n]P[m]	DataSet[1-50]P[1-50] = a, b, c, ...;
DataLegend	DataLegend = ON OFF;
DataLegendGrid	DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);
DataLegendGridLine	DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);
GraphType	GraphType = Type;
GraphLayout	GraphLayout = Type;
GroupStackLabels	GroupStackLabels = "label1","label2",...;
GroupStackSegment Labels	GroupStackSegmentLabels = "label1","label2",...;
PlotArea	PlotArea = (xlocation, ylocation, width, height);
ShowGroupStackLabels	ShowGroupStackLabels = ON OFF;
StackDisplayOrder	StackDisplayOrder = mode;
StackedBar Connectors	StackedBarConnectors = OFF   LINE   FILL;
StackLabel	StackLabel = Type;
ViewPoint	ViewPoint = (CARTESIAN SPHERICAL, a,b,c); (for <i>3D BarChart</i> only)
ZAxisLabels	ZAxisLabels = ((ON OFF, Color, FontName, FontSize, Angle, interiorAlignment); (for <i>3D BarChart</i> only)

## Box Chart



### Generally Supported Parameter Types

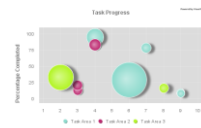
ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
ActiveLabels[1-50]	<code>ActiveLabels = ("Label1", "URL1", "Target1"), ...;</code>
Axis	See Chapter 6, <b>Axis Modifications</b> for various parameters available
Axis Thickness	<code>AxisThickness = 15;</code>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
BuildAnimationEnabled	<code>BuildAnimationEnabled = ON OFF;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

BoxActiveLabels	<code>BoxActiveLabels = ("Label1", "URL1", "Target1"), ...;</code>
BoxFence	<code>BoxFence = ON   OFF;</code>
BoxFillPattern	<code>BoxFillPattern = (type, color1, color2, imageURL), ...;</code>
BoxHeight	<code>BoxHeight = Height;</code>
BoxLabels	<code>BoxLabels = "Label1", "Label2", ...;</code>
BoxLimitLines	<code>BoxLimitLines = (limit1-1, limit1-2, ... limit1-N), ... (limitM-1, limitM-2, ... limitM-N);</code>
BoxLimitLineStyle	<code>BoxLimitLineStyle = (type1, width1, color1), ... (typeN, widthN, colorN);</code>
BoxSymbolWidth	<code>BoxSymbolWidth = Percent;</code>
BoxWidth	<code>BarWidth = Percent1, Percent2, ...;</code>
DataAxis	<code>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>

DataPointActiveLabels[n]	DataPointActiveLabels(n) = ("Label1", "URL1", "Target1"), ...;
DataPointColor	DataPointColor = Color;
DataPointJitter	DataPointJitter = ON   OFF;
DataPointSymbol	DataPointSymbol = (type1, size1, style1, bordercolor1, borderwidth1, imagel, color1), ...;
DataType	DataType = Type;
DataSets	DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;
DataSet[n]	DataSet[1-50] = a, b, c, ...;
FenceActiveLabels	FenceActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;
FencePosition	FencePosition = OVER   UNDER;
GraphLayout	GraphLayout = VERTICAL   HORIZONTAL;
MeanActiveLabels	MeanActiveLabels = ("Label1", "URL1", "Target1"), ...;
MeanColor	MeanColor = Color;
MeanLine	MeanLine = (type, width, color);
MeanSymbol	MeanSymbol = (type1, size1, style1, bordercolor1, borderwidth1, imagel, color1), ...;
MedianColor	MedianColor = Color;
NaturalDisplayOrder	NaturalDisplayOrder = ON   OFF;
MinimumDataPoints	MinimumDataPoints = int_val;
OutlierActiveLabels[1-50]	OutlierActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;
OutlierColor	OutlierColor = Color;
OutlierSymbol	OutlierSymbol = (type1, size1, style1, bordercolor1, borderwidth1, imagel, color1), ...;
PercentileN	PercentileN = N;
PlotArea	PlotArea = (xlocation, ylocation, width, height);
PlotType	PlotType = STANDARD   EDA   GAUSSIAN   TENNINETY;
RelativeBoxSymbolWidth	RelativeBoxSymbolWidth = OFF   LINEAR   SQRT ;
ShowDataPoints	ShowDataPoints = ON   OFF;
WhiskerType	WhiskerType = BOX   LINE;

## Bubble Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6,

	<i>Axis Modifications</i> for various parameters available
Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

AddDataPoint	AddDataPoint = ("Name", X, Y, Z, "Label", "URL", "Target"), ...;
BubbleAxis	BubbleAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
BubbleColorTable[n]	BubbleColorTable[1-50] = color1,color2...;
BubbleFillPattern	BubbleFillPattern = (type, color1, color2, imageURL), ...;
BubbleScale	BubbleScale = (MinValue, MaxValue, AREA DIAMETER, PointColor), ...;
BubbleSets	BubbleSets = ("Name1", Color1), ("Name2", Color2), ...;
BubbleSets[n]	BubbleSetn = (x,y,z), (x,y,z), ...;
BubbleSymbol	BubbleSymbol = (SymType, MaxSize, Style, BorderColor, BorderWidth, SymbolColor,ShadowWidth), ...;
BubbleSymbolAnimationStyle	BubbleSymbolAnimationStyle = SCALE   FADE   NONE;
LineStyle	LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;
LineDropShadow	LineDropShadow = (color, offsetx, offsety, size);
PlotArea	PlotArea = (xlocation, ylocation, width, height);

## Combo Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
--------------------	--

Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	<code>AxisThickness = 15;</code>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
BuildAnimationEnabled	<code>BuildAnimationEnabled = ON OFF;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
DataLegend	<code>DataLegend = ON OFF;</code>
DataLegendGrid	<code>DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);</code>
DataLegendGridLine	<code>DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);</code>
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes</i> , or <i>Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions</i> , or <i>Boxes</i> for various parameters available.

### Specifically Supported Parameters

BarFillPattern	<code>BarFillPattern = (type, color1, color2, imageURL), ...;</code>
Bar3DDepth	<code>Bar3DDepth = Number;</code>
BarSymbol	<code>BarSymbol = (BarSymbolType, BarColor);</code>
BarValueLabel	<code>BarValueLabel = (mode, color, font name, width);</code>
BarValueLabelBox	<code>BarValueLabelBox = (color, mode, depth);</code>
BarValueLabelStyle	<code>BarValueLabelStyle = labelposition1, labelposition2, ...;</code>
BarActiveLabels	<code>BarActiveLabels = ("Label", "URL", "Target"), ...;</code>
BarBorder	<code>BarBorder = (LineType, LineWidth, Color);</code>
BarColorTable[n]	<code>BarColorTable[1-50] = color1,color2...;</code>
BarWidth	<code>BarWidth = Percent1,Percent2,...;</code>
DataAxis	<code>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
DataSets	<code>DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;</code>
DataSet[n]	<code>DataSet[1-50] = a, b, c, ...;</code>
DrawOrder	<code>DrawOrder = Symbol;</code>
GraphType	<code>GraphType = Type;</code>
GraphLayout	<code>GraphLayout = Type;</code>



PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>
StackLabel	<code>StackLabel = Type;</code>
Line3DDepth	<code>Line3DDepth = depth;</code>
LineAxis	<code>LineAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
LineColorTable[n]	<code>LineColorTable[1-50] = color1,color2...;</code>
LineDropShadow	<code>LineDropShadow = (color, offsetx, offsety, size);</code>
LineFillPattern	<code>LineFillPattern = (type, color1, color2, imageURL), ...;</code>
LineLabels[n]	<code>LineLabels[1-50] = ("Label", "URL", "Target"), ...;</code>
LineSets	<code>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</code>
LineSet[n]	<code>LineSet[1-50] = y1, y2, y3,...;</code>
LineStyle	<code>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor,ShadowWidth), ...;</code>
LineSymbolSpotlights	<code>LineSymbolSpotlights = (start,stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;</code>
LineValueLabel	<code>LineValueLabel = (mode, color, font name, width);</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
LineWidth	<code>LineWidth = PercentDepth;</code>
StackDisplayOrder	<code>StackDisplayOrder = mode;</code>
StackedBarConnectors	<code>StackedBarConnectors = OFF   LINE   FILL;</code>
StackLabel	<code>StackLabel = Type;</code>

## Diagram/Map Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>

Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

---

AppendPolyDataToActiveLabels	AppendPolyDataToActiveLabels = ON OFF;
Edges	Edges = (NodeStart, NodeEnd, Color, Direction, LineStyle, LineWidth, ArrowStyle, ArrowLength, ArrowWidth), ...;
NodeBox	NodeBox = (Color, BorderType, BorderWidth, ImageURL, ImageFormat, BorderColor);
NodeDrag	NodeDrag = ON OFF;
NodeLabel	NodeLabel = (Color, FontName, FontSize, Angle), ...;
Nodes	Nodes = (Name, Label, X, Y), ...;
PolyActiveLabels	PolyActiveLabels = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
PolyColor	PolyColor = (tagName, color), ...;
PolySet	PolySet = (tagName, x1,y1,x2,y2,...), ...;

## Dial Chart



### Generally Supported Parameter Types

---

Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

---

Dials	Dials = (Name, StartAngle, StopAngle, RadiusPercentage, NONE INSIDE OUTSIDE), ...;
-------	--

DialActiveLabels	DialActiveLabels = (Name, Label, URL, Target), ...;
DialBorders	DialBorders = (Name, Type, Thickness, Color, NONE CENTER ENDTOEND), ...;
DialClip	DialClip = clipType;
DialClipPad	DialClipPad = N;
DialDelete	DialDelete = (Name ALL), ...;
DialFills	DialFills = (Name, Color, NONE CENTER ENDTOEND), ...;
DialFillPattern	DialFillPattern = (type, color1, color2, imageURL), ...;
DialFormats	DialFormats = (Name, FLOAT INTEGER DECIMAL, formatExpression), ...;
DialHandAnimationStyle	DialHandAnimationStyle = GROW   FADE   NONE
DialOuterBorder	DialOuterBorder = (color1, color2, width);
DialOuterFillPattern	DialOuterFillPattern = (type, color1, color2, imageURL);
DialScale	DialScale = (Name, MinValue, MaxValue, StepValue), ...;
DialSize	DialSize = (minWidth, minHeight, maxWidth, maxHeight);
DialSectorAnimationStyle	DialSectorAnimationStyle = GROW   FADE   NONE
DialSquare	DialSquare = Switch;
DialTics	DialTics = (Name, Color, LineWidth, PercentofRadius), ...;
DialTicLabels	DialTicLabels = (Name, Label1, Label2, ..., LabelN), ...;
DialTicLabelStyles	DialTicLabelStyles = (Name, ON OFF, LabelPos, Color, FontName, FontSize, Angle, interiorAlignment), ...;
Hands	Hands = (Name, TipColor, ShaftColor, DialName, HandLabel), ...;
HandActiveLabels	HandActiveLabels = (Name, Label, URL, Target), ...;
HandBorders	HandBorders = (Name, lineType, LineWidth, Color);
HandButtonBorder	HandButtonBorder = ( lineType, LineWidth, Color);
HandButtonEdgeHighlights	HandButtonEdgeHighlights = (start,stop,gap,size), ...;
HandData	HandData = (Name, Value, Length), ...;
HandDelete	HandDelete = (Name ALL), ...;
HandDrag	HandDrag = "ON" "OFF";
HandDropShadow	HandDropShadow = (color, offsetx, offsety, size);
HandLabels	HandLabels = ("Name", "ON OFF", labelPos, color, font, size, angle, justification), ...;
HandStyles	HandStyles = (Name, NEEDLELINE NEEDLEFILL NEEDLEBUTTON  SHARP ROUND BLOCK LINE NONE, TipWidth, ShaftWidth), ...;
Sectors	Sectors = (Name, Color, DialName, OuterRadius, InnerRadius, SectorLabel), ...;
SectorActiveLabels	SectorActiveLabels = (Name, Label, URL, Target), ...;
SectorBorders	SectorBorders = (Name, Type, LineWidth, Color), ...;
SectorColors	SectorColors = (Name,color),...;

SectorData	<code>SectorData = (Name, StartValue, StopValue), ...;</code>
SectorDelete	<code>SectorDelete = (Name ALL), ...;</code>
SectorDrag	<code>SectorDrag = "ON" "OFF";</code>
SectorEdgeHighlights	<code>SectorEdgeHighlights = (start,stop,gap,size), ...;</code>
SectorFillPattern	<code>SectorFillPattern = (type, color1, color2, imageURL), ...;</code>
SectorLabels	<code>SectorLabels = (Name, ON OFF, LabelPos, Color, FontName, FontSize, Angle, interiorAlignment), ...;</code>



## Heat Map Chart

### Generally Supported Parameter Types

ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

CellTextAutoColorThreshold	<code>CellTextAutoColorThreshold = 0-100 %;</code>
GridBlockActiveLabels	<code>GridBlockActiveLabels = (Name, Label, URL, Target), ...;</code>
GridBlockBackgroundColor	<code>GridBlockBackgroundColor = color;</code>
GridBlockCellColorType	<code>GridBlockCellColorType = type;</code>
GridBlockColors	<code>GridBlockColors = (color1, color2,...colorN);</code>
GridBlockColors	<code>GridBlockColors = (color1, color2,...colorN);</code>
GridBlockColorSpectrum	<code>GridBlockColorSpectrum = (color1,color2,min,max,gradientstep);</code>
GridBlockExpressions	<code>GridBlockExpressions = ("operator",value1,value2,color),...;</code>
GridBlockBottomLabel	<code>GridBlockBottomLabel=("mode", color, "font name", font size, angle, interiorAlignment);</code>
GridBlockLeftLabel	<code>GridBlockLeftLabel=("mode", color, "font name", font size, angle, interiorAlignment);</code>
GridBlockRightLabel	<code>GridBlockRightLabel=("mode", color, "font name", font size, angle, interiorAlignment);</code>
GridBlockTopLabel	<code>GridBlockTopLabel=("mode", color, "font name", font size, angle, interiorAlignment);</code>

GridBlockBottomLabels	GridBlockBottomLabels= <i>L1,L2,L3 ...</i> ;
GridBlockLeftLabels	GridBlockLeftLabels= <i>L1,L2,L3 ...</i> ;
GridBlockRightLabels	GridBlockRightLabels= <i>L1,L2,L3 ...</i> ;
GridBlockTopLabels	GridBlockTopLabels= <i>L1,L2,L3 ...</i> ;
GridBlockLayout	GridBlockLayout=( <i>Height,Width</i> );
GridBlockLine	GridBlockLine=( <i>"Style", width, color</i> );
GridBlockSort	GridBlockSort=( <i>SORT TYPE,SORT ORDER</i> );
GridBlockLabel	GridBlockLabel= ( <i>"mode",color,"Font",size</i> );
GridBlockLabels	GridBlockLabels= <i>L1,L2,L3 ...</i> ;
GridBlockValues	GridBlockValues= <i>N1,N2,N3 ...</i> ;
GridBlockValueFormat	GridBlockValueFormat( <i>Format Type, FormatExpr</i> );
GridBlockValueStyle	GridBlockValueStyle=( <i>"mode",color,"Font",size</i> );

## Histogram Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ( <i>"Label1", "URL1", "Target1"</i> ), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	AxisThickness = <i>15</i> ;
Background	Background = ( <i>Color, BorderType, BorderWidth,</i> <i>"ImageURL", ImageFormat, BorderColor</i> );
BackgroundFillPattern	BackgroundFillPattern = ( <i>type, color1, color2,</i> <i>imageURL</i> ), ...;
BuildAnimationEnabled	BuildAnimationEnabled = <i>ON OFF</i> ;
ChartElementSpacing	ChartElementSpacing = <i>spacing</i> ;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = ( <i>LineColor1, bgColor1, borderColor1,</i> <i>bgImage1, ImageFormat1</i> ), ...;
Legend	Legend = ( <i>"Label", Color, "FontName", FontSize,</i> <i>Angle, interiorAlignment</i> );
NoteSets	NoteSets = ( <i>"Name1", Justify1</i> ), ( <i>"Name2", Justify2</i> ), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

Bar3DDepth	Bar3DDepth = <i>Number</i> ;
BarActiveLabels	BarActiveLabels = (" <i>Label</i> ", " <i>URL</i> ", " <i>Target</i> "), ...;
BarBorder	BarBorder = ( <i>LineType</i> , <i>LineWidth</i> , <i>Color</i> );
BarColorTable[n]	BarColorTable[1-50] = <i>color1,color2...</i> ;
BarColorTable[n]P[m]	BarColorTable[1-50]P[1-50] = <i>color1,color2...</i> ;
BarFillPattern	BarFillPattern = ( <i>type</i> , <i>Color1</i> , <i>Color2</i> , <i>imageURL</i> ), ...;
BarFillPattern [1-50]P[1-50]	BarFillPattern [1-50]P[1-50] = ( <i>type</i> , <i>Color1</i> , <i>Color2</i> , <i>imageURL</i> ), ...; (for STACKEDGROUPED BAR)
BarValueLabel	BarValueLabel = ( <i>mode</i> , <i>color</i> , <i>font name</i> , <i>width</i> );
BarValueLabelBox	BarValueLabelBox = ( <i>color</i> , <i>mode</i> , <i>depth</i> );
BarValueLabelStyle	BarValueLabelStyle = <i>labelposition1</i> , <i>labelposition2</i> , ...;
BarWidth	BarWidth = <i>Percent1,Percent2,...</i> ;
DataAxis	DataAxis = ( <i>XAxis1</i> , <i>YAxis1</i> ), ( <i>XAxis2</i> , <i>YAxis2</i> ), ...;
DataSets	DataSets = ( <i>Label1</i> , <i>Color1</i> , <i>Type1</i> ), ( <i>Label2</i> , <i>Color2</i> , <i>Type2</i> ), ...;
DataSet[n]	DataSet[1-50] = <i>a</i> , <i>b</i> , <i>c</i> , ...;
DataSet[n]P[m]	DataSet[1-50]P[1-50] = <i>a</i> , <i>b</i> , <i>c</i> , ...;
GraphType	GraphType = <i>Type</i> ;
GraphLayout	GraphLayout = <i>Type</i> ;
GroupStackLabels	GroupStackLabels = " <i>label1</i> ", " <i>label2</i> ", ...;
GroupStackSegment Labels	GroupStackSegmentLabels = " <i>label1</i> ", " <i>label2</i> ", ...;
HistogramBin	HistogramBin = ( <i>HistogramBinType</i> , <i>HistogramBinSize</i> );
HistogramScale	HistogramScale = ( <i>histogramMinValue</i> , <i>histogramMaxValue</i> );
HistogramType	HistogramType = <i>BYNUMER PERCENTAGE PROBABLILTY</i> ;
PlotArea	PlotArea = ( <i>xlocation</i> , <i>ylocation</i> , <i>width</i> , <i>height</i> );
ShowGroupStackLabels	ShowGroupStackLabels = <i>ON OFF</i> ;
StackDisplayOrder	StackDisplayOrder = <i>mode</i> ;
StackLabel	StackLabel = <i>Type</i> ;

## Line Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = (" <i>Label1</i> ", " <i>URL1</i> ", " <i>Target1</i> "), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available

---

Axis Thickness	<code>AxisThickness = 15;</code>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
BuildAnimationEnabled	<code>BuildAnimationEnabled = ON OFF;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

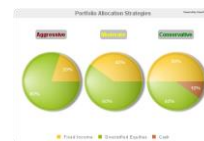
### Specifically Supported Parameters

---

ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
DataLegend	<code>DataLegend = ON OFF;</code>
DataLegendGrid	<code>DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);</code>
DataLegendGridLine	<code>DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);</code>
DwellOffset	<code>DwellOffset = size;</code>
GraphType	<code>GraphType = Type;</code>
Line3DDepth	<code>Line3DDepth = depth;</code>
LineAxis	<code>LineAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
LineFillPattern	<code>LineFillPattern = (type, color1, color2, imageURL), ...;</code>
LineSets	<code>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</code>
LineSet[1-50]	<code>LineSet[1-50] = y1, y2, y3, ...;</code>
LineStyle	<code>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ... ;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth), ...;</code>
LineSymbolSpotlights	<code>LineSymbolSpotlights = (start, stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
LineWidth	<code>LineWidth = PercentDepth;</code>
PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>

StackLabel                      StackLabel = *Type*;

## MultiPie Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = <i>spacing</i> ;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

BestFit	BestFit = ON OFF
LabelPos	LabelPos = <i>Float</i> ;
Pie3DDepth	Pie3DDepth = <i>Pixels</i> ;
PieAngles	PieAngles = Value1, Value2, Value3, Value4, ..., ValueN;
PieDropShadow	PieDropShadow = (color, offsetx, offsety, size);
PieEdgeHighlights	PieEdgeHighlights = (start, stop, gap, size), ...;
PieSquare	PieSquare = ON/OFF;
PieBackgrounds	PieBackgrounds = (Region Tuple) , ...;
PieLabel	PieLabel = (State, Color, FontName, FontSize, Angle, InteriorAlignment) , ...;
PieLabelBox	PieLabelBox = = (Region Tuple) , ...;
PieLayout	PieLayout = (Orientation, Row, Columns);
PieLabelLocation	PieLabelLocation = <i>Location</i> ;
PieMargin	PieMargin = <i>Integer</i> ;
PieLabels	PieLabels = Label1, Label2, Label3, ... LabelN;
SliceBorder	SliceBorder = (LineStyle, Width, Color);
SliceSets	SliceSets = ("Name", Color, "State");



SliceSet[N]	<code>SliceSet = Value1, Value2, Value3, Value4, ..., ValueN;</code>
SliceFillPattern	<code>SliceFillPattern = (type, color1, color2, imageURL), ...;</code>
SliceFormat	<code>SliceFormat = (FormatType, "FormatExpr");</code>
SliceLabel	<code>SliceLabel = (State, Color, FontName, FontSize, Angle, interiorAlignment);</code>
SliceLabelBox	<code>SliceLabelBox = (Color, BorderType, BorderWidth);</code>
SliceLabelContent	<code>SliceLabelContent = Content;</code>
SliceLabelContentDelimiter	<code>SliceLabelContentDelimiter = "delimiter";</code>
SliceLabelLine	<code>SliceLabelLine = (LineStyle, LineWidth, Color);</code>
SliceLabels	<code>SliceLabels = Label1, Label2, Label3, Label4, ..., LabelN;</code>
SliceLabelStyle	<code>SliceLabelStyle = (Style);</code>
SlicePos[N]	<code>SlicePos[N] = PiePosition1, PiePosition2, ...;</code>

## Pareto Chart



### Generally Supported Parameter Types

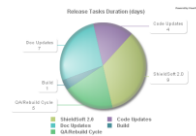
ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	<code>AxisThickness = 15;</code>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
BuildAnimationEnabled	<code>BuildAnimationEnabled = ON OFF;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

## Specifically Supported Parameters

---

BarFillPattern	BarFillPattern = (type, color1, color2, imageURL), ...;
Bar3DDepth	Bar3DDepth = Number;
BarValueLabel	BarValueLabel = (mode, color, font name, width);
BarValueLabelBox	BarValueLabelBox = (color, mode, depth);
BarValueLabelStyle	BarValueLabelStyle = labelposition1, labelposition2, ...;
BarActiveLabels	BarActiveLabels = ("Label", "URL", "Target"), ...;
BarBorder	BarBorder = (LineStyle, LineWidth, Color);
BarColorTable[n]	BarColorTable[1-50] = color1,color2...;
BarWidth	BarWidth = Percent1,Percent2,...;
CumulativeLineSetName	CumulativeLineSetName = "name";
CumulativeLineStyle	CumulativeLineStyle = (LineStyle, LineWidth, Color, FillColor);
CumulativeLineSymbol	CumulativeLineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor,ShadowWidth);
CumulativeLineValueLabel	CumulativeLineValueLabel = (mode, color, font name, width);
Cumulative LineValueLabelBox	CumulativeLineValueLabelBox = (color, mode, depth);
CumulativeLineValueLabelStyle	CumulativeLineValueLabelStyle = labelposition;
DataAxis	DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
DataSets	DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;
DataSet[n]	DataSet[1-50] = a, b, c, ...;
DrawOrder	DrawOrder = Symbol;
EightyLineSetName	EightyLineSetName = "name";
EightyTwentyLineStyle	EightyTwentyLineStyle = (LineStyle, LineWidth, Color, FillColor);
EightyTwentyLineSymbol	EightyTwentyLineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor,ShadowWidth);
Line3DDepth	Line3Ddepth = depth;
LineSymbolSpotlights	LineSymbolSpotlights = (start,stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;
PlotArea	PlotArea = (xlocation, ylocation, width, height);
ShowEightTwentyLines	ShowEightyTwentyLines = ON OFF;
StackDisplayOrder	StackDisplayOrder = mode;
StackLabel	StackLabel = Type;
TwentyLineSetName	TwentyLineSetName = "name";

## Pie Chart



### Generally Supported Parameter Types

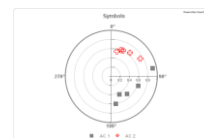
ActiveLabels	<code>ActiveLabels = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
BuildAnimationEnabled	<code>BuildAnimationEnabled = ON OFF;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

LabelPos	<code>LabelPos = Float;</code>
Pie3DDepth	<code>Pie3DDepth = Pixels;</code>
PieAngle	<code>PieAngle = Integer;</code>
PieDropShadow	<code>PieDropShadow = (color, offsetX, offsetY, size);</code>
PieSize	<code>PieSize = (minWidth, minHeight, maxWidth, maxHeight);</code>
PieSquare	<code>PieSquare = Switch;</code>
SliceBorder	<code>SliceBorder = (LineStyle, Width, Color);</code>
SliceColor	<code>SliceColor = Color1, Color2, Color3 ..., ColorN;</code>
SliceData	<code>SliceData = Value1, Value2, Value3, Value4, ..., ValueN;</code>
SliceFillPattern	<code>SliceFillPattern = (type, color1, color2, imageURL), ...;</code>
SliceFormat	<code>SliceFormat = (FormatType, "FormatExpr");</code>
SliceLabel	<code>SliceLabel = (State, Color, FontName, FontSize, Angle, interiorAlignment);</code>
SliceLabelBox	<code>SliceLabelBox = (Color, BorderType, BorderWidth);</code>
SliceLabelContent	<code>SliceLabelContent = Content;</code>

SliceLabelContentDelimiter	<code>SliceLabelContentDelimiter = "delimiter";</code>
SliceLabelLine	<code>SliceLabelLine = (LineStyle, LineWidth, Color);</code>
SliceLabels	<code>SliceLabels = Label1, Label2, Label3, Label4, ...; Labeln;</code>
SliceLabelStyle	<code>SliceLabelStyle = (Style);</code>
SlicePos	<code>SlicePos = Position1, Position2, Positions3, ...;</code>
Slices	<code>Slices = (Value, SliceColor, Label, LabelColor, FontName, FontSize, LabelAngle, LabelBgColor, LabelBgBorder), ...;</code>

## Polar Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

LineSets	<code>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</code>
LineSet[n]	<code>LineSet[1-50] = (x1,y1), (x2,y2), (x3,y3), ...;</code>
LineStyle	<code>LineStyle = (LineStyle, LineWidth, Color, FillColor, LineStyle, FillType), ...;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth), ...;</code>

LineValueLabel	<code>LineValueLabel = (mode, color, font name, width);</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>
PolarLabel	<code>PolarLabel = (mode, color, font name, width);</code>
PolarLabelFormat	<code>PolarLabelFormat = (dataType, formatString);</code>
PolarLabelStep	<code>PolarLabelStep = stepsize;</code>
PolarScale	<code>PolarScale = (min.max, step);</code>
PolarSquare	<code>PolarSquare = ON OFF;</code>
PolarSize	<code>PolarSize = (minWidth, minHeight, maxWidth, maxHeight);</code>
RadialAxesAngles	<code>RadialAxesangles = angle1, angle2, ...;</code>
RadialAxesColors	<code>RadialAxesColors = color1, color2, ...;</code>
RadialAxesFormat	<code>RadialAxesFormat = (dataType, formatString), ...;</code>
RadialAxesTics	<code>RadialAxesTics = ("axisTicLabelMode", axisTicLabelColor, "axisTicLabelFont", axisTicLabelFontSize, axisTicLabelFontAngle, interiorAlignment), ...;</code>
RadialGrids	<code>RadialGrids = (gridRadius, gridLineType, gridLineWidth, gridLineColor, gridAreaColor), ...;</code>

## Radar Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>

Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

---

CenterRadius	<code>CenterRadius = radius;</code>
LineFillPattern	<code>LineFillPattern = (type, color1, color2, imageURL), ...;</code>
LineSets	<code>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</code>
LineSet[n]	<code>LineSet[1-50] = y1, y2, y3, ...;</code>
LineStyle	<code>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, imageURL, SymbolColor, ShadowWidth), ...;</code>
LineValueLabel	<code>LineValueLabel = (mode, color, font name, width);</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>
RadarSize	<code>RadarSize = (minWidth, minHeight, maxWidth, maxHeight);</code>
RadarSquare	<code>RadarSquare = mode;</code>
RadialAxes	<code>RadialAxes = ("axisTitle",minValue,maxValue,stepSize),...;</code>
RadialAxesColors	<code>RadialAxesColors = color1,color2,...;</code>
RadialAxesFormat	<code>RadialAxesFormat = (dataType,formatString), ...;</code>
RadialAxesLabel	<code>RadialAxesLabel = ("axisLabelMode",axisLabelColor,"axisLabelFont",axisLabelFontSize,axisLabelFontAngle,axisLabelInteriorAlignment), ...;</code>
RadialAxesScales	<code>RadialAxesScales = (min1,max1,step1), (min2,max2,step2),...;</code>
RadialAxesTics	<code>RadialAxesTics = ("axisTicLabelMode",axisTicLabelColor,"axisTicLabelFont",axisTicLabelFontSize,axisTicLabelFontAngle,interiorAlignment), ...;</code>
RadialAxesTitles	<code>RadialAxesTitles = "title1","title2",...;</code>
RadialGrids	<code>RadialGrids = (gridRadius,gridLineType,gridLineWidth,gridLineColor,gridAreaColor), ...;</code>

## Stock Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	AxisThickness = 15;
Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

Bar3Ddepth	Bar3DDepth = Number;
BarActiveLabels	BarActiveLabels = ("Label", "URL", "Target"), ...;
BarBorder	BarBorder = (LineType, LineWidth, Color);
BarFillPattern	BarFillPattern = (type, color1, color2, imageURL), ...;
BarColorTable[n]	BarColorTable[1-50] = color1,color2...;
BarWidth	BarWidth = Percent1,Percent2,...;
DataAxis	DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
DataSets	DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;
DataSet[n]	DataSet[1-50] = a, b, c, ...;
DrawOrder	DrawOrder = Symbol, ...;
GraphType	GraphType = Type;
GraphLayout	GraphLayout = Type;

GraphType	<code>GraphType = Type;</code>
Line3Ddepth	<code>Line3DDepth = depth;</code>
LineAxis	<code>LineAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
LineColorTable[n]	<code>LineColorTable[1-50] = color1,color2...;</code>
LineFillPattern	<code>LineFillPattern = (type, color1, color2, imageURL), ...;</code>
LineLabels[1-50]	<code>LineLabels[1-50] = ("Label", "URL", "Target"), ...;</code>
LineSets	<code>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</code>
LineSet[n]	<code>LineSet[1-50] = y1, y2, y3, ...;</code>
LineStyle	<code>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL,SymbolColor,ShadowWidth), ...;</code>
LineValueLabel	<code>LineValueLabel = (mode, color, font name, width);</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
LineWidth	<code>LineWidth = PercentDepth;</code>
PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>
StackDisplayOrder	<code>StackDisplayOrder = mode;</code>
StackLabel	<code>StackLabel = Type;</code>
StockAxis	<code>StockAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
StockData[1-50]	<code>StockData[1-50] = (High1, Low1, Open1, Close1), (High21, Low2, Open2, Close2), ...;</code>
StockColorTable[n]	<code>StockColorTable[1-50] = color1,color2...;</code>
StockFillPattern	<code>StockFillPattern = (type, color1, color2, imageURL), ...;</code>
StockLabels[1-50]	<code>StockLabels[1-50] = ("Label1", "URL1", "Target1"), ("Label2", "URL2", "Target2"), ...;</code>
StockSets	<code>StockSets = (Label1, Color1, Width1, TicLen1), (Label2, Color2, Width2, TicLen2), ...;</code>
StockWidth	<code>StockWidth = (Width, TicLen);</code>

## Strip Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	<code>ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;</code> See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available



Axis Thickness	<code>AxisThickness = 15;</code>
Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
BackgroundFillPattern	<code>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</code>
ChartElementSpacing	<code>ChartElementSpacing = spacing;</code>
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
NoteSets	<code>NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;</code> See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

AppendDataSet[n]	<code>AppendDataSet[1-50] = y1, y2, y3, ...;</code>
BottomLabels	<code>BottomLabels = "Label1", "Label2", ...;</code>
BottomScale	<code>BottomScale = (MinValue, MaxValue, StepValue);</code>
BottomScroll	<code>BottomScroll = (ScrollMin, ScrollMax);</code>
DataAxis	<code>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code>
DataSets	<code>DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;</code>
DataSet[n]	<code>DataSet[1-50] = a, b, c, ...;</code>
LineStyle	<code>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</code>
LineSymbol	<code>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth), ...;</code>
LineSymbolSpotlights	<code>LineSymbolSpotlights = (start, stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;</code>
LineValueLabelBox	<code>LineValueLabelBox = (color, mode, depth);</code>
LineValueLabelStyle	<code>LineValueLabelStyle = labelposition1, labelposition2, ...;</code>
PlotArea	<code>PlotArea = (xlocation, ylocation, width, height);</code>
StripLayout	<code>StripLayout = (NumSlots, InitialFill, MaxFill, UndefinedString);</code>
TopLabels	<code>TopLabels = "Label1", "Label2", ...;</code>
TopScale	<code>TopScale = (MinValue, MaxValue, StepValue);</code>
TopScroll	<code>TopScroll = (ScrollMin, ScrollMax);</code>
Update	<code>Update;</code>

## Time Chart



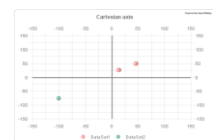
### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

DataAxis	DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
DataSets	DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;
DataSet[n]	DataSet[1-50] = a, b, c, ...;
PlotArea	PlotArea = (xlocation, ylocation, width, height);
TaskColorTable[n]	TaskColorTable[1-50] = color1,color2...;
TaskHeight	TaskHeight = value;
UniqueTaskColors	UniqueTaskColors = ON   OFF;

## X-Y Chart



### Generally Supported Parameter Types

ActiveLabels[1-50]	ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...; See also Chapter 6, <i>Active Labels and Drilldown</i>
Axis	See Chapter 6, <i>Axis Modifications</i> for various parameters available
Axis Thickness	AxisThickness = 15;
Background	Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BackgroundFillPattern	BackgroundFillPattern = (type, color1, color2, imageURL), ...;
BuildAnimationEnabled	BuildAnimationEnabled = ON OFF;
ChartElementSpacing	ChartElementSpacing = spacing;
ColorTable	See Chapter 8, <i>Color</i> for application of color attributes
Grid	Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
Legend	Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; See also Chapter 6, <i>Notes, or Annotations</i>
Title	See Chapter 6, <i>Labels</i> for various parameters available.
TitleBox	See Chapter 6, <i>Regions, or Boxes</i> for various parameters available.

### Specifically Supported Parameters

AddDataPoint	AddDataPoint = ("Name", X, Y, Z, "Label", "URL", "Target"), ...;
DwellOffset	DwellOffset = size;
Line3DDepth	Line3DDepth = depth;
LineAxis	LineAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
LineColorTable[n]	LineColorTable[1-50] = color1,color2...;
LineFillPattern	LineFillPattern = (type, color1, color2, imageURL), ...;
LineSets	LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;
LineSet[n]	LineSet[1-50] = y1, y2, y3,...;
LineStyle	LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;
LineSymbolSpotlights	LineSymbolSpotlights = (start,stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;
LineValueLabelBox	LineValueLabelBox = (color, mode, depth);
LineValueLabelStyle	LineValueLabelStyle = labelposition1, labelposition2, ...;
LineWidth	LineWidth = PercentDepth;
PlotArea	PlotArea = (xlocation, ylocation, width, height);

## 6. CDL Parameters Arranged by Function

Once you have mastered the anatomy of a chart, you may wish to reference CDL parameters by their function. This directory serves as such a reference, with more direct commentary on particular functionality that you will find elsewhere in this Guide.

### ***Active Labels and Drilldown***

All Visual Mining charts, except in the case where charts generated from NetCharts Server or NetCharts Pro as requested as static images only, support the display of informational labels, called dwell or active labels, which are seen whenever the mouse dwells over a specified data value or label. For example, on a Bar chart, the value of an individual bar can be displayed when the mouse cursor hovers over a bar for a short period of time. Alternatively, a user prompt can be displayed when the mouse hovers over a legend item.

If a viewer clicks the mouse while an active label is displayed, the active label may serve to navigate to a predefined URL. This URL itself can serve any of three purposes:

- To replace the current HTML document with any other HTML document
- To alter the display of any named frame or window within the browser.
- To load new chart parameters from a parameter file.

This capability is often called *drilldown*, and is extremely flexible, in that any chart may thus serve as a graphical interface to additional information, be it visual or textual, that is accessible from another document or CGI script.

If the `DwellLabel` parameter is also defined, then a label will automatically be displayed whenever the mouse cursor dwells over a given data value. The `ActiveLabels` parameter defined for a dwell label parameter then specifies the *format* for each label, and not the text value.

### **Parameters Involved, Alphabetically**

<code>ActiveClicks</code>	<code>ActiveClicks = Number;</code>
<code>ActiveLabels</code>	<code>ActiveLabels = ("Label1", "URL1", "Target1"), ...;</code>
<code>ActiveLabels[1-50]</code>	<code>ActiveLabels[1-50] = ("Label", "URL", "Target"), ...;</code>
<code>BackgroundActiveLabel</code>	<code>BackgroundActiveLabel = ("Label", "URL", "Target");</code>
<code>BarActiveLabels</code>	<code>BarActiveLabels = ("Label", "URL", "Target"), ...;</code>
<code>BottomActiveLabels</code>	<code>BottomActiveLabels = ("Label", "URL", "Target");</code>
<code>BottomAxisTitleActiveLabel</code>	<code>BottomAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
<code>BoxActiveLabels</code>	<code>BoxActiveLabels = ("Label", "URL", "Target"),...;</code>
<code>DataPointActiveLabels[1-50]</code>	<code>DataPointActiveLabels[1-50] = ("Label", "URL", "Target");</code>
<code>DwellAnimationHighlightBorderStyle</code>	<code>DwellAnimationHighlightBorderStyle = (lineType, LineWidth, lineColor);</code>
<code>DwellAnimatonHighlightFill</code>	<code>DwellAnimationHighlightFill = (color);</code>
<code>DwellAnimationStyle</code>	<code>DwellAnimationStyle = HIGHLIGHT   NONE;</code>

DwellLabel	<code>DwellLabel = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
FenceActiveLabels[1-50]	<code>FenceActiveLabels[1-50] = ("Label", "URL", "Target");</code>
FooterActiveLabel	<code>FooterActiveLabel = ("Label", "URL", "Target");</code>
GridBlockActiveLabels	<code>GridBlockActiveLabels = ("Label1", "URL1", "Target1"), ...;</code>
HandActiveLabels	<code>HandActiveLabels = (Name, Label, URL, Target), ...;</code>
HeaderActiveLabel	<code>HeaderActiveLabel = ("Label", "URL", "Target");</code>
LeftActiveLabels	<code>LeftActiveLabels = ("Label", "URL", "Target");</code>
LeftTitleActiveLabel	<code>LeftTitleActiveLabel = ("Label", "URL", "Target");</code>
LeftAxisTitleActiveLabel	<code>LeftAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
LegendActiveLabels	<code>LegendActiveLabels = ("Label", "URL", "Target"), ...;</code>
LineLabels[n1-50]	<code>LineLabels[1-50] = ("Label", "URL", "Target"), ...;</code>
MeanActiveLabels	<code>MeanActiveLabels = ("Label", "URL", "Target");</code>
NoteActiveLabels[1-20]	<code>NoteActiveLabels[1-20] = ("Label", "URL", "Target"), ...;</code>
OutlierActiveLabels[1-50]	<code>RightActiveLabels[1-50] = ("Label", "URL", "Target");</code>
PolyActiveLabels	<code>PolyActiveLabels = ("Label", "URL", "Target");</code>
RightActiveLabel	<code>RightActiveLabels = ("Label", "URL", "Target");</code>
RightAxisTitleActiveLabel	<code>RightAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
RightTitleActiveLabel	<code>RightTitleActiveLabel = ("Label", "URL", "Target");</code>
SectorActiveLabels[1-50]	<code>SectorActiveLabels[1-50] = (Name, Label, URL, Target), ...;</code>
SliceActiveLabels[1-50]	<code>SliceActiveLabels[1-50] = (Name, Label, URL, Target), ...;</code>
StockLabels[n]	<code>StockLabels[1-50] = ("Label", "URL", "Target"), ...;</code>
TopActiveLabels	<code>TopActiveLabels = ("Label", "URL", "Target");</code>
TopAxisTitleActiveLabel	<code>TopAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
TopTitleActiveLabel	<code>TopTitleActiveLabel = ("Label", "URL", "Target");</code>

## Axis Modifications

Most Visual Mining charts support the definition and display of one or more axes, depending on the specific chart type. Currently, the following standard axes are defined for most charts, for top, bottom, left, and right axes.

Generally, the bottom and left axes are used by default to map the X and Y data values, although in the Time chart, for example, the top axis is the default X axis.

**Parameters Involved, Alphabetically**

AutoscalePad	AutoscalePad = pad;
AxisThickness	AxisThickness = thickness;
BottomActiveLabels	BottomActiveLabels = ("Label", "URL", "Target");
BottomAxesGaps	BottomAxesGaps = value, value, ...;
BottomAxesLayout	BottomAxesLayout = value, value, ...;
BottomAxisTitle	BottomAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);
BottomAxisTitleActiveLabel	BottomAxisTitleActiveLabel = ("Label", "URL", "Target");
BottomAxisTitleBox	BottomAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BottomColor	BottomColor = Color;
BottomDrawMinorTics	BottomDrawMinorTics = ON OFF;
BottomFormat	BottomFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
BottomLabels	BottomLabels = "Label1", "Label2", ...;
BottomMajorTicColor	BottomMajorTicColor = Color;
BottomMinorTicColor	BottomMinorTicColor = Color;
BottomMargins	BottomMargins = (LeftSideMargin, RightSideMargin);
BottomScale	BottomScale = (MinValue, MaxValue, StepValue);
BottomScaleMode	BottomScaleMode = (mode, logBase), (mode, logBase), ...;
BottomScaleSet	BottomScaleSet = (MinValue, MaxValue, StepValue, Percentage);
BottomScroll	BottomScroll = (ScrollMin, ScrollMax);
BottomTics	BottomTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);
BottomTicLayout	BottomTicLayout = (Mode, SkipCount, StaggerLevels);
BottomTicLocations	BottomTicLocations = value, value, value, ...;
BottomTitle	BottomTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);
BottomTitleActiveLabel	BottomTitleActiveLabel = ("Label", "URL", "Target");
LeftActiveLabels	LeftActiveLabels = ("Label", "URL", "Target");
LeftAxesGaps	LeftAxesGaps = value, value, ...;
LeftAxesLayout	LeftAxesLayout = value, value, ...;
LeftAxisTitle	LeftAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);
LeftAxisTitleActiveLabel	LeftAxisTitleActiveLabel = ("Label", "URL", "Target");
LeftAxisTitleBox	LeftAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);

---

LeftColor	<code>LeftColor = Color;</code>
LeftDrawMinorTics	<code>LeftDrawMinorTics = ON   OFF;</code>
LeftFormat	<code>LeftFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");</code>
LeftLabels	<code>LeftLabels = "Label1", "Label2", ...;</code>
LeftMajorTicColor	<code>LeftMajorTicColor = Color;</code>
LeftMinorTicColor	<code>LeftMinorTicColor = Color;</code>
LeftMargins	<code>LeftMargins = (BottomSideMargin, TopSideMargin);</code>
LeftScale	<code>LeftScale = (MinValue, MaxValue, StepValue);</code>
LeftScaleMode	<code>LeftScaleMode = (mode,logBase), (mode,logBase),...;</code>
LeftScaleSet	<code>LeftScaleSet = (MinValue, MaxValue, StepValue, Percentage);</code>
LeftScroll	<code>LeftScroll = (ScrollMin, ScrollMax);</code>
LeftTics	<code>LeftTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
LeftTicLayout	<code>LeftTicLayout = (Mode, SkipCount, StaggerLevels);</code>
LeftTicLocations	<code>LeftTicLocations = value, value, value, ...;</code>
LeftTitle	<code>LeftTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
LeftTitleActiveLabel	<code>LeftTitleActiveLabel = ("Label", "URL", "Target");</code>
RightActiveLabels	<code>RightActiveLabels = ("Label", "URL", "Target");</code>
RightAxesGaps	<code>RightAxesGaps = value, value, ...;</code>
RightAxesLayout	<code>RightAxesLayout = value, value, ...;</code>
RightAxisTitle	<code>RightAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, , exteriorAlignment);</code>
RightAxisTitleActiveLabel	<code>RightAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
RightAxisTitleBox	<code>RightAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
RightColor	<code>RightColor = Color;</code>
RightDrawMinorTics	<code>RightDrawMinorTics = ON   OFF;</code>
RightFormat	<code>RightFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");</code>
RightLabels	<code>RightLabels = "Label1", "Label2", ...;</code>
RightMajorTicColor	<code>RightMajorTicColor = Color;</code>
RightMinorTicColor	<code>RightMinorTicColor = Color;</code>
RightMargins	<code>RightMargins = (TopSideMargin, BottomSideMargin);</code>
RightScale	<code>RightScale = (MinValue, MaxValue, StepValue);</code>
RightScaleMode	<code>RightScaleMode = (mode,logBase), (mode,logBase),...;</code>
RightScaleSet	<code>RightScaleSet = (MinValue, MaxValue, StepValue, Percentage);</code>
RightScroll	<code>RightScroll = (ScrollMin, ScrollMax);</code>

RightTics	<code>RightTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
RightTicLayout	<code>RightTicLayout = (Mode, SkipCount, StaggerLevels);</code>
RightTicLocations	<code>RightTicLocations = value, value, value, ...;</code>
RightTitle	<code>RightTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
RightTitleActiveLabel	<code>RightTitleActiveLabel = ("Label", "URL", "Target");</code>
TopActiveLabels	<code>TopActiveLabels = ("Label", "URL", "Target");</code>
TopAxesGaps	<code>TopAxesGaps = value, value, ...;</code>
TopAxesLayout	<code>TopAxesLayout = value, value, ...;</code>
TopAxisTitle	<code>TopAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
TopAxisTitleActiveLabel	<code>TopAxisTitleActiveLabel = ("Label", "URL", "Target");</code>
TopAxisTitleBox	<code>TopAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
TopColor	<code>TopColor = Color;</code>
TopDrawMinorTics	<code>TopDrawMinorTics = ON   OFF;</code>
TopFormat	<code>TopFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");</code>
TopLabels	<code>TopLabels = "Label1", "Label2", ...;</code>
TopMajorTicColor	<code>TopMajorTicColor = Color;</code>
TopMinorTicColor	<code>TopMinorTicColor = Color;</code>
TopMargins	<code>TopMargins = (LeftSideMargin, RightSideMargin);</code>
TopScale	<code>TopScale = (MinValue, MaxValue, StepValue);</code>
TopScaleMode	<code>TopScaleMode = (mode, logBase), (mode, logBase), ...;</code>
TopScaleSet	<code>TopScaleSet = (MinValue, MaxValue, StepValue, Percentage);</code>
TopScroll	<code>TopScroll = (ScrollMin, ScrollMax);</code>
TopTics	<code>TopTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);</code>
TopTicLayout	<code>TopTicLayout = (Mode, SkipCount, StaggerLevels);</code>
TopTicLocations	<code>TopTicLocations = value, value, value, ...;</code>
TopTitle	<code>TopTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
TopTitleActiveLabel	<code>TopTitleActiveLabel = ("Label", "URL", "Target");</code>

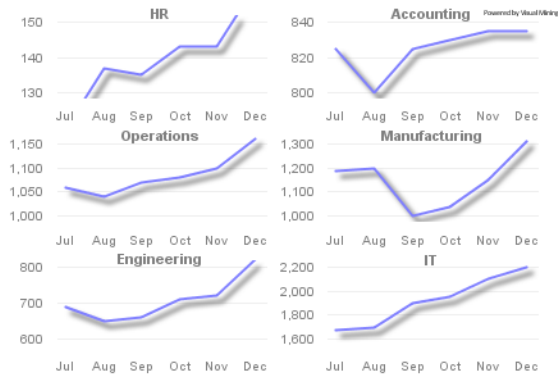
## Grids

All Visual Mining rectangular charts (e.g. bar, line, combo, pareto, etc.) support the display of one or more optional grids behind the data. The grid, if used, forms a background behind the chart, making measurement against the axes easier when reading it.

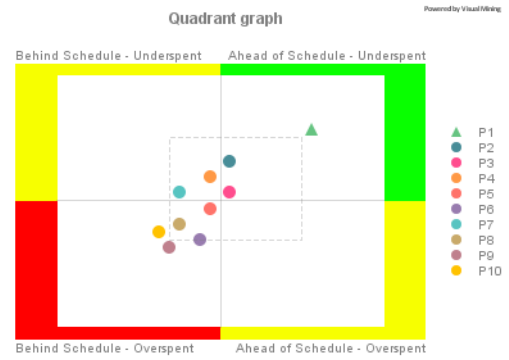


With Visual Mining charts, you can specify several grid characteristics, including different colored and styled grid lines, grid background, and grid depth. Additionally, you can define more than one grid per chart, which is useful when your chart requires independent Y axes.

In additions to lines and bars, you can also use images in the grid. The two charts below indicate some of the variety one can achieve using the grid parameters.



LineChart featuring multiple datasets and grids mapped to multiple axis.



XYChart with multiple datasets and axis, with a grid using color fills.

### Parameters Involved, Alphabetically

Grid is the fundamental parameter for setting up grids, and you must use it first when setting up other grid parameters.

Grid	<code>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</code>
Grid3DDepth	<code>Grid3DDepth = depth;</code>
GridAxis	<code>GridAxis = (XAxis1, YAxis1), ...;</code>
GridLine	<code>GridLine = (LineType, LineStyle, LineWidth), ... ;</code>

### Labels

Label definitions are used extensively throughout Visual Mining charts in order to display titles, legends, axis tics, data labels, etc. In most cases, labels use a standard vector, or “tuple,” to define the text and its appearance. Thus, you are able to control:

- The text itself, or in the case of axis tic labels, only whether the label is shown or not
- Color
- Font name and attributes
- Font size
- Text angle

## Font Names and Attributes

The font names available to NetCharts depend on the Operating System and the Java Virtual Machine being used to run the software. For example, the fonts available to NetCharts running on Solaris will be different than the fonts available running on Windows.

Font names can be augmented with additional font style information. Adding "Plain", "Bold" or "Italic" to the font name modifies the style as specified. For example "Courier Bold Italic" is a valid font name specification. By default NetCharts chooses a BOLD style for the specified font so "TimesRoman" is equivalent to "TimesRoman Bold". To get a standard version of a font, add "Plain" to the name.

Font names can also be augmented with "underline", "overline", "linethrough", "ascent=N", "descent=N", "leading=N" and "maxLineAdvance=N".

Any combination of style modifiers is allowed. For example:

```
TimesRoman Plain Italic underline overline linethrough ascent=5 descent=0 leading=-5
```

is a valid font specification which uses a non-bold italic type, draws a line above, below and through the text, and controls the amount of space allocated for character ascents, descents and the space between consecutive lines.

*maxLineAdvance* controls the maximum length of a string before NetCharts breaks to a new line.

## Parameters Involved, Alphabetically

The following parameters follow the label standard format, allowing you to customize your text.

BottomAxisTitle	<code>BottomAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
BottomTicks	<code>BottomTicks = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);</code>
BottomTitle	<code>BottomTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
DwellLabel	<code>DwellLabel = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
Footer	<code>Footer = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
Header	<code>Header = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment, extend);</code>
GridBlockLabel	<code>GridBlockLabel = ("mode", color, "Font name", Font size, Angle, interiorAlignment)</code>
LeftAxisTitle	<code>LeftAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
LeftTicks	<code>LeftTicks = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);</code>
LeftTitle	<code>LeftTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment, extend);</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
RightAxisTitle	<code>RightAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
RightTicks	<code>RightTicks = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);</code>

RightTitle	<code>RightTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment, extend);</code>
TopAxisTitle	<code>TopAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</code>
TopTics	<code>TopTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);</code>
TopTitle	<code>TopTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);</code>

## Legends

Legends, which are text or text combined with symbols, are often used as titles or as interpretive keys to the chart data. Legends are easily modified to suit any number of potential uses beyond titling a data set.

All Visual Mining charts, except for the Diagram chart, support the display of a legend anywhere on the chart, including inside the grid. The entries for the legend can be fully specified, or can be loaded automatically using the names and colors assigned to all data sets being displayed.

The fundamental parameter for setting up legends is `Legend`. You must use this one if you wish to adjust any other legend-related parameters.

## Parameters Involved, Alphabetically

The following parameters follow the label standard format, allowing you to customize your text.

DataLegend	<code>DataLegend = ON OFF;</code>
DataLegendGrid	<code>DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);</code>
DataLegendGridLine	<code>DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);</code>
Legend	<code>Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);</code>
LegendActiveLabels	<code>LegendActiveLabels = ("Label", "URL", "Target"), ...;</code>
LegendAxis	<code>LegendAxis = (XAxis, Yaxis);</code>
LegendBox	<code>LegendBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
LegendBoxSize	<code>LegendBoxSize = (MaxWidth, MaxHeight);</code>
LegendItems	<code>LegendItems = ("Label1", Color1, SymType1, SymSize1, SymStyle1, LineType1, LineWidth1, LineColor1, Patternfill, Patternforeground, Patternbackground, Image, Shadowwidth),</code>
LegendItemBox	<code>LegendItemBox = (lineType, lineWidth, lineColor);</code>
LegendLayout	<code>LegendLayout = (Type, Location, X, Y, Justify, Columns, ItemOrder,);</code>
LegendSymbolPostion	<code>LegendSymbolPostion = LEFT   RIGHT;</code>

## Notes, or Annotations

When designing a chart, you can place text annotations, or notes, anywhere on the graph, both inside the chart and alongside it. Notes may have optional arrows with up to three bends in them, and three end-

point shapes. Notes may also become targets for drilldowns to other charts and data (see Active Labels, above).

Using Notes, items of interest can be easily identified, and critical regions can be highlighted. Notes may also be used to turn charts into illustrations.

Notes combine the functions of *Label* and *Region*, since they are text on various sorts of box backgrounds. The fundamental parameter for setting up notes is **NoteSets**.

### Parameters Involved, Alphabetically

NoteActiveLabels	NoteActiveLabels[1-20] = ("Label", "URL", "Target"), ...;
NoteArrow	NoteArrow = (LineType1, LineWidth1, LineColor1, ArrowType1, ArrowStyle1), (LineType2, LineWidth2, LineColor2, ArrowType2, ArrowStyle2), ...;
NoteAxis	NoteAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
NoteBox	NoteBox = (Color1, BorderType1, BorderWidth1, ImageURL1, ImageFormat1), ...;
NoteLabel	NoteLabel = ("Label1", Color1, "FontName1", FontSize1, Angle1, , interiorAlignment1), ...;
NoteSets	NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...;
NoteSet[n]	NoteSet1 = ("Text1", X, Y, X1, Y1, X2, Y2, X3, Y3), ...;
NotesDrawnBeforeData	NotesDrawnBeforeData = ON OFF;

## Regions, or Boxes

Region definitions are used extensively throughout Visual Mining charts in displaying titles, legends, data labels, dwell labels, backgrounds, and notes. A region is displayed as a rectangular box with a specified color and border type. In most cases, these boxes are optional and need not appear at all.

Stylistically, we recommend that you match the chart background (as opposed to its grid) to the color of the web page you are using, so that it blends in and does not distract the viewer from the data. Similarly, we recommend that you use boxes for titles, notes, and legends very sparingly, as these can also be distracting, and do not add to the meaning of the graph. Only dwell labels really require some contrasting color or edge décor to set them off when they pop up.

All region parameters use a basic group, or “tuple,” of attributes:

- Main color
- Border type (raised, inset, shadowed, box, or none)
- Border width, in pixels
- Image URL, which indicates an image file that serves at the box color
- Image format (tiled, centered, sized to fit)
- Border color

Additionally some regions support a corner style and corner color. Corner styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, Bottom Left. Legal values are **SQUARE**, **SNIP** and **ROUND**. The default is **SQUARE**. The *CornerColor* attribute specifies the color to fill corners of a region when the *CornerStyle* is **SNIP** or **ROUND**.

**Parameters Involved, Alphabetically**

Background	<code>Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
BottomAxisTitleBox	<code>BottomAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
DwellBox	<code>DwellBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
FooterBox	<code>FooterBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
HeaderBox	<code>HeaderBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
LeftAxisTitleBox	<code>LeftAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
LeftTitleBox	<code>LeftTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
LegendBox	<code>LegendBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
NoteBox	<code>NoteBox = (Color1, BorderType1, BorderWidth1, ImageURL1, ImageFormat1, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor), ...;</code>
PieBackgrounds	<code>PieBackgrounds = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);</code>
RightAxisTitleBox	<code>RightAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
RightTitleBox	<code>RightTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>
TopAxisTitleBox	<code>ToptAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);</code>

## 7. CDL Parameter Definitions

As was discussed in Chapter 2, CDL parameters may either be used in the standard HTML format for applets, or within the `NFPParamScript` format used for applets and templates for other NetCharts solutions

As in the rest of the guide, the attribute list for each CDL parameter points to common attributes, which are defined in Chapter 4, when the attribute name is in *italic*. When you see an attribute listed in *italic*, it means that you should look to that section for specific details about the attribute. If you recognize a term from the list of attributes-in-common, but it is not shown in italic, that means that there is some parameter-specific information about the attribute, so the description is provided with that element.

In the *Examples* for a parameter-specific attribute, the attribute is shown in **bold**, to help you locate it. In actual code, attributes would not be bolded.

### ActiveClicks

---

```
ActiveClicks = Number;
```

In Active Label processing on NetCharts Applets, use `ActiveClicks` to specify a number of mouse clicks needed to activate the label. The user can then click the mouse while an active label is displayed, resulting in a predefined URL being executed. This URL can serve any of three purposes:

- To replace the current HTML document with any other HTML document
- To alter the display of any named frame or window within the browser
- To load new chart parameters from a parameter file

This capability is sometimes referred to as “*drilldown*.”

#### *Used in These Charts*

All

#### *Attributes*

Number

#### *Number*

Number is the number of clicks of the mouse that the user must make to activate a label.

#### *Example:*

```
ActiveClicks = 2;
```

#### *Values*

A whole number, preferably one to three.

#### *Default*

1

## ActiveLabels

---

```
ActiveLabels[N] = ("Label", "URL", "Target"), ...;
```

The `ActiveLabels` parameter is the basis for significant interactivity within the Visual Mining charting applications. For each chart, it can provide a set of one or more labels that respond to mouse dwell or mouse clicks.

If the `DwellLabel` parameter is also defined, then a label will automatically be displayed whenever the mouse cursor dwells over a given data value. The `ActiveLabels` parameter defined for a dwell label parameter then specifies the format of each label, and not the text value.

All of the other `ActiveLabels` attributes are optional and may be used to control the dwell label for one or more data values, axis labels, legend labels or titles. In most charts, the `ActiveLabels` parameter is specified using a numeric ID (such as `ActiveLabels1` or `ActiveLabels5`) to indicate which data set is being referenced. Each tuple defined in the `ActiveLabels` parameter is matched with the corresponding data value defined in the `DataSet` or equivalent parameter. If too many active labels are defined, the extra tuples are ignored. If too few are defined, the remaining data values will use the default dwell label generated.

All of the other `ActiveLabels` attributes are optional and may be used to control the dwell label for one or more data values, axis labels, legend labels or titles. In most charts, the `ActiveLabels` parameter is specified using a numeric ID (such as `ActiveLabels1` or `ActiveLabels5`) to indicate which data set is being referenced. Each grouped, parenthesized set, or “tuple,” defined in the `ActiveLabels` parameter is matched with the corresponding data value defined in the `DataSet` or equivalent parameter. If too many active labels are defined, the extra tuples are ignored. If too few are defined, the remaining data values will use the default dwell label generated.

### Used in These Charts

All

### Example:

```
ActiveLabels = ("", "Barchart9Mon.html", "InfoFrame"),
               ("", "Barchart9Tue.html"),
               ("Barchart9Wed.html"),
               (OUTLINE, "Barchart9Thu.html"),
               (OUTLINE, "Barchart9Fri.html");
```

### Attributes

<i>Label</i>	<i>URL</i>	<i>Target</i>
--------------	------------	---------------

## ActiveLabels[n]

---

```
ActiveLabels[1-50] = ("Label", "URL", "Target"), ...;
```

In most charts, the `ActiveLabels` parameter is specified using a numeric ID (such as `ActiveLabels1` or `ActiveLabels5`) to indicate which data set is being referenced. Each such set defined in the `ActiveLabelsn` parameter is matched with the corresponding data value defined in the `DataSet` or equivalent parameter. If too many active labels are defined, the extra sets are ignored. If too few are defined, the remaining data values will use the default dwell label generated.

*Used in These Charts*

All

*Example:*

```
ActiveLabels2 = ("One Hundred", "Barchart9Mon.html", "InfoFrame"),
                ("One Hundred\nTwenty Five", "noinfo.html"),
                ("Two Hundred\nForty Five", "Barchart9Wed.html"),
                ("OUTLINE", "Barchart9Thu.html"),
                ("OUTLINE", "Barchart9Fri.html");
```

*Attributes*

<i>Label</i>	<i>URL</i>	<i>Target</i>
--------------	------------	---------------

**AddDataPoint**

```
AddDataPoint = ("Name", X, Y, Z, "Label", "URL", "Target"), ...;
```

AddDataPoint is used in bubble charts, but is *only used if you are plotting dynamically*. The attributes correspond with data points named in the BubbleSets parameter. AddDataPoint must be the only command sent that is followed by the Update command. Using AddDataPoint allows the bubble chart to be updated rapidly without refreshing the screen or reconfiguring the layout. The Label, URL, and Target attributes are optional, and if specified, are used to define an ActiveLabel for the given data point. If NULL is substituted for either or both of the (X, Y) pair values, the point will not be drawn.

*Used in These Charts*

Bubble, X-Y

*Example:*

```
AddDataPoint = ("R1", 320, 199, 3.85),
                ("R2", 445.8, 622, 2.21, "Horizon", "horizon.html", "_self");
```

*Attributes*

<i>Label</i>	<i>Name</i>	<i>Target</i>	<i>URL</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
--------------	-------------	---------------	------------	----------	----------	----------

**X, Y, and Z**

X, Y, and Z represent the dynamically updated coordinates of the Named data points vector, along the associated X, Y, and Z axes.

*Example:*

```
AddDataPoint = ("R1", 320, 199, 3.85);
```

*Values*

Real numbers

*Default*

None



---

## AppendDataSet[n]

---

```
AppendDataSet[1-50] = y1, y2, y3, ...;
```

`AppendDataSet` defines a list of Y values for each data set defined by the `DataSets` parameter. The data is appended onto the data queue for the specified `DataSet`. Data is extracted from the queue when an `Update` statement is passed. If the value "null" is substituted for a y value, the symbol is not drawn for that slot.

### *Used in These Charts*

Strip

### *Example:*

```
AppendDataSet1 = 35, 38, 42, 41, 40, 37.5, 36.125, 35, 38;
```

### *Attributes*

Y

Y

---

Y represent the dynamically updated Y-coordinates of the data points for the specific data set.

### *Example:*

```
AppendDataSet1 = 35, 38, 42, 41, 40, 37.5, 36.125, 35, 38;
```

### *Values*

Real numbers

null            The symbol is not drawn for that position

### *Default*

None

---

## AppendPolyDataToActiveLabels

---

```
AppendPolyDataToActiveLabels = ON | OFF;
```

If `AppendPolyDataToActiveLabels` is ON, then Diagram Polygon Active Labels will automatically contain both the polygon label and the current data value associated with the polygon. If the value is OFF, the Diagram Polygon Active Labels will only contain the polygon label. The default value is ON.

### *Used in These Charts*

Diagram

**Example:**

```
AppendPolyDataToActiveLabels = OFF;
```

**Attributes**

Mode

---

**AntiAlias**

---

```
AntiAlias = mode;
```

NetCharts includes support for anti-aliased fonts and or graphics, which can be used to produce smooth text and lines in a chart. The CDL parameter, `AntiAlias = ON|ONDRAW|ONTEXT|OFF`, controls this behavior. `ON` will AntiAlias both text and lines. Starting with version 5.0, the default value is `ON`.

**Used in These Charts**

All

**Example:**

```
AntiAlias = ON;
```

**Attributes**

Mode

---

**AutoscalePad**

---

```
AutoscalePad = pad;
```

`AutoscalePad` specifies a hint to the NetCharts logic that is used automatically determine an axis range. When no `AxisScale` parameters are specified for an axis, the axis automatically chooses a range based on the data values that are being plotted. `Autoscalepad` can be used to control these automatically chosen values.

`AutoscalePad` is specified as a percentage of the data range. For example, if a data series has a minimum value of 100 and a maximum value of 200, NetCharts will autoscale the axis to range from 100 to 200. If in addition `AutoscalePad=10` is specified, NetCharts will autoscale the axis to start at 10% below 100 and end at 10% above 200. If a minimum of maximum value is the data series is 0, that value will not be modified by `AutoscalePad`.

**Used in These Charts**

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
AutoscalePad = 10;
```

---

## AxesGaps

---

```
AxesGaps= value, value, ...;
```

The `AxesGaps` parameter is used to specify the percentage of available space that should be allocated for gaps between axes. This parameter is used in conjunction with `AxesLayout`. The values specified in `AxesLayout` and `AxesGaps` are summed and each axis and gap is drawn in a space relative to its contribution to that sum. For Example `LeftAxisLayout=20, 40, 20` and `LeftAxisGaps=10, 10` would assign 20% of the axis space to Axis1, 10% to the gap between Axis1 and Axis2, 40% to Axis2, 10% to the gap between Axis2 and Axis 3 and 20% to Axis3.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
AxesGaps = 2;
```

### *Attributes*

*Value*

---

## AxesLayout

---

```
AxesLayout = value1, value2, value3, ...;
```

`AxesLayout` is used to specify the percentage of available space that each axis will occupy.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
AxesLayout = 85, 15;
```

### *Attributes*

*Value*

---

## AxesLayoutDirection

---

```
AxesLayoutDirection = mode, ...;
```

The `AxesLayoutDirection` parameter is used to specify the direction of the new axis, `OUTWARD` or `NORMAL`.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
AxesLayoutDirection = NORMAL;
```

**Values**

NORMAL	Draws a 2D bar
OUTWARD	

**Attributes**

Mode

**AxesSizes**

```
AxesSizes = (bottomsize, topsize, leftsize, rightsize);
```

`AxesSizes` specifies the amount of space that will be allocated to draw top, bottom, left and right axes. `AxesSizes` is typically used in conjunction with `PlotArea` to further control the layout of the `PlotArea` of a chart.

`bottomsize` - amount of vertical space in chart allocated to the bottom axes. If `bottomsize` is a number between 0 and 1, it is interpreted as a percentage of the total plot area height. If `bottomsize`  $\geq 1$  it is interpreted as an absolute size in pixels.

`topsize` - amount of vertical space in chart allocated to the top axes. If `topsize` is a number between 0 and 1 it is interpreted as a percentage of the total plot area height. If `topsize`  $\geq 1$  it is interpreted as an absolute size in pixels.

`leftsize` - amount of horizontal space in chart allocated to the left axes. If `leftsize` is a number between 0 and 1, it is interpreted as a percentage of the total plot area width. If `leftsize`  $\geq 1$  it is interpreted as an absolute size in pixels.

`rightsize` - amount of space in chart allocated to the right axes. If `rightsize` is a number between 0 and 1 it is interpreted as a percentage of the total plot area width. If `rightsize`  $\geq 1$  it is interpreted as an absolute size in pixels.

**Used in These Charts**

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
AxesSizes = (1,1,1,1);
```

**Attributes**

<code>bottomsize</code>	<code>topsize</code>	<code>leftsize</code>	<code>rightsize</code>
-------------------------	----------------------	-----------------------	------------------------

## AxisThickness

```
AxisThickness = thickness;
```

The `AxisThickness` parameter defines the axis thickness. If thickness is greater than 1, then all axes will be displayed in 3D with the given thickness. This 3D look will only be active when one or more 3D grids have been specified. The default value is 0.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

#### Example:

```
AxisThickness = 15;
```

### Attributes

*Thickness*

## AxisTitle

```
TopAxisTitle[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
BottomAxisTitle[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
LeftAxisTitle[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
RightAxisTitle[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
```

The `AxisTitle` parameter specifies the label attributes for the axis title, which centered along the given axis, just above the grid. When the `Header` parameter, whether because of the use of a legend, or for some other reason, creates a title that seems visually unbalanced, you may find this parameter produces a more pleasing chart title.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.

The legal values for `interiorAlignment` and `exteriorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

#### Example:

```
TopAxisTitle = ("Ceres Prototype Project Schedule\n ", black, "Helvetica", 12,
LEFT);
```

### Attributes

*Label*      *Color*              *FontName*      *FontSize*      *Angle*      *interiorAlignment*

## AxisTitleActiveLabel

---

```
TopAxisTitleActiveLabel[N] = ("Label","URL","Target"), ...;
BottomAxisTitleActiveLabel[N] = ("Label","URL","Target"), ...;
LeftAxisTitleActiveLabel[N] = ("Label","URL","Target"), ...;
RightAxisTitleActiveLabel[N] = ("Label","URL","Target"), ...;
```

The `AxisTitleActiveLabel` parameter specified a custom active label to be associated with the axis title. That is, these labels will be displayed whenever the mouse “dwells” over the axis title.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
AxisTitleActiveLabel = ("Label", "URL", "Target");
```

### Attributes

`Label`      `Target`      `URL`

## AxisTitleBox

---

```
TopAxisTitleBox[N] = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
BottomAxisTitleBox[N] = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
LeftAxisTitleBox[N] = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
RightAxisTitleBox[N] = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
```

The `AxisTitleBox` parameter specifies the region attributes for the axis title centered along the axis. The image-related attributes need not be used unless you want to use an image texture on the box.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
TopAxisTitleBox = (lightgray, SHADOW, 3,,gray);
```

### Attributes

`Color`                      `BorderType`                      `BorderWidth`                      `ImageURL`  
`ImageFormat`                      `BorderColor`

## AxisZoom

---

```
BottomZoom[N] = ON/OFF;
LeftZoom[N] = ON/OFF;
RightZoom[N] = ON/OFF;
TopZoom[N] = ON/OFF;
```

`AxisZoom` permits one to zoom into a portion of a chart. When `AxisZoom = ON`, clicking and dragging a rectangle over a chart will zoom with respect to that axis on the chart. This zoom operation does not require the axis to be scrollable.

**Example:**

```
LeftZoom = ON;
```

**Attributes**

(Switch)

## Background

---

```
Background = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle,
BRCornerStyle, BLCornerStyle, CornerColor);
```

The `Background` parameter, which is universal to Visual Mining charting programs, provides a visual background for the chart. It can have a border with a width of your choice, or have an image arranged in a variety of ways. We recommend that you use `Background` to make the chart blend in with your HTML page.

**Used in These Charts**

All

**Example:**

```
Background = (darkPink, NONE, 0, "../classes/netcharts/demo/flock.gif", SQUARE,
SNIP, SQUARE, SNIP, white);
```

**Attributes**

<i>BorderColor</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>Color</i>
<i>ImageFormat</i>	<i>ImageURL</i>	<i>TRCornerStyle</i>	<i>BRCornerStyle</i>
<i>BLCornerStyle</i>	<i>CornerColor</i>		

## XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

## BackgroundActiveLabel

---

```
BackgroundActiveLabel = ("Label", "URL", "Target");
```

`BackgroundActiveLabel` defines a single active label content and destination for the background of a chart.

*Used in These Charts*

All

*Example:*

```
BackgroundActiveLabel = ("Overview chart, click for details", "detail.html",
    "_frame1");
```

*Attributes*

*Label*                      *URL*                      *Target*

**BackgroundFillPattern**

BackgroundFillPattern = (type, color1, color2, imageURL), ...;

The BackgroundFillPattern parameter provides a visual pattern fill for the background of a chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - Foreground color for patterns - Starting color for gradients - Ignored in images	
<b>color 2</b>	This color is used in the following ways: - Background color for patterns - Stopping color for gradients - Ignored in images	
<b>imageURL</b>	The URL to an image to use as the fill	



### *Used in These Charts*

All

#### **Example:**

```
BackgroundFillPattern = (GRADIENTVERTICAL, blue, white);
```

#### **Attributes**

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

---

## Bar3DDepth

---

```
Bar3DDepth = Number;
```

Bar3DDepth defines the depth of the bars in a bar chart.

### *Used in These Charts*

Bar, Combo, Pareto, Stock

#### **Example:**

```
Bar3DDepth = 10;
```

#### **Attributes**

*Number*

#### **Number**

---

Apparent depth of a bar in a bar chart, in pixels.

#### **Example:**

```
Bar3DDepth = 5;
```

#### **Values**

0	Draws a 2D bar
1 or greater	Whole number depth in pixels

#### **Default**

10

---

## BarActiveLabels

---

```
BarActiveLabels = ("Label", "URL", "Target"), ...;
```

Specifies sets of active labels attached to bars in a bar chart. Each grouped set in parentheses, or “tuple,” has a corresponding set within a `DataSet` parameter.

### *Used in These Charts*

Bar, Combo, Pareto, Stock

### *Example:*

```
BarActiveLabels = ("OUTLINE", "Barchart8Mon.html", "InfoFrame"),
                  ("OUTLINE", "Barchart8Tue.html", InfoFrame),
                  ("No Log Info", "noinfo.html", InfoFrame);
```

### *Attributes*

<i>Label</i>	<i>URL</i>	<i>Target</i>
--------------	------------	---------------

## **BarAnimationStyle**

---

```
BarAnimationStyle = GROW | FADE | NONE
```

Specifies how the bars initially appear in a chart. This parameter is only valid in SVG or SVGWeb output formats.

### *Attributes*

*Style*

### *Style*

---

*Style* refers to the manner in which bars are first rendered in a bar chart.

### *Example:*

```
BarAnimationStyle = GROW;
```

### *Values*

GROW            The bars grow from the origin of the axis to their actual values.

FADE            The bars fade in.

NONE            The bars are immediately visible.

### *Default*

NONE

## **BarBorder**

---

```
BarBorder[N] = (LineType, LineWidth, Color);
```

For graphs using bars, `BarBorder` specifies the line style to be used for the border of all bars. The default line color is black.

**Used in These Charts**

Bar, Combo, Pareto, Stock

**Example:**

```
BarBorder = (DASHED, 2, DarkGray);
```

**Attributes**

LineType	LineWidth	Color
----------	-----------	-------

**LineType**

LineType specifies the style of the border to be drawn on a chart's bars.

**Example:**

```
BarBorder = (DASHED, 2, DarkGray);
```

**Values**

SOLID	Draws a solid line of LineWidth thickness.
DOTTED	Draws a dotted line of LineWidth thickness.
DASHED	Draws a dashed line of LineWidth thickness.
DOTDASH	Draws a dot-dashed line of LineWidth thickness.

**Default**

SOLID

**LineWidth**

LineWidth specifies the width in pixels of the border to be drawn on a chart's bars.

**Example:**

```
BarBorder = (DASHED, 2, DarkGray);
```

**Values**

1 or greater    Whole number width in pixels

**Default**

1

**BarColorTable[n]**

```
BarColorTable[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

BarColorTable defines a set of colors for dataset N that overrides all other color specifications for that set. The parameters used for specifying the color of bars in a chart are (in ascending order of precedence)

ColorTable, DataSets, BarSymbol, BarFillPattern and BarColorTable. BarColorTable is used most frequently to color some specific bar.

For example

```
BarColorTable2 = , , blue;
```

will change the third bar in the second series to blue, while all other bars in the chart continue to be colored by one of the other color related parameters.

The colors you can use are defined in the common Color attribute (Chapter 4) .

### *Used in These Charts*

Bar, Combo, Pareto, Stock

### *Example:*

```
BarColorTable2 = , , red;
```

### *Attributes*

None

## **BarColorTable[n]P[m]**

---

```
BarColorTable[1-50]P[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

BarColorTable[n]P[m] is used only for grouped stacked BarCharts (GraphType=GROUPSTACK). It defines a set of colors for datasetN that overrides all other color specifications for that set. The parameters used for specifying the color of bars in a chart are (in ascending order of precedence) ColorTable, DataSets, BarSymbol, BarFillPattern and BarColorTable[1-50]M[1-50]. BarColorTable is used most frequently to color some specific bar.

For example:

```
BarColorTable2P1 = blue,blue;
```

will change the color the first bar of the second data series blue at the first tic location and at the second tic location.

## **BarCorners**

---

```
BarCorners = Number;
```

BarCorners specifies the corner style to be used for the 2D bars in a chart. The attributes correspond to specific corners of the bar beginning at the top left and advancing in clockwise sequence to the bottom left. The default corner style is SQUARE. Elements which can be selected for each corner are: SQUARE, SNIP and ROUND. SNIP trims the square corner from the bar obliquely. ROUND substitutes a smoothly rounded edge for the square corner. The amount to be snipped or rounded can be adjusted by adding an underscore modifier to the elements SNIP and ROUND followed by a value. A whole number value will set the number of pixels to be removed. A fractional value will set the percentage of the overall bar to be removed. To set the percentage, enter a fractional value between 0 and 1 and place it after the underscore (e.g. "SNIP\_0.09" will cause 9% of the corner to be removed; "ROUND\_0.50" will cause 50% of the

corner to be rounded). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point.

### *Used in These Charts*

Bar, Combo, Pareto, Stock

### *Example:*

```
BarCorners = ("SNIP_0.09", "SNIP_0.09", SQUARE, SQUARE);
```

### *Attributes*

TopLeft                      TopRight                      BottomRight                      BottomLeft

---

### *TopLeft*

TopLeft specifies the style and size of the upper left corner to be drawn on a chart's bars.

### *Example:*

```
BarCorners = (SNIP, SQUARE, SQUARE, SQUARE);
```

### *Values*

SQUARE	Draws a square corner.
SNIP	Cuts the corner off of obliquely.
ROUND	Draws a rounded corner.
SNIP_val	Cuts the corner off obliquely based upon the number of pixels or percentage shown in <i>val</i> .
ROUND_val	Draws a rounded corner based upon the number of pixels or percentage shown in <i>val</i> .

### *Default*

SQUARE

---

### *TopRight*

TopRight specifies the style and size of the upper right corner to be drawn on a chart's bars.

### *Example:*

```
BarCorners = (SQUARE, SNIP, SQUARE, SQUARE);
```

---

### *BottomRight*

BottomRight specifies the style and size of the lower right corner to be drawn on a chart's bars.

### *Example:*

```
BarCorners = (SQUARE, SQUARE, SNIP, SQUARE);
```

## BottomLeft

---

`BottomLeft` specifies the style and size of the lower left corner to be drawn on a chart's bars.

### Example:

```
BarCorners = (SQUARE, SQUARE, SNIP, SQUARE);
```

## BarDropShadow

---

```
BarDropShadow = (color, offsetx, offsety, size);
```

`BarDropShadow` places a shadow on the background field of the bar chart. The color, orientation, and size of the shadow can be defined. The tuple element `color` sets the color of the shadow. That color value is interpolated to complete transparency as it reaches the end of the shadow's blur area. `offsetx` and `offsety` define the center point of the shadow; `offsetx` sets the x-axis offset from the chart's center-point; `offsety` sets the y-axis offset. When an offset attribute is set to a whole number value, the position of the drop shadow is offset from the chart's center point by the number of pixels set by that whole number. When an offset is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow (the set of bars). The center of the drop shadow is repositioned based upon the values or percentages set for `offsetx` and `offsety`. Offset attribute values may be positive or negative. `size` sets the size of the blur area, the region beyond the actual drop shadow shape where the shadow is extended and blurred into transparency. The size of this blurred region is controlled by the `size` attribute. The blurred region becomes larger and more diffuse as the value of `size` is increased. When `size` is set to a whole number value, the size of the blurred area is defined in pixels. When `size` is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow. When using a fractional value, enclose the value in double-quotes to "escape" the decimal point.

### Used in These Charts

Bar, Combo, Pareto

### Example:

```
BarDropShadow = (color, offsetx, offsety, size);
```

### Attributes

Color	Offsetx	offsety	Size
-------	---------	---------	------

### Color

---

`Color` specifies the base color of the shadow to be drawn behind a chart's bars.

### Example:

```
BarDropShadow = (black, "-.05", "-0.05", 55);
```

---

## Offsetx

---

`Offsetx` specifies the x-coordinate offset from center.

### Example:

```
BarDropShadow = (black, 25 -10, 25);
```

---

## Offsety

---

`Offsety` specifies the y-coordinate offset from center.

### Example:

```
BarDropShadow = (black, 25 -10, 25);
```

---

## Size

---

`Size` specifies the width of the blurred area.

### Example:

```
BarDropShadow = (black, "-.05","-.05", 55);
```

---

## BarFillPattern

---

`BarFillPattern[N]` = (*type, color1, color2, imageURL, ...*);

The `BarFillPattern` parameter provides a visual pattern fill for bars in a bar or combo chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient

	GRADIENRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
<b>Images</b>		
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

Bar, Combo, Pareto

### Example:

```
BarFillPattern = (GRADIENTVERTICAL,blue,white), ...;
```

### Attributes

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

## BarHighlights

```
BarHighlights = (type,start,stop,top,right, bottom,left,width,height,topLeft,topRight,bottomRight,bottomLeft), ...;
```

The `BarHighlights` parameter provides a visual pattern fill in a bar or combo chart. It adds or overlays a fill pattern over one or more existing fill patterns to produce multiple effects. The *width* or *height* of the pattern can be modified. The gap between the sides of the bar and the fill pattern being applied can be modified. Gradient patterns can be set using the *type* attribute. Only gradient patterns may be used. A *type* value of NONE suppresses the highlights. The element *start* sets the beginning color of the gradient; the element *stop* sets the end color of the gradient. Color values are interpolated between the two. The elements *top*, *right*, *bottom*, and *left* specify the size of the gaps between the edge of the highlight and the associated edge of the bar. When the values for *top*, *right*, *bottom*, and *left* are specified as whole numbers, they set the distance between the edge of the highlight and the edge of the bar in pixels. When set to a fractional number between 0 and 1, they set the gap to a percentage of the width of the bar. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point. The element *width* controls the width of the highlight; the element *height* controls the height. When values for *width* and *height* are specified using whole numbers, they set the distance in pixels. If *width* or *height* is set to -1, `BarHighlights` fills all of the space available after taking into account any gaps specified in



previous attributes. If *width* or *height* is set to a fractional number between 0 and 1, `BarHighlights` sets the width or height of the highlight using the percentage of available width or height of the bar.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENTRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
<b>start</b>	This color is used in the following ways: - <i>Starting color for gradients</i>	
<b>stop</b>	This color is used in the following ways: - <i>Stopping color for gradients</i>	

### Used in These Charts

Bar, Combo, Pareto, Stock

### Example:

```
BarHighlights = (GRADIENTRADIAL, yellow, white, 15, 15, 15, 15, -1, -1);
```

### Attributes

<i>type</i>	<i>start</i>	<i>stop</i>
<i>top</i>	<i>right</i>	<i>left</i>
<i>width</i>	<i>height</i>	

## BarRightFillPattern

```
BarRightFillPattern = (type, color1, color2, imageURL), ...;
```

The `BarRightFillPattern` parameter provides a visual pattern fill for the right surface of a bar in a 3D bar 3D combo or Pareto chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)

	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

**Used in These Charts**

Bar, Combo, Pareto, Stock

**Example:**

```
BarRightFillPattern = (GRADIENTVERTICAL,blue,white), ...;
```

**Attributes**

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

**BarRightFillPattern[n]P[m]**

```
BarRightFillPattern[1-50]P[1-50] = (type, color1, color2, imageURL), ...;
```

The `BarRightFillPattern[n]P[m]` parameter provides a visual pattern fill for the right surface of a bar in a 3D stacked bar 3D combo chart or Pareto chart. Used for bar series in grouped stacked BarCharts (GraphType=GROUPSTACK). In grouped stacked BarCharts, `BarRightFillPattern[n]P[m]` defines the right fill pattern applied to a 3D stack at a single tic location. See also: `DataSet[n]P[m]`.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type

	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

StackedBar

#### Example:

```
BarTopRightFillPattern1P2 = (GRADIENTVERTICAL,blue,white), ...;
```

#### Attributes

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

## BarSpotlights

```
BarSpotlights = (start,stop,center,centeroffsetx,centeroffsety, focusoffsetx, focusoffsety,radius),...;
```

Adds or overlays a color fill over one or more existing fill patterns to produce multiple layered color effects. The spotlight “illuminates” the bars of the bar or combo chart. The center of the spotlight and its focus can be adjusted independently by adjusting offsets from the bar chart center point. The elements *centeroffsetx* and *centeroffsety* set the x and y-coordinates of the center of the spotlight as an offset of the chart center point. When set to whole numbers, *centeroffsetx* and *centeroffsety* specify the number of pixels to offset

from the chart center point. When set to fractional values (between 0 and 1), they are interpreted as percentages of the width of the bars. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point. The element *focusoffsetx* is the offset from the chart center which defines the x-coordinate of the focus point of the spotlight. The element *focusoffsety* is the offset from the chart center which defines the y-coordinate of the focus point of the spotlight. When set to whole numbers, *focusoffsetx* and *focusoffsety* specify the offset from the center in pixels. When set to fractional values (between 0 and 1), they are interpreted as percentages of the width of the bars. The element *radius* sets the size of the spotlight, from its center to its edge. When set to a whole number, it sets the size of the *radius* in pixels; when set to a fractional value, it sets the radial diameter of the spotlight based upon that percentage of the minimum height and width of the bars on the chart.

### Used in These Charts

Bar, Combo, Pareto, Stock

#### Example:

```
BarSpotlights = (purple_40,blue_155,RIGHT,50,-50,100,150,250);
```

#### Attributes

start	stop	center
centeroffsetx	centeroffsety	focusoffsetx
focusoffsety	radius	

### Start

---

*Start* specifies the first of the two colors which will be interpolated to produce a gradient spotlight.

#### Example:

```
BarSpotlights = (purple_40, blue_155, LEFT, 20,10,120,-120,250);
```

### Stop

---

*Stop* specifies the second of two colors which will be interpolated to produce a gradient spotlight.

#### Example:

```
BarBorder = (DASHED, 2, DarkGray);
```

### Center

---

*Center* specifies the position around of the center of the chart where the spotlight center will be placed.

#### Example:

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Values**

RIGHT	Offsets the center point of the spotlight to the right of the center point of the chart.
LEFT	Offsets the center point to the left.
TOP	Offsets the center point to the top.
BOTTOM	Offsets the center point to the bottom.
CENTER	Uses the chart center point for the spotlight center point.
TOPRIGHT	Offsets the center point of the spotlight to the top right.
TOPLEFT	Offsets the center point of the spotlight to the top left.
BOTTOMRIGHT	Offsets the center point of the spotlight to the bottom right.
BOTTOMLEFT	Offsets the center point of the spotlight to the bottom left.

**Default**

CENTER

**Centeroffsetx**

---

Centeroffsetx specifies the x-coordinate offset for the spotlight center.

**Example:**

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Centeroffsety**

---

Centeroffsety specifies the y-coordinate offset for the spotlight center.

**Example:**

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Focusoffsetx**

---

Focusoffsetx specifies the x-coordinate offset for the center of the spotlight's focus.

**Example:**

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Focusoffsety**

---

Focusoffsety specifies the y-coordinate offset for the center of the spotlight's focus.

**Example:**

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

---

## Radius

---

`Radius` specifies the length of the radius of the spotlight from the center of the spotlight.

### Example:

```
BarSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

---

## BarStyle

---

`BarStyle[N]` = (*mode, color, font name, width*); Remember to change these VALUES

Defines the label value to be displayed on each group of bars. This parameter overrides the `BottomLabels` parameter (for VERTICAL bars) and the `LeftLabels` parameter (for HORIZONTAL bars).

### Used in These Charts

Bar, Combo, Pareto

### Example:

```
BarValueLabel = ("ON", black, "Helvetica", 18);
```

### Attributes

Mode    Color    Font Name    Width

---

## BarSymbol

---

`BarSymbol` = (*BarSymbolType, BarSymbolColor*),...;

Defines the style and color of the bars in a barset. Values specified in this parameter override values set in the `DataSets` parameter.

### Used in These Charts

Bar, Combo

### Example:

```
BarSymbol = (CYLINDER,blue), (RECTANGLE,red);
```

### Attributes

`BarSymbolType`    `BarSymbolColor`

---

### *BarSymbolType*

---

The visualization style for the bars in a barset. Legal values are BAR, TRIANGLEBAR, DIAMONDBAR, CYLINDER, PIEHORIZONTAL and PIEVERTICAL. The default value is BAR.

## BarTopFillPattern

BarTopFillPattern[N] = (type, color1, color2, imageURL), ...;

The BarTopFillPattern parameter provides a visual pattern fill for the top surface of a bar in a 3D bar 3D combo or Pareto chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - Foreground color for patterns - Starting color for gradients - Ignored in images	
<b>color 2</b>	This color is used in the following ways: - Background color for patterns - Stopping color for gradients - Ignored in images	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

Bar, Combo, Pareto, Stock

### Example:

```
BarTopFillPattern = (GRADIENTVERTICAL,blue,white), ...;
```

### Attributes

Type	Color1	Color2
ImageFormat	ImageURL	

## BarTopFillPattern[n]P[m]

BarTopFillPattern[1-50]P[1-50] = (*type, color1, color2, imageURL*), ...;

The BarTopFillPattern[n]P[m] parameter provides a visual pattern fill for the top surface of a bar in a 3D stacked bar 3D combo chart or Pareto chart. Used for bar series in grouped stacked BarCharts (GraphType=GROUPSTACK). In grouped stacked BarCharts, BarTopFillPattern[n]P[m] defines the top fill pattern applied to a 3D stack at a single tic location. See also: DataSet[n]P[m].

Type	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENTRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

StackedBar

### Example:

```
BarTopFillPattern1P2 = (GRADIENTVERTICAL,blue,white), ...;
```



**Attributes**

Type	Color1	Color2
ImageFormat	ImageURL	

---

**BarValueLabel**

---

BarValueLabel[N] = (mode, color, font name, width);

Defines the label value to be displayed on each group of bars. This parameter overrides the BottomLabels parameter (for VERTICAL bars) and the LeftLabels parameter (for HORIZONTAL bars).

**Used in These Charts**

Bar, Combo, Pareto

**Example:**

```
BarValueLabel = ("ON", black, "Helvetica", 18);
```

**Attributes**

Mode	Color	Font Name	Width
------	-------	-----------	-------

---

**BarValueLabelBox**

---

BarValueLabelBox = (color, mode, depth);

Defines the label box to be displayed with each group of bars.

**Used in These Charts**

Bar, Combo, Pareto

**Example:**

```
BarValueLabelBox = (grey, RAISED, 3);
```

**Attributes**

Color	Mode	Depth
-------	------	-------

---

**BarValueLabelStyle**

---

BarValueLabelStyle = labelposition1, labelposition2, ... labelpositionN;

Defines where the BarValueLabel text will display for each dataset.

**Used in These Charts**

Bar, Combo, Pareto

**Example:**

```
BarValueLabelStyle = MIDDLE, MIDDLE, MIDDLE;
```

**Attributes**

Label Position

---

**BarWidth**

---

```
BarWidth = Percent1,Percent2,... PercentN;
```

Defines the relative width of the bars in a barset as a percentage of available space. Legal values are 1-100. A smaller number results in more space between bars. WidthN defines the Width for DataSetN.

**Used in These Charts**

Bar, Combo, Histogram, Pareto, Stock

**Example:**

```
BarWidth = 60,30; <!-- Barset1 bars fills 60% of the space available -->
```

**Attributes**

---

**BestFit**

---

```
BestFit = ON/OFF
```

BestFit when turned ON attempts to auto fit each pie and its labels into a square.

**Used in These Charts**

MultiPie

**Example:**

```
BestFit = ON
```

---

**BottomActiveLabels**

---

```
BottomActiveLabels = ("Label", "URL", "Target"), ...;
```

The bottom axis labels become active labels when this parameter is used. Each set in parenthesis has a corresponding set within a DataSet parameter.

**Used in These Charts**

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
BottomActiveLabels = ("", "../cgibin/LA01.cgi", "frame1"),
                    ("OUTLINE", "../cgibin/LA02.cgi", "frame1"),
                    ("", "../cgibin/LA03.cgi", "frame1");
```

**Attributes**

*Label*                      *URL*                      *Target*

**BottomAxisTitleActiveLabel**

```
BottomAxisTitleActiveLabel = ("Label", "URL", "Target");
```

`BottomAxisTitleActiveLabel` defines a single active label destination for the `BottomAxisTitle` parameter. Using this element also requires use of the `ActiveClicks` parameter if you wish to navigate from this interaction.

**Used in These Charts**

All

**Example:**

```
BottomAxisTitleActiveLabel = ("Destination", "demo.html", "frame1");
```

**Attributes**

*Label*                      *URL*                      *Target*

**BottomAxisTitleBox**

```
BottomAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor,
TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `BottomAxisTitleBox` parameter specifies the region attributes for the axis title centered along the axis. The image-related attributes need not be used unless you want to use an image texture on the box.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
BottomAxisTitleBox = ("lightgray", SHADOW, 3,,,gray);
```

**Attributes**

*Color*                      *BorderType*                      *BorderWidth*                      *ImageURL*  
*ImageFormat*      *BorderColor*

**BottomColor**

```
BottomColor = Color;
```

`BottomColor` controls the color of the bottom axis and the tic marks, but not the tic mark labels. The default axis color is black. If the `NULL` color is specified, the axis color is not changed by this parameter.

**Example:**

```
BottomColor = xB5D5F0;
```

**Attributes**

Color

---

## BottomDrawMinorTics

---

```
BottomDrawMinorTics = ON/OFF;
```

`BottomDrawMinorTics` controls whether or not bottom tics are drawn. The default value is ON.

**Example:**

```
BottomDrawMinorTics = OFF;
```

**Attributes**

(Switch)

---

## BottomFormat

---

```
BottomFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
```

The `BottomFormat` parameter defines the format of an axis' numeric tic labels.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
BottomFormat = (INTEGER);  
BottomFormat = (FLOAT, "$%,9.2f", ,);  
BottomFormat = (DATE, "%M/%y", "1/1/00 12:00", "1M");  
BottomFormat = (INTEGER, "$%dK");
```

**Attributes**

FormatType    FormatExpr    TimeBase    TimeUnit

---

## BottomLabels

---

```
BottomLabels = "Label1", "Label2", ...;
```

The `BottomLabels` parameter specifies a list of custom tic mark labels that will be used instead of the numeric labels automatically generated by the axis. The `BottomLabels` will be evenly placed along the axis, overriding any tic placement specified by the `StepValue` attribute.

In a Bar, Combo, Pareto, or Stock Chart, the `BarLabels` parameter overrides the `BottomLabels` (for vertical bars) parameters.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
BottomLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie",  
              "Steve", "Paul";
```

### *Attributes*

`Label`

---

## BottomMargins

---

```
BottomMargins = (LeftSideMargin, RightSideMargin);
```

The `BottomMargins` parameter specifies the gap, in pixels, at the beginning and end of the bottom axis. Most often used to prevent clipping of data points at the extreme ends of the scale.

### *Example:*

```
BottomMargins = (20, 20);
```

---

## BottomScale

---

```
BottomScale = (MinValue, MaxValue, StepValue);
```

The `BottomScale` parameter specifies the minimum and maximum data values which will be displayed along the bottom axis. If the `BottomScale` parameter is not defined, or the `MinValue` and `MaxValue` parameters are the same or one of them is not defined, then the tic mark locations will be automatically determined based on the actual data values being displayed. That is, the axis will be "autoscaled" using the current data values to determine "reasonable" values for `MinValue`, `MaxValue` and `StepValue`. If values are supplied for any of `MinValue`, `MaxValue`, or `StepSize`, those values will be used as part of the autoscaling.

In Bar charts, the `BottomScale` parameter automatically determined by the number of bars being displayed.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Time, X-Y

### *Example:*

```
BottomScale = ("1 Apr 96", "1 Jun 96", "14d");
```

### *Attributes*

`MinValue`                      `MaxValue`                      `StepValue`

---

## *MinValue*

---

`MinValue` sets the absolute lower visible limit for the bottom axis scale.

### *Example:*

```
BottomScale = ("1 Apr 96", "1 Jun 96", "14d");
```

### *Values*

Any real number, date, or time less than `MaxValue`

### *Default*

None

---

## *MaxValue*

---

`ScrollMax` sets the absolute upper visible limit for the bottom axis scale.

### *Example:*

```
BottomScale = ("1 Apr 96", "1 Jun 96", "14d");
```

### *Values*

Any real number, date, or time greater than `MinValue`

### *Default*

None

---

## *StepValue*

---

`StepValue` is optional, and may be used to specify a given step between tic marks along the bottom axis, starting with the `MinValue`. If `StepValue` is not an even multiple of the difference between the `MinValue` and `MaxValue`, then no tic mark will be displayed at the `MaxValue`. If `StepValue` is not defined, tic marks will be placed at "reasonable" locations along the axis, depending on the range of values being displayed.

### *Example:*

```
BottomScale = ("1 Apr 96", "1 Jun 96", "14d");
```

### *Values*

Any real number, date, or time between `MinValue` and `MaxValue`

### *Default*

1

---

## **BottomScroll**

---

```
BottomScroll = (ScrollMin, ScrollMax);
```

The `BottomScroll` parameter specifies a range of values through which the bottom axis can be scrolled. When the `ScrollMin` and `ScrollMax` attributes are defined for the axis, the axis will be displayed as a slider bar, using the axis color defined, with a white background. The relative size of the slider represents the percentage of the entire range currently being displayed. That is, it graphically depicts the size of the current axis range (`MinValue` and `MaxValue` attributes) relative to the scrollable region (`ScrollMin` and `ScrollMax` attributes). See the `BottomScale` parameter for `MinValue` and `MaxValue` definitions.

`BottomScroll` should only be used in conjunction the `BottomScale` parameter.

**Example:**

```
BottomScroll = (0, 98);
```

**Attributes**

`ScrollMin`                      `ScrollMax`

**ScrollMin**

---

`ScrollMin` sets the lower visible limit for a scrollbar defined with `BottomScroll`

**Example:**

```
BottomScroll = (0, 98);
```

**Values**

<MinValue

**Default**

None

**ScrollMax**

---

`ScrollMax` sets the upper visible limit for a scrollbar defined with `BottomScroll`

**Example:**

```
BottomScroll = (0, 98);
```

**Values**

> MaxValue

**Default**

None

**BottomTicLength**

---

```
BottomTicLength = Number;
```

The `BottomTicLength` parameter defines the size of axis tic marks which are displayed along the bottom axis of a chart. The parameter will reset the automatically generated tic length. The value defines the number of pixels to use for the length of the tic mark. By default, the number of pixels used is the

width of the character zero (0) as found in the font applied to the label. Setting the `BottomTicLength` to the value -1 will cause the default size to be used.

### Attributes

*Number*

### Number

Apparent length of a bottom axis tic mark in a chart, in pixels.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

#### Example:

```
BottomTicLength = 10;
```

### Values

0 No effect (zero length tics are not drawn).  
1 or greater Whole number length in pixels

### Default

-1

## BottomTics

```
BottomTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment,backgroundColor,rotationPoint);
```

The `BottomTics` parameter specifies the label attributes for the tic marks displayed along the bottom axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the `BottomTics` parameter. If any attribute is not defined, any previous value of that attribute will be used.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>backgroundColor</code>	Background color of the tic label area
<code>rotationPoint</code>	For rotated axis labels, anchor point for the rotation

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

The legal values for `rotationPoint` are `LEFT`, `RIGHT`.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

#### Example:

```
BottomTics = ("ON", black, "Helvetica", 10, LEFT,null,RIGHT);
```



**Attributes**

Mode	Color	FontName	FontSize	Angle
InteriorAlignment	BackgroundColor	RotationPoint		

**Mode**

Mode determines whether or not the tic labels are shown on that axis.

**Example:**

```
BottomTics = ("ON", black, "Helvetica", 10, LEFT,null,RIGHT);
```

**Values**

ON Show tic labels for this axis  
 OFF Don't show tic labels on this axis

**Default**

ON

**BottomTitle**

```
BottomTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);
```

BottomTitle describes an optional title, or label, that sits on the bottom of a chart, and uses standard attributes for string text, text color, font, font size, and label rotation. As with , BottomTitle is universally available in Visual Mining chart applications.

interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
exteriorAlignment	Specifies the alignment for the entire Title object.

The legal values for interiorAlignment and exteriorAlignment are LEFT, RIGHT, or CENTER.

**Used in These Charts**

All, but most useful in Pie, Diagram, and Dial, which do not have axes.

**Example:**

```
BottomTitle = ("Financial Status", royalblue, Helvetica, 14, 0);
```

**Attributes**

Label	Color	FontName	FontSize	Angle	interiorAlignment
-------	-------	----------	----------	-------	-------------------

**BottomTitleBox**

```
BottomTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);
```

The BottomTitleBox specifies a background region just for the BottomTitle parameter.

### *Used in These Charts*

All

### *Attributes*

*BorderColor*                    *BorderType*                    *BorderWidth*                    *Color*  
*ImageFormat*    *ImageURL*

---

## BoxActiveLabels

---

```
BoxActiveLabels = ("Label","URL","Target"), ...;
```

The `BoxActiveLabels` parameter specified a custom active label to be associated with the `BoxLabels` on a box chart. That is, these labels will be displayed whenever the mouse “dwells” over the optional name of a box data series.

### *Used in These Charts*

Box

### *Example:*

```
BoxActiveLabels = ("Set1", "URL", "Target");
```

### *Attributes*

*Label*                    *Target*                    *URL*

---

## BoxFence

---

```
BoxFence = Mode;
```

`BoxFence` specifies whether or not to draw fences at the ends of the Inter Quartile Range (the box).

### *Used in These Charts*

Box Chart

### *Example:*

```
BoxFence = ON;  
BoxFence = OFF;
```

### *Attributes*

Type

### *Type*

---

Type refers to the fences at the ends of the box.

**Example:**

```
BoxFence = ON;
```

**Values**

ON Draw fences at the ends of the Inter Quartile Range, (the box).  
 OFF Do not draw fences.

**Default**

ON

## BoxFillPattern

BoxFillPattern[N] = (type, color1, color2, imageURL), ...;

The BoxFillPattern parameter provides a visual pattern fill for a box displayed in a chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - Foreground color for patterns - Starting color for gradients - Ignored in images	
<b>color 2</b>	This color is used in the following ways: - Background color for patterns - Stopping color for gradients - Ignored in images	
<b>imageURL</b>	The URL to an image to use as the fill	

### *Used in These Charts*

Box

#### **Example:**

```
BoxFillPattern = (GRADIENTVERTICAL,blue,white);
```

#### **Attributes**

Type	Color1	Color2
ImageFormat	ImageURL	

## **BoxHeight**

---

```
BoxHeight[N] = Height;
```

BoxHeight describes the height (in pixels) for every box displayed.

### *Used in These Charts*

Box Chart

#### **Example:**

```
Boxheight = 5;
```

#### **Attributes**

Height

#### **Height**

---

Height determines box height in whole pixels.

#### **Example:**

```
Boxheight = 5;
```

#### **Values**

0 Default height will be selected based on the size of the display and the number of data sets being displayed

The default height will also be used if the specified height is taller than the amount of space that is physically available to a given box.

#### **Default**

Calculated value

---

## BoxLabels

---

```
BoxLabels = "Label1","Label2", ...;
```

The BoxLabels parameter specifies optional label to be associated with the data series on a box chart. If the `GraphLayout` of the `BoxChart` is `VERTICAL` then BoxLabels will be displayed on the Bottom Axis. If the `GraphLayout` of the `BoxChart` is `HORIZONTAL` then BoxLabels will be displayed on the Left Axis.

### *Used in These Charts*

Box

### *Example:*

```
BoxLabels = "Set1", "Set2";
```

---

## BoxLimitLines

---

```
BoxLimitLines = (limit1-1,limit1-2,...limit1-N),...(limitM-1, limitM-2,... limitM-N);
```

BoxLimitLines allows the user to specify the values for the limit lines for individual data series.

### *Used in These Charts*

Box Chart

### *Example:*

```
BoxLimitLines = (2,4,6), (3,6,9), ...;
```

### *Attributes*

*Limit*

---

## BoxLimitLineStyle

---

```
BoxLimitLineStyle = (type1,width1,color1),...(typeN,widthN,colorN);
```

BoxLimitLineStyles defines the look of the limit lines for each data series. All limit lines for a single data series must have the same style.

The legal values for `type` are `SOLID`, `DASHED`, `DOTTED` and `DOTDASH`.

### *Used in These Charts*

Box Chart

### *Example:*

```
BoxLimitLineStyle = (SOLID,1,red), (SOLID,1,green);
```

### *Attributes*

Type Width Color

## **BoxSymbolWidth**

---

`BoxSymbolWidth = percentage;`

`BoxSymbolWidth` controls the width of the box relative to the width of the fences. It is specified as a percent. A value of 100 would cause the box width and fence width to be equal.

### *Used in These Charts*

Box Chart

### *Example:*

```
BoxSymbolWidth = 95;
```

### *Attributes*

*Width*

### *Default*

95

## **BoxWidth**

---

`BoxWidth = percentage;`

`BoxWidth` controls the width of the box and fences relative to the space allotted to the data series. It is specified as a percent. If a dataset is allocated 100 pixels of space in the chart, setting the `BoxWidth` to 50 would cause the box and fences to use half of that space for their width.

### *Used in These Charts*

Box Chart

### *Example:*

```
BoxWidth = 75;
```

### *Attributes*

*Width*

### *Default*

75

## BubbleSymbolAnimationStyle

---

```
BubbleSymbolAnimationStyle = SCALE | FADE | NONE
```

Specifies how the bubbles initially appear in a bubble chart. This parameter is only valid in SVG or SVGWeb output formats.

### Attributes

Style

### Style

---

style refers to the manner in which bubbles are first rendered in a bubble chart.

### Example:

```
BubbleSymbolAnimationStyle = SCALE;
```

### Values

SCALE      The bubbles grow from a diameter of zero to their actual diameters.

FADE        The bubbles fade in.

NONE        The bubbles are immediately visible.

### Default

NONE

## BubbleAxis

---

```
BubbleAxis[N] = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
```

BubbleAxis indicates one or more axes for the bubble graph, which must correspond to matching data sets.

### Used in These Charts

Bubble

### Example:

```
BubbleAxis = (BOTTOM, LEFT), (BOTTOM, RIGHT);
```

### Attributes

XAxis

Yaxis

## BubbleColorTable

```
BubbleColorTable[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

`BubbleColorTable` defines a set of colors for dataset N that overrides all other color specifications for that set. The parameters used for specifying the color of bubbles in a chart are (in ascending order of precedence) `ColorTable`, `DataSets`, `BubbleSymbol`, `BubbleFillPattern` and `BubbleColorTable`. `BarColorTable` is used most frequently to color some specific bar.

For example

```
BubbleColorTable2 = , ,blue;
```

will change the third bubble in the second series to blue, while all other bubbles in the chart continue to be colored by one of the other color related parameters.

The colors you can use are defined in the common `Color` attribute (Chapter 4) .

### Used in These Charts

Bubble

### Example:

```
BubbleColorTable2 = , , red;
```

### Attributes

None

## BubbleFillPattern

```
BubbleFillPattern[N] = (type, color1, color2, imageURL), ...;
```

The `BubbleFillPattern` parameter provides a visual pattern fill inside the bubble area of a chart.

Type	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient



Images		
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

Bubble

#### Example:

```
BubbleFillPattern = (GRADIENTFDIAG,blue,white);
```

#### Attributes

Type	Color1	Color2
ImageFormat	ImageURL	

## BubbleScale

```
BubbleScale[N] = (MinValue, MaxValue, AREA|DIAMETER, PointColor), ...;
```

BubbleScale controls the bubble scale to be displayed for the named bubble sets.

### Used in These Charts

Bubble

#### Example:

```
BubbleScale = (10,100,DIAMETER,white), (100,500,AREA,green);
```

#### Attributes

MinValue	MaxValue	AREA or DIAMETER	PointColor
----------	----------	------------------	------------

#### MinValue

MinValue specifies the minimum Z value that will result in the display of a bubble. Data points with Z values smaller than minValue will not be drawn.

#### Example:

```
BubbleScale = (10,100,DIAMETER,white), (100,500,AREA,green);
```

**Values**

Any real number  $\geq 0$ .

**Default**

None

**MaxValue**

---

`MaxValue` is the maximum value of `z` in the `BubbleSet`, which is displayed with a symbol the size of `MaxSize`. The sizes of bubbles with other `Z` values will be drawn proportionally.

**Example:**

```
BubbleScale = (10, 100, DIAMETER, white), (100, 500, AREA, green);
```

**Values**

Any real number  $\geq 0$ .

**Default**

None

**Area or Diameter**

---

This switch specifies how the relative sizes of bubbles with a `z` value  $< \text{maxValue}$  are determined. For `AREA`, the area of the bubbles is proportional to the `Z` value defined. For `DIAMETER`, the diameter of the bubbles is proportional to the `Z` value defined.

**Example:**

```
BubbleScale = (10, 100, DIAMETER, white), (100, 500, AREA, green);
```

**Values**

One of `AREA` or `DIAMETER`

**Default**

None

**PointColor**

---

If `PointColor` is specified, a dot is drawn in the center of the bubble in the color specified. Otherwise, this attribute behaves as does the common `Color` attribute.

**Example:**

```
BubbleScale = (10, 100, DIAMETER, white), (100, 500, AREA, green);
```

### Values

See the common `Color` attribute in **Chapter 4** for details.

### Default

blue

## BubbleSets

---

```
BubbleSets[N] = ("Name1", Color1), ("Name2", Color2), ...;
```

`BubbleSets` defines the data sets for a bubble graph; these data sets are (X,Y,Z) that can be rendered as a series of bubbles— or other symbols, such as squares or diamond—connected by optional lines. The bubbles are graphed along defined X and Y axes, with a third coordinate, Z, that determines the relative size of the bubble.

### Used in These Charts

Bubble

### Example:

```
BubbleSets = ("Server #1",x0572c6),("Server #2",xE32F41);
```

### Attributes

Name            *Color*

### Name

---

`Name` is the value assigned to a data point from the `BubbleSets` parameter, and represents an (X,Y,Z) numeric vector.

### Example:

```
BubbleSets = ("Server #1",x0572c6),("Server #2",xE32F41);
```

### Values

Any string value

### Default

None

## BubbleSet[n]

---

```
BubbleSetn = (x,y,z), (x,y,z), ...;
```

Defines a vector of (x,y,z) values for the named bubble set defined by the `BubbleSets` parameter. The z value defines the relative size of the bubble within the `BubbleSet`. If `NULL` is substituted for any part of the vector, the bubble will not be drawn. A value of z equal to zero results in having no bubble drawn, as well.

*Used in These Charts*

Bubble

*Example:*

```
BubbleSet1 = (1,27,10), (5,50,20), (10,100,30), (23,125,40), (56,170,50),
              (65,220,60), (68,280,70);
BubbleSet2 = (3,27,100), (5,40,200), (8,125,300), (26,137,400), (75,260,500);
```

*Attributes*

None

**BubbleSymbol**


---

```
BubbleSymbol[N] = (SymType, MaxSize, Style, BorderColor, BorderWidth,
                  SymbolColor,ShadowWidth), ...;
```

BubbleSymbol controls the display of the bubbles or symbols for the named bubble sets. You should specify as many groups, or “tuples,” as there are bubble sets.

*Used in These Charts*

Bubble

*Example:*

```
BubbleSymbol = (CIRCLE,80,FILLED,null,null), (SQUARE,60,OUTLINED,xe3e3e3,1);
```

*Attributes*

<i>SymType</i>	<i>MaxSize</i>	<i>Style</i>	<i>BorderColor</i>	<i>BorderWidth</i>
<i>ShadowThickness</i>	<i>SymbolColor</i>			

**SymbolColor**


---

SymbolColor specifies for foreground color of the LineSymbol.

**ShadowThickness**


---

ShadowThickness specifies for size of the shadow behind a BubbleSymbol. Any number other than 0 causes NetCharts to choose a shadow size based on the size of the symbol. Specifying a value of 0 suppresses the shadow.

**SymType**


---

SymType indicates the type of symbol displayed for these bubbles.

*Example:*

```
BubbleSymbol = (CIRCLE,80,FILLED,null,null), (SQUARE,60,OUTLINED,xe3e3e3,1);
```

**Values**

CIRCLE	Displays circles
SQUARE	Displays squares
DIAMOND	Displays diamonds
CROSS	Displays crosses
TARGET	Displays targets (bulls-eye)
TRIANGLEDOWN	Displays downward pointing triangles
TRIANGLEUP	Displays upward pointing triangles

**Default**

CIRCLE

**MaxSize**

MaxSize indicates the maximum size of the bubble symbol, in pixels.

**Example:**

```
BubbleSymbol = (CIRCLE, 80, FILLED, null, null), (SQUARE, 60, OUTLINED, xe3e3e3, 1);
```

**Values**

Any whole pixel value.

**Default**

None

**Style**

Style controls how the bubble symbol should be drawn.

**Example:**

```
BubbleSymbol = (CIRCLE, 80, FILLED, null, null), (SQUARE, 60, OUTLINED, xe3e3e3, 1);
```

**Values**

FILLED	symbol is filled with the bubble set color
OUTLINED	only the outline is drawn, using the bubble set color
BOTH	symbol is filled with the bubble set color and the outline is drawn using the BorderColor.

**Default**

OUTLINED

**BuildAnimationEnabled**

```
BuildAnimationEnabled = ON | OFF;
```

`BuildAnimationEnabled` enables or disables all animation on charts that are delivered in SVG format.

### *Used in These Charts*

ALL

### *Example:*

```
BuildAnimationEnabled = OFF;
```

### *Attributes*

*Mode*

---

## CellTextAutoColorThreshold

---

`CellTextAutoColorThreshold` = *range*

Color distance threshold (between the grid foreground and grid background colors) that determines whether a foreground text color swap is necessary for visibility.

### *Used in These Charts*

Heat Map

### *Example:*

```
CellTextAutoColorThreshold = 20;
```

### *Attributes*

*Range*

### *Values*

*0-100 Percent*

---

## CenterRadius

---

`CenterRadius[N]` = (*radius*);

The `CenterRadius` parameter defines the diameter of the circle at the origin of the chart.

### *Used in These Charts*

Radar

### *Example:*

```
CenterRadius = 15;
```

### *Attributes*

*Radius*

---

## ChartElementSpacing

---

```
ChartElementSpacing = spacing;
```

`ChartElementSpacing` defines the space between the elements of a chart. Chart elements include Titles, Legends, Axes and the PlotArea

`spacing` - the size in pixels of the horizontal and vertical space between chart elements.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
ChartElementSpacing = 3;
```

### *Attributes*

*Spacing*

---

## Charts

---

```
Charts = (Name1, Type1, Width1, Height1), (Name2, Type2, Width2, Height2), ...;
```

The `Charts` parameter is the main support for the display of multiple charts within a Multi-chart. It defines the name and type for each chart to be displayed. For each chart defined in this parameter, the corresponding `ChartURL` and `ChartScript` parameters will be processed to define the chart itself.

If a chart name or type of `BREAK` is specified, then the subsequent charts will be displayed on a new row or column. In this way, multiple rows or columns with a different number of charts may be displayed. If the optional `Width` and `Height` attributes are defined, they are used to determine the percentage of space that should be allocated to each chart.

**IMPORTANT NOTE:** Because the Multi-chart allows the definition of multiple `ChartScripts`, the standard `NFPParamScript` parameter is not supported. Consequently, all parameters must be defined using the standard `<param>` tag within the applet parameter, as shown in the examples below.

### *Used in These Charts*

Multi-chart

### *Example:*

```
<PARAM NAME=Charts VALUE='
  ("Piechart1",PIECHART),
  ("Piechart2",PIECHART),
  ("Piechart3",PIECHART); '>
```

```

<PARAM NAME=Charts VALUE='
  ("Bar1", BARCHART),
  ("Bar2", BARCHART),
  (BREAK),
  ("Bar3", BARCHART),
  ("Bar4", BARCHART) '>

<PARAM NAME=Layout VALUE="(ROWS)">      <!-- charts laid out in rows -->
<PARAM NAME=Charts VALUE='
  ("Sales", PIECHART),
  ("Expense", BARCHART),
  ("Bonus", XYCHART),
  (BREAK),
  ("Growth", COMBOCHART),
  (BREAK),
  ("Budget", XYCHART),
  ("Salary", PIECHART);
'>

```

**Attributes**

Name	Type	Width	Height
------	------	-------	--------

**Type**

Type specifies the type of chart shown as one element of the multi-chart.

**Example:**

```

<PARAM NAME=Charts VALUE='
  ("Piechart1", PIECHART),
  ("Linechart", LINECHART),
  ("Piechart2", PIECHART); '>

```

**Values**

BARCHART  
 BOXCHART  
 COMBOCHART  
 DIAGRAM  
 DIALCHART  
 LINECHART  
 PARETOCHART  
 PIECHART  
 STOCKCHART  
 STRIPCHART  
 TIMECHART  
 XYCHART

**Default**

None. You must use one of the values above.

**Width, Height**

The `Width` and `Height` attributes are optional, and specify the width and height of the component charts. The `Width` and `Height` attributes are interpreted as "relative" sizes, depending on the total number of charts in each row or column. If no `Width` or `Height` attributes are specified, then a default value of 1 is



used. It is easiest to specify the width and height as percentages in the range 1 to 100, although that is not required.

Only the first chart in each row need have the width and height specified, since those values will be used as the defaults for the subsequent charts. To get uniform widths in each row, you can specify 1 for the width of each chart.

**Example:**

```
<PARAM NAME=Layout VALUE="(ROWS)">      <!-- charts laid out in rows -->
<PARAM NAME=Charts VALUE='
    ("Sales", PIECHART, 1, 25),          <!-- gets 25% of chart height -->
    ("Expense", BARChart),
    ("Bonus", XYCHART),
    (BREAK),
    ("Growth", COMBOCHART, 1, 50),      <!-- gets 50% of chart height -->
    (BREAK),
    ("Budget", XYCHART, 1, 25),        <!-- gets 25% of chart height -->
    ("Salary", PIECHART);
'>
```

**Values**

1 to 100    Relative sizes, in percentages

**Default**

1            Makes all widths and heights uniform

## ChartHeight

---

ChartHeight = height;

The ChartHeight parameter allows a chart writer to specify the height of a chart.

**Used in These Charts**

All

**Example:**

```
ChartHeight = 200;
```

**Attributes**

Height

## ChartName

---

ChartName = name;

The ChartName parameter allows a chart writer to specify the name of a chart.

**Used in These Charts**

All

**Example:**

```
ChartName = "Chart XYZ";
```

**Attributes**

Name

**ChartScript[n]**

```
ChartScript[1-20] = "parameter definition script";
```

The `ChartScript` parameter takes the place of the standard `NFPParamScript` parameter for Multi-charts. It is similar, though, in that it allows a developer to specify any number of parameters for the chart.

In the Multi-chart, each `ChartScript` parameter defines the parameters for the chart types defined in the `Charts` parameter. Accordingly, care should be taken to ensure that the appropriate types of parameters are defined for a given chart type. Use the format in the example, below.

**Used in These Charts**

Multi-chart

**Example:**

```
<PARAM NAME=Charts VALUE='("Bar1",BARCHART), ("Bar2",BARCHART),
(BREAK),
("Bar3",BARCHART), ("Bar4",BARCHART) '>
<PARAM NAME=Layout VALUE="(ROWS)">
<PARAM NAME = ChartScript1 VALUE='
Background = (white, NONE);
Header = ("1. Most Requested Pages", black, Helvetica, 12);
DwellLabel = ("", black, "Helvetica", 9);
DwellBox = (xe3e3e3, SHADOW, 2);
ColorTable = xB5D5F0, xBEA9AD, xDACE98, xEBF0F3, xAABAC5, xBFC1A0;
BottomTics = ("ON", black, "Helvetica", 9);
LeftTics = ("ON", black, "Helvetica", 9);
LeftScale = (0, 2700);
LeftFormat = (INTEGER);
BarLabels = "Home\nPage", "NetCharts", "Examples", "Products";
GraphType = GROUP;
DataSets = ("Server1", NULL);
DataSet1 = 2694, 780, 628, 513;
Bar3DDepth = 3;
'>
```

**Attributes**

Insert the appropriate chart parameters within `ChartScript`.

**ChartType**

```
ChartType = type;
```

The `ChartType` parameter allows a chart writer to specify the type of chart to be used.

**Used in These Charts**

All

**Example:**

```
ChartType = barchart;
```

**Attributes**

Type

---

**ChartURL[n]**

---

```
ChartURL[1-20] = "URL";
```

ChartURL specifies the location of the chart definition file to be used to define the corresponding chart. The URL may use either the HTTP or FILE protocol, depending on the browser environment in which the applet is running. In all cases, if a relative URL is given, then the document base of the applet will be used to locate the relative file.

The ChartURL parameter may be used *in addition to* or *instead of* the ChartScript parameter. If both are specified, then the ChartURL parameter is processed first, allowing developers to specify default parameters using the URL file, which can be overridden by the parameters defined in the ChartScript parameter.

**Used in These Charts**

Multi-chart

**Example:**

```
ChartURL1 = "chart1.cdl";
```

**Attributes**

URL

---

**ChartWidth**

---

```
ChartWidth = width;
```

The ChartWidth parameter allows a chart writer to specify the width of a chart.

**Used in These Charts**

All

**Example:**

```
ChartWidth = 250;
```

**Attributes**

Width

---

**Color**

---

```
TopColor[N] = Color;
```

```
BottomColor[N] = Color;  
LeftColor[N] = Color;  
RightColor[N] = Color;
```

The `Color` parameter controls the color of the given axis and the tic marks, but not the tic mark labels. The default axis color is black. If the `NULL` color is specified, the axis color is not changed by this parameter.

**Example:**

```
BottomColor = xB5D5F0;
```

**Attributes**

`Color`

---

## ColorTable

---

```
ColorTable = Color1, Color2, Color3, Color4, Color5, ...;
```

`ColorTable` is a powerful way to control the appearance of charts and impose both uniformity and color harmony upon them. `ColorTable` supersedes whatever system color table is in use. The colors in a color table will repeat in sequence whenever the number of data sets exceeds the number of colors defined in the `ColorTable` parameter.

The colors you can use are defined in the common `Color` attribute (Chapter 4).

**Used in These Charts**

All

**Example:**

```
ColorTable = xB5D5F0, xBEA9AD, xDACE98, xEBF0F3, xAABAC5, xBFC1A0, xF2E0D4;
```

**Attributes**

None

---

## CumulativeLineSetName

---

```
CumulativeLinesetName = name;
```

Name assigned to the cumulative line set. Used in the legend if the Legend CDL parameter does not define a label for the cumulative line.

**Used in These Charts**

Pareto

**Example:**

```
CumulativeLineSetName = "Cumulative Percentage Line"
```

## CumulativeLineStyle

---

CumulativeLineStyle = (Type, LineWidth, Color, FillColor, LineType);

This parameter specifies the line style to be displayed for the cumulative percentage line.

### Used in These Charts

Pareto

### Example:

```
CumulativeLineStyle = (SOLID, 3, blue, blue, NORMAL);
```

### Attributes

Color    FillColor    LineType    LineWidth    Type

### FillColor

---

If this attribute is not NULL, then the area under the cumulative line will be filled with the given color.

### Example:

```
CumulativeLineStyle = (SOLID, 3, red, pink, NORMAL);
                        <!-- red line with pink fill -->
```

### Values

NULL                    Also, value left unspecified: No color fills the area under the line.  
Any legal color        Area under the line is filled. See Chapter 4 for the Color attribute.

### Default

None

### Type

---

The style of line to draw

### Values

NONE  
SOLID  
DOTTED  
DASHED  
DOTDASH

### Default

SOLID

### LineType

---

The type of line to use to connect the points in the line set.

**Values**

NORMAL  
FIT  
BOTH

**Default**

NORMAL

**CumulativeLineStyle**


---

```
CumulativeLineStyle = (Type, Size, Style, BorderColor, BorderWidth, ImageURL,
SymbolColor,ShadowWidth) ;
```

`CumulativeLineStyle` specifies the symbols to be displayed for the cumulative line.

**Used in These Charts**

Pareto

**Example:**

```
CumulativeLineStyle = (CIRCLE, 6, BOTH, white, 1, grey, 0);
```

**Attributes**

<i>BorderColor</i>	<i>BorderWidth</i>	<i>ImageURL</i>	<i>ShadowWidth</i>
Size	Style	SymbolColor	Type

**Size**


---

`Size` specifies the size of the symbol in pixels. This attribute is ignored for `IMAGE` symbols.

**Example:**

```
CumulativeLineStyle = (CIRCLE, 6, BOTH, white, 1);
```

**Values**

Any integer value in pixels

**Default**

None

**SymbolColor**


---

`SymbolColor` specifies for foreground color of the `CumulativeLineStyle`.

**ShadowThickness**


---

`ShadowThickness` specifies for size of the shadow behind a `CumulativeLineStyle`. Any number other than 0 causes NetCharts to choose a shadow size based on the size of the symbol. Specifying a value of 0 suppresses the shadow.

---

## Style

---

Style specifies how the `LineStyle` should be drawn, including `FILLED`, `OUTLINED`, or `BOTH`. If `FILLED` is specified, the symbol is filled with the line set color. If `OUTLINED` is specified, only the outline is drawn, using the line set color. If `BOTH` is specified, then the symbol is filled with the line set color and the outline is drawn using the `borderColor`.

### Example:

```
CumulativeLineStyle = (CIRCLE, 6, BOTH, white, red);
```

### Values

<code>FILLED</code>	Symbol is filled with the cumulative line color.
<code>OUTLINED</code>	Only the outline is drawn, using the cumulative line color.
<code>BOTH</code>	Symbol is filled with the cumulative line color and the outline is drawn using the <code>borderColor</code>

### Default

None

### Type

---

Type specifies the type of symbol to be displayed on the line set.

### Example:

```
CumulativeLineStyle = (SQUARE, 6, BOTH, cyan, 1);
```

### Values

<code>NONE</code>	No symbol is displayed.
<code>CIRCLE</code>	Displays circles
<code>SQUARE</code>	Displays squares
<code>DIAMOND</code>	Displays diamonds
<code>CROSS</code>	Displays crosses
<code>TARGET</code>	Displays targets (bulls-eye)
<code>TRIANGLEDOWN</code>	Displays downward pointing triangles
<code>TRIANGLEUP</code>	Displays upward pointing triangles
<code>IMAGE</code>	If specified, the <code>ImageURL</code> attribute is required and will be used to load a GIF image for the symbol.

### Default

None

---

## CumulativeLineValueLabel

---

```
CumulativeLineValueLabel = (mode, color, font name, width);
```

Defines the label value to be displayed for each point in the cumulative line.

***Used in These Charts***

Pareto

***Example:***

```
CumulativeLineValueLabel = ("ON", black, "Helvetica", 18);
```

***Attributes***

Mode Color Font Name Width

---

**CumulativeLineValueLabelBox**

---

```
CumulativeLineValueLabelBox = (color, mode, depth);
```

Defines the line label box to be displayed with each point in the cumulative line.

***Used in These Charts***

Pareto

***Example:***

```
CumulativeLineValueLabelBox = (grey, RAISED, 3);
```

***Attributes***

Color Mode Depth

---

**CumulativeLineValueLabelStyle**

---

```
CumulativeLineValueLabelStyle = labelposition;
```

Defines where the line value label text will display for each point in the cumulative line.

***Used in These Charts***

Pareto

***Example:***

```
CumulativeLineValueLabelStyle = TOP;
```



**Values**

TOPLEFT	Displayed at the top left point
TOP	Displayed at the top of the point
TOPRIGHT	Displayed at the top right point
LEFT	Displayed at the left of the point
CENTER	Displayed at the center point
RIGHT	Displayed at the right of the point
BOTTOMLEFT	Displayed at the bottom left point
BOTTOM	Displayed at the bottom of the point
BOTTOMRIGHT	Displayed at the bottom right point

**Default**

TOP

**Attributes**

Label Position

**DataAxis**


---

```
DataAxis[N] = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
```

DataAxis defines the X and Y axes for associated data sets in charts using bars.

**Used in These Charts**

Bar, Box, Combo, Pareto, Stock, Strip, Time

**Example:**

```
DataAxis = (BOTTOM, LEFT), (BOTTOM, RIGHT);
```

**Attributes**

XAxis                      YAxis

**DataLegend**


---

```
DataLegend = ON|OFF;
```

DataLegend enables the display of a Data Legend on a Bar, Line or Combo chart. A Data Legend will present the chart data values in a table that is combined with the Bottom or Left Axis tic labels. The position of the DataLegend depends on the GraphType Parameter. If GraphType=VERTICAL the Data Legend will appear below the bottom axis. If GraphType=HORIZONTAL the Data Legend will appear to the left of the left axis.

**Used in These Charts**

Bar, Combo, Line

---

## DataLegendGrid

---

```
DataLegendGrid = (LineColor, bgColor, borderColor, bgImage, ImageFormat);
```

This parameter specifies the border and background colors for the Data Legend.

### *Used in These Charts*

Bar, Combo, Line

### *Example:*

```
DataLegendGrid = (green, white, black, null, CENTER);
```

### *Attributes*

LineColor	BackgroundColor	BorderColor	BackgroundImage
ImageFormat			

---

### *LineColor*

`LineColor` specifies the color of the grid lines in the Data Legend. See the common `Color` attribute in **Chapter 4** for details.

---

### *BackgroundColor*

`BackgroundColor` specifies the color for the Data Legend's background. See the common `Color` attribute for details.

---

### *BackgroundImage*

`BackgroundImage` specifies an image file for the Data Legend's background. See the common `Image` attribute for details.

---

## DataLegendGridLine

---

```
DataLegendGridLine = (LineType, LineStyle, LineWidth);
```

Defines the line properties for the Data Legend defined in the `DataLegendGrid` parameter, above.

### *Used in These Charts*

Bar, Combo, Line

### *Example:*

```
DataLegendGridLine = (HORIZONTAL, DOTTED, 2);
```

### *Attributes*

LineType	LineStyle	LineWidth
----------	-----------	-----------

---

## *LineType*

---

Tells where to draw the lines specified with a `DateLegendGrid` parameter.

### *Values*

BOTH	draw both horizontal and vertical lines (default)
VERTICAL	draw vertical lines only
HORIZONTAL	draw horizontal lines only
NONE	draw no grid lines

### *Default*

BOTH

---

## *LineStyle*

---

`LineStyle` tells how to draw the grid lines in a Data Legend as defined in a `DateLegendGrid` parameter.

### *Values*

SOLID	solid lines (default)
DOTTED	dotted lines
DASHED	dashed lines
DOTDASH	dot-dash lines

### *Default*

SOLID

---

## **DataPointActiveLabels(n)**

---

```
DataPointActiveLabels(n) = ("Label1","URL1","Target1"),...;
```

`DataPointActiveLabels` define the active labels associated with raw data points.

### *Used in These Charts*

Box Chart

### *Example:*

```
DataPointActiveLabels = (lightgray, SHADOW, 3,,,gray);
```

### *Default*

ON

### *Attributes*

*Color*

## DataPointColor

---

`DataPointColor = Color;`

DataPointColor allows users to specify the color to be used to display the raw data points. This value is used if no color is specified in the DataPointSymbol parameter.

### Used in These Charts

Box Chart

### Example:

```
DataPointColor = red;
```

### Attributes

Color

## DataPointJitter

---

`DataPointJitter = ON | OFF;`

DataPointJitter is used to increase the visibility of individual points when displaying raw data. The jitter option adds random horizontal jitter to the x values of each data point, allowing multiple points with the same Y value to be distinguishable.

### Used in These Charts

Box Chart

### Example:

```
DataPointJitter = ON;
DataPointJitter = OFF;
```

### Default

ON

### Attributes

Mode

## DataPointSymbol

---

`DataPointSymbol = (type1, size1, style1, bordercolor1, borderwidth1,image1,color1),...;`

DataPointSymbol is used to define the style in which to draw data points when displaying raw data.

typeN	the type of symbol to use for points in data series N. Legal values are NONE, CIRCLE, SQUARE, DIAMOND, CROSS, TARGET, TRIANGLEDOWN, TRIANGLEUP, IMAGE
-------	--

sizeN	the size in pixels of the symbols for points in data series N
styleN	the drawing style for points in data series N. Legal values are FILLED, OUTLINED or BOTH
borderColorN	the color of the border for points in data series N
borderWidthN	the width in pixels of the border for points in data series N
imageN	the image to use for displaying for points in data series N
colorN	the color for points in data series N

**Used in These Charts**

Box Chart

**Example:**

```
DataPointSymbol = (CIRCLE,3,FILLED,red,2,,green);
```

**Attributes**

*BorderColor, BorderWidth, Color, Image, Type, Size, Style*

**DataSet[n]**

```
DataSet[1-50] = a, b, c, ...;
```

`DataSetn` defines a list of numeric data values for each data set defined by the `DataSets` parameter in a charts having bars, such as bar, box, combo, pareto, strip, and time charts. Each data set may contain a different number of values. If the value, `NULL`, is substituted for a number, nothing will be drawn in that bar position.

**Used in These Charts**

Bar, Box, Combo, Pareto, Stock, Strip, Time

**Example:**

```
DataSets      = ("Hamburgers", x33ffcc), ("Donuts", x9999ff), ("Cheez Doodles",
                xff6600), ("Exercise", xffcc33);
DataSet1     = 50, 12, 32, 52, 65, 40, 87;
DataSet2     = 40, 30, 77, 10, 25, 83, 9;
DataSet3     = 45, 50, 89, 33, 99, 44, 31;
DataSet4     = null, -4, -7, -12, -16, -23, -26;
```

**Attributes**

No attributes, as such, are used. Actual data appears in this parameter.

## DataSet[n]P[m]

---

`DataSet[1-50]P[1-50] = a, b, c, ...;`

`DataSetn` defines a list of numeric values for each dataset in a grouped, stacked BarChart (`GraphType=GROUPSTACK`). In a grouped stacked BarChart each tic location can display multiple data sets, and each data set can contain multiple data values.

Consider this example:

```

DataSets = ("Set1"), ("Set2");
DataSet1P1 = 0.6, 0.7;
DataSet1P2 = 2.0, 1.1;
DataSet1P3 = 1.5, 2.0;
DataSet2P1 = 0.7, 0.9;
DataSet2P2 = 1.3, 2.1;
DataSet2P3 = 2.1, 1.4;

```

This chart contains two data sets; each set has 3 values to display at each tic on the chart.

`DataSet1P1` defines two values - the first value in the first set at each tic mark. `DataSet2P3` defines two values - the third value of the second set at each tic.

## DataSets

---

`DataSets[N] = (DataSetName1, BorderType1, BorderWidth1, "ImageURL", ImageFormat1, BorderColor1), ...;`

`DataSets` defines a list of data sets with the given name color and type for charts with bars in them, such as bar, box, combo, pareto, strip, and time charts. The `Name` attributes will be used as items in the legend; and the `Color` attribute will be used for each bar in the data set. The optional `Type` attribute indicates the shape of bar to use. `DataSets` must be paired with the corresponding `DataSetn` parameter(s); you may specify from 1 up to 50 data sets.

### Used in These Charts

Bar, Box, Combo, Pareto, Stock, Strip, Time

### Example:

```

DataSets      = ("Server #1",,,,,), ("Server #2",,,,,), ("Server #3",,,,,);
DataSet1     = 100, 125, 245.78, 147, 67;
DataSet2     = 85, 156, 179.5, 211, 123;
DataSet3     = 97, 87, 56, 267, 157;

```

### Attributes

<i>Label</i>	<i>Color</i>	<i>Type</i>
--------------	--------------	-------------

### Type

---

`Type` indicates, in context of the `DataSets` parameter, the kind of bar that you see. Some `Type` values may produce unusual results if the `GraphType` is `STACK` or `ROWS`. If one of the pie types is selected, the alternate pie color will be extracted from the first element of a color table.

**Example:**

```
DataSets      = ("Potatoes", xCC9933, DIAMONDBAR), ("Green Beans", darkgreen,
BAR), ("Tomatoes", xCC0033, CYLINDER), ("Corn", wheat, TRIANGLEBAR);
```

**Values**

**BAR** A standard rectangular 3-D bar is shown

**CYLINDER** The bars are 3-D, and the cross-section of the bars is circular

**DIAMONDBAR** The bars are 3-D, and the cross-section of the bars is a diamond shape

**PIEHORIZONTAL** The bars are a horizontal pie shape

**PIEVERTICAL** The bars are a vertical pie shape

**TRIANGLEBAR** The bars are 3-D, and the cross-section of the bars is triangular.

**Default**

**BAR**

**Data Type**


---

```
Data Type[N] = Type;
```

`DataSets` defines the type of data in the box chart's data sets. This helps the chart to process the data.

**Used in These Charts**

Box

**Example:**

```
Data Type      = RAW;
Data Type      = SUMMARY;
```

**Attributes**

Type

**Type**


---

Type indicates whether the data for the box chart has been statistically processed or not.

**Example:**

```
Data Type      = RAW;

Data Type      = SUMMARY;
Data Sets      = ("Sub-Compact", x00ab9c, BOX, 1),
                ("Compact", xf0887f),
                ("MiniVan", xf7bb83),
                ("Truck", x3fbae3),
                ("Luxury\nSedan", xf189af);

DataSet1       = 9.4, 10.2, 11.2, 7.5, 12.7, 22, 31;
DataSet2       = 10.4, 10.6, 10.8, 7.8, 13.5, 19;
DataSet3       = 17.6, 19.5, 24.0, 15.3, 30.5;
DataSet4       = 10.4, 11.0, 12.0, 7.8, 12.5;
DataSet5       = 25.6, 28.5, 33.0, 20.3, 35.5, 6;
```

**Values**

RAW	Data is raw statistical data. Percentiles and other ancillary information will be computed based on these values.
SUMMARY	Percentile information has already been computed. When a SUMMARY is used, the data is formatted as follows: Value 1     25 <sup>th</sup> percentile Value 2     50 <sup>th</sup> percentile Value 3     75 <sup>th</sup> percentile Value 4     Smallest non-outlier value Value 5     Largest non-outlier value Value 6 +   Outliers, if any

**Default**

None

**DebugClear**


---

```
DebugClear = debugFilter;
```

The `DebugClear` parameter clears the user selected debug information on the Java Console in applets mode.

**Used in These Charts**

All

**Example:**

```
DebugClear = ALL;
```

**Attributes***debugFilter***DebugSet**


---

```
DebugSet = debugFilter;
```

The `DebugSet` parameter allows users to generate debug information on the Java Console in applets mode for the following operations.

AGENT	Agent status and errors.
ALL	No filtering, ALL debug information.
AXIS	Axis generation and rendering information.
BEANS	Loadable data object information.
CACHE	Internal memory cache status and error messages.
DWELL	Any issues relating to dwell (popup/active) labels.



FILE	File I/O and interpretation
GRAPH	Graph rendering status and errors
HTTP	HTTP Request/Response headers and status codes.
IMAGE	Image loading errors (e.g. attempting to load a background.png image for a chart.)
JDBC	Java DataBase Connectivity (JDBC) information including driver, connections, SQL, and result sets.
LEGEND	Legend generation and rendering.
LICENSE	License location and processing.
NOTES	Annotation generation and rendering information.
PARAM	Parameter parsing errors
REMOTE	Reports Named Data Set (NDS) processing information and errors.
SECURITY	User authentication information.
SERVER	Server specific issues, port requests, ACL, event scheduling, connection limits, and more.
SYMBOL	Flags problems related to drawing symbols.
THREAD	Threading status and error messages.

The example below shows how to generate debug information on the Java Console for FILE operations, using the following syntax:

```
DebugSet = FILE;
DebugClear = FILE;
```

The DebugSet command enables the printing of debug messages for all subsequent FILE keywords, including the processing of all FileFormat Parameters. The debug messages will continue to be printed until the end of the Parameter script or the DebugClear directive is seen.

For example, the following script defines a simple piechart, enabling debug messages for all FILE statements:

```
DebugSet = FILE;
FileFormat = ("", "|", "\n", null, null, "/");
Slices = FILE "pietest.dat";
```

Resulting in the following debug output on the Java Console:

```
NFFile: startDelim = <>
NFFile: itemDelim = <|>
NFFile: endDelim = <\n>
NFFile: comments = </>
NFFile: ++++++
NFFile: Parameter = Slices
NFFile: Filename = <pietest.dat>
NFFile: Opening http://mycompany.com/reports/pietest.dat
NFFile: Item(1): 10|blue|Fred Smith NFFile: Item(2): 20|red|Sally Jane
NFFile: Item(3): 30|green|Oscar Jones
NFFile: Processed 3 Item(s)
NFFile: -----
```

### *Used in These Charts*

All

**Example:**

```
DebugSet = FILE;
```

**Attributes**

*debugFilter*

**DialActiveLabels**

```
DialActiveLabels[N] = (Name, Label, URL, Target), ...;
```

`DialActiveLabels` defines a list of active label destinations for a dial in a dial chart.

**Used in These Charts**

Dial

**Example:**

```
DialActiveLabels = ("Destination", "demo.html", "frame1");
```

**Attributes**

*Label*                                      *URL*                                      *Target*

**DialBorders**

```
DialBorders[N] = (Name, Type, Thickness, Color, NONE|CENTER|ENDTOEND), ...;
```

`DialBorders` controls the decorative line border around a dial in a dial chart. Its attributes are standard line attributes, except for the switch.

**Used in These Charts**

Dial

**Example:**

```
DialBorders = ("Hour Dial", SOLID, 2, lightgray, CENTER);
```

**Attributes**

*Name*                                      *LineType*                                      *LineWidth*                                      *Color*                                      (Switch)

**Switch**

This switch controls the dial border's polygon behavior.

**Example:**

```
DialBorders = ("Hour Dial", SOLID, 2, lightgray, CENTER);
```

**Values**

NONE	The border doesn't connect its ends
CENTER	Includes the (circular) center of the dial in the borders
ENDTOEND	Only includes the points in the border.

**Default**

No defaults

**DialClip**

```
DialClip = clipType;
```

DialClip specifies quadrant(s) of the Dial Chart to be displayed using the entire graph space. For instance, specifying TOP will cause the dial to use the entire graph space to display the top-half of the dial. Specifying BOTTOMRIGHT will cause the dial to use the entire graph space to display only the bottom-right quadrant of the dial. The default is NONE which causes the entire dial to be displayed.

**Used in These Charts**

Dial

**Example:**

```
DialClip = TOP;
```

DialClip values can be:

TOPLEFT	TOP	TOPRIGHT
LEFT	NONE	RIGHT
BOTTOMLEFT	BOTTOM	BOTTOMRIGHT

**DialClipPad**

```
DialClipPad = N;
```

DialClipPad specifies the margin to be used in conjunction with DialClip.

**Used in These Charts**

Dial

**Example:**

```
DialClipPad = 10;
```

**DialDelete**

```
DialDelete = (Name|ALL), ...;
```

DialDelete is used to delete a specific dial, or all dials, in a dial chart.

*Used in These Charts*

Dial

*Example:*

```
DialDelete = ("Hour Dial"), ("Minute Dial");
DialDelete = ALL;
```

*Attributes*

(Switch)

*Switch*

This switch allows you to either name the dials that are to be deleted, or to delete all at once.

*Example:*

```
DialDelete = ("Hour Dial"), ("Minute Dial");
DialDelete = ALL;
```

*Values*

Name            A string that names a dial  
ALL             All the dials

*Default*

No defaults

**DialFillPattern**

```
DialFillPattern = (name, type, color1, color2, imageURL), ...;
```

The `DialFillPattern` parameter provides a visual pattern fill inside the dial of a dial chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBTDIAG	top left to bottom right style gradient
	GRADIENTRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient

Images		
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

**Used in These Charts**

Dial

**Example:**

```
DialFillPattern = ("Name", GRADIENTHORIZONTAL, blue, white);
```

**Attributes**

Name	Type	Color1	Color2
ImageFormat	ImageURL		

**DialFills**

```
DialFills = (Name, Color, NONE|CENTER|ENDTOEND), ...;
```

DialFills controls the decorative fill inside a dial in a dial chart. Its attributes are standard area attributes, except for the switch.

**Used in These Charts**

Dial

**Example:**

```
DialFills = ("Hour Dial", xebf0f3, CENTER);
```

**Attributes**

Name	Color	(Switch)

**Switch**

This switch controls the dial border's polygon behavior.

**Example:**

```
DialFills = ("Hour Dial", xebf0f3, CENTER);
```

**Values**

NONE	The border doesn't connect its ends
CENTER	Includes the (circular) center of the dial in the borders
ENDTOEND	Only includes the points in the border.

**Default**

No defaults

**DialFormats**


---

```
DialFormats = (Name, FLOAT|INTEGER|DECIMAL, formatExpression), ...;
```

`DialFormats` allow for the formatting of dial labels.

**Used in These Charts**

Dial

**Example:**

```
DialFormats = ("Hour Dial", FLOAT, %.2f);
```

**Attributes**

<i>Name</i>	(Switch)	<i>formatExpression</i>
-------------	----------	-------------------------

**Switch**


---

This switch controls the dial format type. If the format type is `INTEGER` or `FLOAT`, the input data value is expected to be of type integer or float and will be parsed as such (if string conversion is necessary). The format itself is a C-language style `sprintf` format. Some examples:

Data	Type	Format	Output
1000	INTEGER	%d	1000
1000	INTEGER	%, %d	\$1,000
1000	INTEGER	%d%	1000%
1000	FLOAT	%f	1000.0
1000	FLOAT	%.2f	1000.00
1000	FLOAT	%, .2f	\$1,000.00

If the format type is `DECIMAL`, the format syntax is consistent with those defined in the Java `DecimalFormat` spec.

**Default**

FLOAT

**DialHandAnimationStyle**


---

```
DialHandAnimationStyle = GROW | FADE | NONE
```

Specifies how the dial hands initially appear in a dial chart. This parameter is only valid in SVG or SVGWeb output formats.

### Attributes

Style

### Style

---

Style refers to the manner in which dial hands are first rendered in a dial chart.

#### Example:

```
DialHandAnimationStyle = GROW;
```

#### Values

GROW	The dial hands rise from a diameter of zero to their actual values.
FADE	The dial hands fade in.
NONE	The dial hands are immediately visible.

#### Default

NONE

## Dials

---

```
Dials = (Name, StartAngle, StopAngle, RadiusPercentage, NONE|INSIDE|OUTSIDE), ...;
```

The `Dials` parameter, essential to dial charts, arranges the appearance for a set of uniquely named dials within a dial chart. There may be more than one dial in a dial chart, and they may overlap. For example, an analog clock can be considered to have three overlapping dials: hour, minute, and second. Dials may also be arranged next to each other in concentric circles. Dials are layers one atop the other, the first one defined being on the bottom and subsequent dials above it, overlapping.

#### Used in These Charts

Dial

#### Example:

```
Dials = ("Hour Dial", 0, 360, 100, INSIDE),
        ("Minute Dial", 0, 360, 100, INSIDE),
        ("Second Dial", 0, 360, 100, NONE);

Dials = ("Internal Pressure", -135, 135, 100, INSIDE),
        ("Atmospheric Pressure", -135, 135, 60, INSIDE),
        ("Pressure Change", -135, 135, 30, INSIDE);
```

#### Attributes

Name	StartAngle	StopAngle	RadiusPercentage	(Switch)
------	------------	-----------	------------------	----------

## *StartAngle*

---

`StartAngle` indicates the angle from which the dial starts, going counter-clockwise from the vertical. This parameter, along with `StopAngle`, allows you to create dials that occupy less than a full circle.

### *Example:*

```
Dials = ("Hour Dial", 0, 360, 100, INSIDE),
        ("Minute Dial", 0, 360, 100, INSIDE),
        ("Second Dial", 0, 360, 100, NONE);

Dials = ("Internal Pressure", -135, 135, 100, INSIDE),
        ("Atmospheric Pressure", -135, 135, 60, INSIDE),
        ("Pressure Change", -135, 135, 30, INSIDE);

Dials = ("Dial", -180, 180, 100, INSIDE);
```

### *Values*

Numerical degrees from 0 to + or -360

### *Default*

No defaults

## *StopAngle*

---

`StopAngle` indicates the angle at which the dial stops, going counter-clockwise from the vertical. This parameter, along with `StartAngle`, allows you to create dials that occupy less than a full circle.

### *Example:*

```
Dials = ("Hour Dial", 0, 360, 100, INSIDE),
        ("Minute Dial", 0, 360, 100, INSIDE),
        ("Second Dial", 0, 360, 100, NONE);

Dials = ("Internal Pressure", -135, 135, 100, INSIDE),
        ("Atmospheric Pressure", -135, 135, 60, INSIDE),
        ("Pressure Change", -135, 135, 30, INSIDE);

Dials = ("Dial", -180, 180, 100, INSIDE);
```

### *Values*

Numerical degrees from 0 to + or -360

### *Default*

No defaults

## *RadiusPercentage*

---

`RadiusPercentage` controls the width of the dial, in the radial direction, that the dial occupies in the chart. This parameter allows you to create concentric dials.



**Example:**

```
Dials = ("Hour Dial", 0, 360, 100, INSIDE),
        ("Minute Dial", 0, 360, 100, INSIDE),
        ("Second Dial", 0, 360, 100, NONE);

Dials = ("Internal Pressure", -135, 135, 100, INSIDE),
        ("Atmospheric Pressure", -135, 135, 60, INSIDE),
        ("Pressure Change", -135, 135, 30, INSIDE);

Dials = ("Dial", -180, 180, 100, INSIDE);
```

**Values**

Numerical percentage from 0 to 100

**Default**

No defaults

**Switch**

This switch controls where the tics fall on the dial's face.

**Example:**

```
Dials = ("Hour Dial", 0, 360, 100, INSIDE),
        ("Minute Dial", 0, 360, 100, INSIDE),
        ("Second Dial", 0, 360, 100, NONE);

Dials = ("Internal Pressure", -135, 135, 100, INSIDE),
        ("Atmospheric Pressure", -135, 135, 60, INSIDE),
        ("Pressure Change", -135, 135, 30, INSIDE);

Dials = ("Dial", -180, 180, 100, INSIDE);
```

**Values**

NONE	No tics are shown
INSIDE	Tics appear within the dial's face
OUTSIDE	Tics appear outside (but adjacent to) the dial's face

**Default**

No defaults

**DialScale**

```
DialScale = (Name, MinValue, MaxValue, StepValue);
```

The `DialScale` parameter specifies the minimum and maximum data values that will be displayed for the named dials in a dial chart. If the `DialScale` parameter is not defined, or the `MinValue` and `MaxValue` parameters are the same or one of them is not defined, then the tic mark locations will be automatically determined based on the actual data values being displayed. That is, the dial will be "autoscaled" using the current data values to determine "reasonable" values for `MinValue`, `MaxValue` and `StepValue`. If values are supplied for any of `MinValue`, `MaxValue`, or `StepSize`, those values will be used as part of the autoscaling.

*Used in These Charts*

Dial

*Example:*

```
DialScale = ("Hour Dial",0,12,1), ("Minute Dial",0,60,1), ("Second Dial",0,60,1);
```

*Attributes*

Name	MinValue	MaxValue	StepValue
------	----------	----------	-----------

*MinValue*

MinValue sets the absolute lower visible limit for the dial scale.

*Example:*

```
DialScale = ("Hour Dial",0,12,1), ("Minute Dial",0,60,1), ("Second Dial",0,60,1);
DialScale = ("Internal Pressure",0,10000,1000), ("Atmospheric
Pressure",10,100,1), ("Pressure Change",0,1000,100);
```

*Values*

Any real number less than MaxValue

*Default*

None

*MaxValue*

MaxValue sets the absolute upper visible limit for the dial scale.

*Example:*

```
DialScale = ("Hour Dial",0,12,1), ("Minute Dial",0,60,1), ("Second Dial",0,60,1);
DialScale = ("Internal Pressure",0,10000,1000), ("Atmospheric
Pressure",10,100,1), ("Pressure Change",0,1000,100);
```

*Values*

Any real number greater than MinValue

*Default*

None

*StepValue*

StepValue is optional, and may be used to specify a given step between tic marks along the bottom axis, starting with the MinValue. If StepValue is not an even multiple of the difference between the MinValue and MaxValue, then no tic mark will be displayed at the MaxValue. If StepValue is not defined, tic marks will be placed at "reasonable" locations along the axis, depending on the range of values being displayed.

*Example:*

```
DialScale = ("Internal Pressure",0,10000,1000), ("Atmospheric
Pressure",10,100,1), ("Pressure Change",0,1000,100);
```

### Values

Any real number between `MinValue` and `MaxValue`

### Default

1

## DialSectorAnimationStyle

---

`DialSectorAnimationStyle = GROW | FADE | NONE`

Specifies how the dial sectors initially appear in a dial chart. This parameter is only valid in SVG or SVGWeb output formats.

### Attributes

Style

### Style

---

`style` refers to the manner in which dial sectors are first rendered in a dial chart.

### Example:

```
DialSectorAnimationStyle = GROW;
```

### Values

`GROW` The dial sectors rise from a diameter of zero to their actual values.

`FADE` The dial sectors fade in.

`NONE` The dial sectors are immediately visible.

### Default

`NONE`

## DialSize

---

`DialSize = (minWidth, minHeight, maxWidth, maxHeight) ;`

The `DialSize` parameter can be used to set minimum and maximum sizes for the actual dial in a dial chart. This allows programmers to guarantee that the dial will always be the same size regardless of the length of the strings in the legend or tic labels. `DialSize` has the following interaction with `DialSquare`; if the minimum or maximum dimensions specified are not square, and `DialSquare` is `ON`, then the dial will be inscribed in a square with a dimension that ranges from the smallest of the minimum values to the smallest of the maximum values.

### Used in These Charts

Dial

**Example:**

```
DialSize = (100,100,200,200);
```

**Attributes**

*minWidth, minHeight, maxWidth, maxHeight*

---

**DialSquare**

---

```
DialSquare = mode;
```

The `DialSquare` parameter tells the dial chart that the appearance of the dial should be kept as high as it is wide.

**Used in These Charts**

Dial

**Example:**

```
DialSquare = ON|OFF;
```

**Attributes**

*Mode*

---

**DialTicLabels**

---

```
DialTicLabels = (Name, Label1, Label2 ..., LabelN), ...;
```

`DialTicLabels` allows you to label the tic marks for a named dial in a dial chart.

**Used in These Charts**

Dial

**Example:**

```
DialTicLabels = ("Hour Dial", "12", "1", "2", "3", "4", "5", "6", "7", "8",  
               "9", "10", "11");  
DialTicLabels = ("Internal Pressure", "0", "1000", "2000", "3000", "4000",  
               "5000", "6000", "7000", "8000", "9000", "10,000");
```

**Attributes**

*Name*                      *Label*

---

**Label**

---

`Label` can be one of several strings that identify the tic marks around a dial in a dial chart. This is equivalent to putting numbers around the face of a clock.

**Example:**

```
DialTicLabels = ("Hour Dial", "12", "1", "2", "3", "4", "5", "6", "7", "8",
               "9", "10", "11");
DialTicLabels = ("Internal Pressure", "0", "1000", "2000", "3000", "4000",
               "5000", "6000", "7000", "8000", "9000", "10,000");
```

**Values**

Use any string. If there are more labels than tics, only the labels that correspond to tic marks will be shown. If there are fewer labels than tic marks, all labels will be shown.

**Default**

None

**DialTicLabelStyles**

```
DialTicLabelStyles = (Name, ON/OFF, LabelPos, Color, FontName, FontSize, Angle, interiorAlignment), ...;
```

`DialTicLabelStyles` controls the appearance of the tic mark labels defined with the `DialTicLabels` parameter.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are LEFT, RIGHT, or CENTER.

**Used in These Charts**

Dial

**Example:**

```
DialTicLabelStyles = ("Hour Dial", "ON", 1.1, black, "Helvetica", 14, 0),
                   ("Minute Dial", "ON", 1.1, black, "Helvetica", 14, 0);
```

**Attributes**

<i>Name</i>	(Switch)	<i>LabelPos</i>	<i>Color</i>	<i>FontName</i>
<i>FontSize</i>	<i>Angle</i>	<i>interiorAlignment</i>		

**Switch**

This switch turns the tic labels' visibility on and off.

**Example:**

```
DialTicLabelStyles = ("Hour Dial", "ON", 1.1, black, "Helvetica", 14, 0),
                   ("Minute Dial", "ON", 1.1, black, "Helvetica", 14, 0);
```

**Values**

ON	Show the tic labels
OFF	Hide the tic labels

**Default**

None

**LabelPos**

---

The `LabelPos` attribute in the dial chart works the same way the `LabelPos` parameter does for the pie chart: it defines the position of the tic mark labels relative to the width of the dial. A value of 1.1 will place each label just outside the dial face, while a value of 0.6 will place each label inside of the dial face. For instance, clocks are often made with their tic mark labels on the outside of the dial face, but gauges tend to have them just on the inside of the dial face along with the tic marks.

**Example:**

```
DialTicLabelStyles = ("Hour Dial", "ON", 1.1, black, "Helvetica", 14, 0),
                    ("Minute Dial", "ON", 1.1, black, "Helvetica", 14, 0);
```

**Values**

Positive real numbers, generally between 0 and 2.

**Default**

None

**DialTics**

---

```
DialTics = (Name, Color, LineWidth, PercentofRadius), ...;
```

`DialTics` controls the appearance of the tic marks (short lines) around a dial in a dial chart.

**Used in These Charts**

Dial

**Example:**

```
DialTics = ("Hour Dial", gray, 1, 10), ("Minute Dial", gray, 1, 5);
DialTics = ("Internal Pressure", black, 2, 5),
           ("Internal Pressure", green, 2, 5),
           ("Pressure Change", red, 1, 4);
```

**Attributes**

Name	Color	LineWidth	PercentofRadius
------	-------	-----------	-----------------

**PercentofRadius**

---

`PercentofRadius` controls the length of the tics as a percentage of the dial's radius.

**Example:**

```
DialTics = ("Hour Dial", gray, 1, 10), ("Minute Dial", gray, 1, 5);
DialTics = ("Internal Pressure", black, 2, 5),
           ("Internal Pressure", green, 2, 5),
           ("Pressure Change", red, 1, 4);
```

**Values**

Percentages from 0 to 100

**Default**

No defaults

**DrawFences**

---

```
DrawFences = ON | OFF;
```

The `DrawFences` parameter specifies whether or not to draw fences on the chart.  
The default is ON.

**Used in These Charts**

Box

**Example:**

```
DrawFences = OFF;
```

**Attributes**

*Mode*

**DrawOrder**

---

```
DrawOrder[N] = (SYMBOL, ...);
```

The `DrawOrder` parameter defines the order in which Bar, Line, LineFill and Stock options will be drawn.

BAR	refers to ALL bar sets
LINE	refers to line sets which do NOT have a fill
LINEFILL	refers to line sets which DO have a fill
STOCK	refers to ALL stock sets

**Used in These Charts**

Combo, Pareto, Stock

**Example:**

```
DrawOrder = (BAR);
```

**Attributes**

*Symbol*

**DwellAnimationHighlightBorderStyle**

---

```
DwellAnimationHighlightBorderStyle = (lineType, LineWidth, lineColor);
```

Defines the border style to be applied to a datapoint when `DwellAnimationStyle = HIGHLIGHT`.

### *Used in These Charts*

All

### *Example:*

```
DwellAnimationHighlightBorderStyle = (DASHED,1,BLACK);
```

### *Attributes*

<i>LineType</i>	<i>LineWidth</i>	<i>LineColor</i>
-----------------	------------------	------------------

### *LineType*

---

#### *Values*

SOLID	A solid line is displayed.
DOTTED	A dotted line is displayed.
DASHED	A dashed line is displayed.
DOTDASH	Alternating dots and dashes are displayed.

#### *Default*

SOLID

## **DwellAnimationHighlightFill**

---

```
DwellAnimationHighlightFill = Color;
```

Defines the color used to fill a datapoint when `DwellAnimationStyle = HIGHLIGHT`.

### *Used in These Charts*

All

### *Example:*

```
DwellAnimationHighlightFill = BLUE);
```

### *Attributes*

*Color*

#### *Default*

NONE



## DwellAnimationStyle

---

DwellAnimationStyle = HIGHLIGHT | NONE

Defines how the chart behaves when the mouse dwells over a data point. DwellAnimation parameters are only valid for image output types (i.e. DwellAnimation does not use the applet supported SVG and SVGweb output formats).

### Example:

```
DwellAnimationStyle = HIGHLIGHT;
```

### Values

**HIGHLIGHT** The data point is highlighted using the values specified in `DwellAnimationHighlightFill` and `DwellAnimationHighlightBorderStyle`.

**NONE** No highlight is applied to the data point.

### Default

NONE

## DwellBox

---

DwellBox[N] = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);

The `DwellBox` parameter is optional. If specified, it defines a box to be displayed as a background for each dwell label specified by a `DwellLabel` parameter. The box will be automatically scaled to fit each dwell label.

### Used in These Charts

All

### Example:

```
DwellBox = (yellow, RAISED, 3);
```

### Attributes

<i>Color</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>ImageURL</i>
<i>ImageFormat</i>	<i>BorderColor</i>	<i>TRCornerStyle</i>	<i>BRCornerStyle</i>
<i>BLCornerStyle</i>	<i>CornerColor</i>		

### XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

## DwellLabel

---

```
DwellLabel[N] = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);
```

If the `DwellLabel` parameter is defined, then a label will automatically be displayed whenever the mouse cursor dwells over a given data value. The attributes defined for the `DwellLabel` parameter specify the format of each label, not its text value.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

### Used in These Charts

All

#### Example:

```
DwellLabel = ("", black, "Courier", 12, LEFT);
```

#### Attributes

<i>Label</i>	<i>Color</i>	<i>FontName</i>	<i>FontSize</i>
<i>Angle</i>			

## DwellOffset

---

```
DwellOffset = size;
```

The `DwellOffset` specifies the length of the sides and define the square area around a "hotspot" associated with an active label. When the mouse moves into this hotspot, an active label, if defined, will be displayed. When the mouse is clicked within this hotspot, the target URL, if defined, will be "drilled" to. By default, `DwellOffset = 20`, which means the hotspot for a datapoint will be a 20 by 20 pixel square centered over the data point. Set `DwellOffset` to something smaller to prevent overlap between the hotspots of tightly packed datapoints.

If you are using a symbol then the symbol size is added on to the dwell hotspot size to compute the dwell offset area.

### Used in These Charts

Line, XY

#### Example:

```
DwellOffset = 5;
```

#### Attributes

*size*

## Edges

---

Edges[N] = (NodeStart, NodeEnd, Color, Direction, LineStyle, LineWidth, ArrowStyle, ArrowLength, ArrowWidth), ...;

Within a diagram chart, the Edges parameter lists and defines the appearance of the lines to be drawn between named nodes.

### Used in These Charts

Diagram

#### Example:

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);

Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);

Edges = ("CEO", "coo-node", dimgray, NONE, SOLID, 1),
("coo-node", "sales-node", dimgray, NONE, SOLID, 1),
("coo-node", "marketing-node", dimgray, NONE, SOLID, 1),
("sales-node", "VP Sales", dimgray, NONE, SOLID, 1),
("coo-node", "COO", dimgray, NONE, SOLID, 1),
("marketing-node", "VP Marketing", dimgray, NONE, SOLID, 1),
("VP Marketing", "Webmaster", dimgray, NONE, SOLID, 1);
```

#### Attributes

NodeStart	NodeEnd	Color	Direction	LineStyle
LineWidth	ArrowStyle	ArrowLength	ArrowWidth	

#### NodeStart

---

NodeStart names the node where the edge begins.

#### Example:

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);

Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);

Edges = ("CEO", "coo-node", dimgray, NONE, SOLID, 1),
("coo-node", "sales-node", dimgray, NONE, SOLID, 1),
("coo-node", "marketing-node", dimgray, NONE, SOLID, 1),
("sales-node", "VP Sales", dimgray, NONE, SOLID, 1),
("coo-node", "COO", dimgray, NONE, SOLID, 1),
("marketing-node", "VP Marketing", dimgray, NONE, SOLID, 1),
("VP Marketing", "Webmaster", dimgray, NONE, SOLID, 1);
```

#### Values

The identifying string, in double-quotes, of a node named with the Nodes parameter.

#### Default

No defaults

---

## NodeEnd

---

NodeEnd names the node where the edge ends.

### Example:

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);

Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);

Edges = ("CEO", "coo-node", dimgray, NONE, SOLID, 1),
        ("coo-node", "sales-node", dimgray, NONE, SOLID, 1),
        ("coo-node", "marketing-node", dimgray, NONE, SOLID, 1),
        ("sales-node", "VP Sales", dimgray, NONE, SOLID, 1),
        ("coo-node", "COO", dimgray, NONE, SOLID, 1),
        ("marketing-node", "VP Marketing", dimgray, NONE, SOLID, 1),
        ("VP Marketing", "Webmaster", dimgray, NONE, SOLID, 1);
```

### Values

The identifying string, in double-quotes, of a node named with the Nodes parameter.

### Default

No defaults

### Direction

---

Direction the direction of the arrow head(s), if any, for the edge.

### Example:

```
Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);

Edges = ("CEO", "coo-node", dimgray, NONE, SOLID, 1),
        ("coo-node", "sales-node", dimgray, NONE, SOLID, 1),
        ("coo-node", "marketing-node", dimgray, NONE, SOLID, 1),
        ("sales-node", "VP Sales", dimgray, NONE, SOLID, 1),
        ("coo-node", "COO", dimgray, NONE, SOLID, 1),
        ("marketing-node", "VP Marketing", dimgray, NONE, SOLID, 1),
        ("VP Marketing", "Webmaster", dimgray, NONE, SOLID, 1);
```

### Values

NONE	No Arrows are shown
FROMTO	Arrow from NodeStart node to NodeEnd node
TOFROM	Arrow from NodeEnd to NodeStart node
BOTH	Arrow in both directions

### Default

FROMTO

### ArrowStyle

---

ArrowStyle, if used, determines the style of the arrowhead(s) on the edge. This attribute is optional.

**Example:**

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);  
Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);
```

**Values**

NONE	No Arrows are shown
SHARP	Triangular tip with sharper corners than BLOCK
ROUND	Circular tip
BLOCK	Triangular tip
LINE	Plain line tip

**Default**

BLOCK

**ArrowLength**

---

ArrowLength determines the length, in pixels, from the start to the tip of the arrow(s) on an edge. This attribute is optional.

**Example:**

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);  
Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);
```

**Values**

Whole pixel numbers

**Default**

No defaults

**ArrowWidth**

---

ArrowWidth determines the width, in pixels, of the arrow(s) on an edge. This attribute is optional.

**Example:**

```
Edges = ("node1", "node2", black, NONE, SOLID, 3, BLOCK, 15, 8);  
Edges = ("server", "client", black, FROMTO, SOLID, 1, BLOCK, 9, 4);
```

**Values**

Whole pixel numbers

**Default**

No defaults

---

## EightyLineSetName

---

```
EightyLineSetName = name;
```

Name assigned to the 80% line. Used in the legend if the Legend CDL parameter does not define a label for the 80% line.

### *Used in These Charts*

Pareto

### *Example:*

```
EightyLineSetName = "80% Line"
```

---

## EightyTwentyLineStyle

---

```
EightyTwentyLineStyle = (Type, LineWidth, Color, FillColor, LineType);
```

This parameter specifies the line style to be displayed for the 80/20 lines.

### *Used in These Charts*

Pareto

### *Example:*

```
EightyTwentyLineStyle = (SOLID, 3, blue, blue, NORMAL);
```

### *Attributes*

<i>Color</i>	<i>FillColor</i>	<i>LineType</i>	<i>LineWidth</i>	<i>Type</i>
--------------	------------------	-----------------	------------------	-------------

### *FillColor*

---

If this attribute is not NULL, then the area under the 80/20 lines be filled with the given color.

### *Example:*

```
EightyTwentyLineStyle = (SOLID, 3, red, pink, NORMAL);  
      <!-- red line with pink fill -->
```

### *Values*

NULL	Also, value left unspecified: No color fills the area under the line.
Any legal color	Area under the line is filled. See Chapter 4 for the <code>Color</code> attribute.

### *Default*

None

### *Type*

---

The style of line to draw

**Values**

NONE  
 SOLID  
 DOTTED  
 DASHED  
 DOTDASH

**Default**

SOLID

**LineType**

---

The type of line to use to connect the points in the line set.

**Values**

NORMAL  
 FIT  
 BOTH

**Default**

NORMAL

**EightyTwentyLineSymbol**

---

`EightyTwentyLineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth) ;`

`EightyTwentyLineSymbol` specifies the symbols to be displayed for the 80/20 lines.

**Used in These Charts**

Pareto

**Example:**

```
EightyTwentyLineSymbol = (CIRCLE, 6, BOTH, white, 1, grey, 0);
```

**Attributes**

<i>BorderColor</i>	<i>BorderWidth</i>	<i>ImageURL</i>	<i>ShadowWidth</i>
Size	Style	SymbolColor	Type

**Size**

---

`Size` specifies the size of the symbol in pixels. This attribute is ignored for `IMAGE` symbols.

**Example:**

```
EightyTwentyLineSymbol = (CIRCLE, 6, BOTH, white, 1);
```

**Values**

Any integer value in pixels

**Default**

None

**SymbolColor**

---

SymbolColor specifies for foreground color of the EightyTwentyLineSymbol.

**ShadowThickness**

---

ShadowThickness specifies for size of the shadow behind a EightyTwentyLineSymbol. Any number other than 0 causes NetCharts to choose a shadow size based on the size of the symbol. Specifying a value of 0 suppresses the shadow.

**Style**

---

Style specifies how the EightyTwentyLineSymbol should be drawn, including FILLED, OUTLINED, or BOTH. If FILLED is specified, the symbol is filled with the line set color. If OUTLINED is specified, only the outline is drawn, using the line set color. If BOTH is specified, then the symbol is filled with the line set color and the outline is drawn using the borderColor.

**Example:**

```
EightyTwentyLineSymbol = (CIRCLE, 6, BOTH, white, red);
```

**Values**

FILLED	Symbol is filled with the cumulative line color.
OUTLINED	Only the outline is drawn, using the cumulative line color.
BOTH	Symbol is filled with the 80/20 line color and the outline is drawn using the BorderColor

**Default**

None

**Type**

---

Type specifies the type of symbol to be displayed on the line set.

**Example:**

```
EightyTwentyLineSymbol = (SQUARE, 6, BOTH, cyan, 1);
```



### Values

NONE	No symbol is displayed.
CIRCLE	Displays circles
SQUARE	Displays squares
DIAMOND	Displays diamonds
CROSS	Displays crosses
TARGET	Displays targets (bulls-eye)
TRIANGLEDOWN	Displays downward pointing triangles
TRIANGLEUP	Displays upward pointing triangles
IMAGE	If specified, the <code>ImageURL</code> attribute is required and will be used to load a GIF image for the symbol.

### Default

None

## FenceActiveLabels

---

```
FenceActiveLabelsN = ("Label1", "URL1", "Target1"),...;
```

FenceActiveLabelsN define the active labels associated with fences in data set N.

### Used in These Charts

Box Chart

#### Example:

```
FenceActiveLabels = ("LabelText",,);
```

### Attributes

*Label, Target, URL*

## FencePosition

---

```
FencePosition = Type;
```

FencePosition specifies whether to draw the fences that are within the Inter Quartile Range, (the box), over or under the box.

### Used in These Charts

Box Chart

#### Example:

```
FencePosition = UNDER;  
FencePosition = OVER;
```

### Attributes

Type

## Type

---

Type refers to the fences that are within the Inter Quartile Range, (the box).

### Values

UNDER            Draw fences under the box  
OVER             Draw fences over the box

### Default

OVER

## FontEncoding

---

The FontEncoding parameter refers to the "codepage" that should be used when mapping fonts. Some examples of FontEncodings are UTF-8, cp1252, cp850, iso 8859. This rarely needs to be changed, even when non-ASCII fonts are being used.

## Footer

---

```
Footer = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);
```

Footer, which is universal to NetCharts applets, describes an optional title, or label, that sits at the bottom of a chart, or at its "foot," and uses standard attributes for string text, text color, font, font size, and label rotation.

### Used in These Charts

All

### Example:

```
Footer = ("This Is A\nMulti-Line\nFooter", darkred);
```

interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
exteriorAlignment	Specifies the alignment for the entire Title object.

The legal values for interiorAlignment and exteriorAlignment are LEFT, RIGHT, or CENTER.

### Attributes

Label   Color   FontName   FontSize   Angle   interiorAlignment   exteriorAlignment

## FooterActiveLabel

---

```
FooterActiveLabel = ("Label", "URL", "Target");
```

`FooterActiveLabel` defines a single active label destination for the footer title.

### *Used in These Charts*

All

### *Example:*

```
FooterActiveLabel = ("Where To Go", "demo.html", "frame1"),
```

### *Attributes*

Label	URL	Target
-------	-----	--------

## FooterBox

---

```
FooterBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle,
BRCornerStyle, BLCornerStyle, CornerColor);
```

The `FooterBox` specifies a background region just for the chart `footer` title.

### *Used in These Charts*

All

### *Attributes*

<i>BorderColor</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>Color</i>
<i>ImageFormat</i>	<i>ImageURL</i>	<i>TRCornerStyle</i>	<i>BRCornerStyle</i>
<i>BLCornerStyle</i>	<i>CornerColor</i>		

## *XXCornerStyle*

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

## Format

---

```
TopFormat[N] = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
BottomFormat[N] = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
LeftFormat[N] = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
RightFormat[N] = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
```

`Format` adjusts the numeric labels that are automatically generated for the given axis, should one be defined.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```

TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
TopFormat = (INTEGER);
TopFormat = (FLOAT, "$%,9.2f", ,);

```

**Attributes**

FormatType      FormatExpr      TimeBase      TimeUnit

**FormatType**

FormatType specifies the type of number being displayed on the given axis.

**Example:**

```
LeftFormat = (INTEGER, "$%f", ,);
```

**Values**

DATE      Axis value are shown as date and/or time values. See **Appendix A: Date and Time Values** for further detail.

FLOAT      Axis values are shown with decimal parts.

INTEGER      Axis values are shown as integers, and are rounded if necessary.

DECIMAL      Axis values are shown as decimals, see <http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html> for more information.

The characters listed here are used in non-localized patterns. Localized patterns use the corresponding characters taken from this formatter's `DecimalFormatSymbols` object instead, and these characters lose their special status. Two exceptions are the currency sign and quote, which are not localized.

Symbol	Location	Localized?	Meaning
0	Number	Y	Digit
#	Number	Y	Digit, zero shows as absent
.	Number	Y	Decimal separator or monetary decimal separator
-	Number	Y	Minus sign
,	Number	Y	Grouping separator
E	Number	Y	Separates mantissa and exponent in scientific notation. <i>Need not be quoted in prefix or suffix.</i>
;	Subpattern boundary	Y	Separates positive and negative subpatterns
%	Prefix or suffix	Y	Multiply by 100 and show as percentage
\u2030	Prefix or suffix	Y	Multiply by 1000 and show as per mille
¤ (\u00A4)	Prefix or suffix	N	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.

,	Prefix or suffix	N	Used to quote special characters in a prefix or suffix, for example, "'#'" formats 123 to "#123". To create a single quote itself, use two in a row: "' o'clock".
---	------------------	---	---

**Default**

INTEGER

**TimeBase**

The `TimeBase` attribute specifies the base date to be used when determining the actual date or time value when using a time unit or numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
LeftFormat = (INTEGER, "$%f", "10", );
```

**Values**

String values representing dates or times

**Default**

None

**TimeUnit**

The `TimeUnit` attribute controls the time multiplier to be used when determining the actual data/time value when using a numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

**Values**

String values representing dates or times

**Default**

None

**GraphLayout**

```
GraphLayout = Type;
```

`GraphLayout` defines the bar orientation in a chart.

### *Used in These Charts*

Bar, Box, Combo, Line, Stock,

### *Example:*

```
GraphLayout = HORIZONTAL;  
GraphLayout = VERTICAL;
```

### *Attributes*

Type

### *Type*

---

Type refers to the direction the bars lie in the graph.

### *Example:*

```
GraphLayout = HORIZONTAL;  
GraphLayout = VERTICAL;
```

### *Values*

HORIZONTAL    The bar sets are oriented running from left to right.  
VERTICAL      The bar sets are oriented rising from bottom to top.

### *Default*

VERTICAL

## **GraphType**

---

```
GraphType = Type;
```

GraphType defines the type of multiple-line graph to be displayed, and mostly affects how the stacking is achieved.

### *Used in These Charts*

Bar, Combo, Line, Stock,

### *Type*

---

Type refers to the manner in which lines or bars are stacked in a chart.

### *Example:*

```
GraphType = STACK;
```

### Values

GROUPSTACK	For bar series only, sets are collected together in groups at each tic, and at each tic a group member has a stack of values.
ROWS	The bar or line sets are displayed separately. If <code>3DDepth</code> is nonzero, then they will be displayed in separate rows, from front to back.
STACK	The bar or line sets are stacked on top of each other. That is, as each data set is drawn, its values are added to previous values displayed. Negative and <code>NULL</code> values are treated as zero.
PERCENT	The bar or line sets are stacked on top of each other, normalized to 100 percent. That is, as each data set is drawn, its values are added to previous values displayed and displayed as a percentage of the total of all values. Negative and <code>NULL</code> values are treated as zero.

### Default

STACK

## Grid

---

```
Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;
```

All Visual Mining charts, except for the Dial chart, Diagram chart, and Pie chart support the display of one or more grids behind the data. The grid layouts and styles can be independently specified, and can be associated with any of the axes being displayed. The `Grid` parameter allows you to specify up to three grid sets.

Since all of the `Grid` parameters are defined as vectors, you can specify more than one grid for the same chart. This allows arbitrary combinations of styles and spacing to achieve a wide array of chart grids.

A `NULL` background color can be assigned to second and subsequent grids so that it will not overwrite the first grid. However, looking at the second example below, if one didn't specify a `NULL` background color, then `white` would have been used as the default and the second grid would completely overwrite the first.

### Used in These Charts

Bar, Box, Bubble, Combo, Line, Pareto, Radar, Stock, Strip, Time, X-Y

### Example:

```
Grid = (green, white, black, "../images/mychartbg.gif", CENTER);

<!-- The following produces a striped grid -->
Grid = (green, white), (black, null);
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

### Attributes

LineColor	BackgroundColor	BorderColor	BackgroundImage
ImageFormat			

### LineColor

---

`LineColor` specifies the color of the grid lines. See the common `Color` attribute in **Chapter 4** for details.

## *BackgroundColor*

---

`BackgroundColor` specifies the color for the grid's background. See the common `Color` attribute for details.

## *BackgroundImage*

---

`BackgroundImage` specifies an image file for the grid's background. See the common `Image` attribute for details.

## **Grid3DDepth**

---

```
Grid3DDepth = depth;
```

In any chart that can display bar data sets, the `Bar3DDepth` parameter defines the depth of each bar in pixels, as well as the depth of the underlying grids.

In all cases, the grids defined in the `Grid` parameters automatically adjust to the current depth setting.

### *Used in These Charts*

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
Grid3DDepth = 13; <!-- displays a 3-D grid 13 pixels deep -->
```

### *Attributes*

Depth

### *Depth*

---

If the `Depth` parameter is set to 0, then 2D bars and grids are displayed. If that parameter is not defined, then a default depth will be chosen.

### *Example:*

```
Grid3DDepth = 10; <!-- displays a 3-D grid 10 pixels deep -->
```

```
Grid3DDepth = 0; <!-- displays a 2-D grid -->
```

### *Values*

Depth in pixels

### *Default*

For charts with 3-D bars, a depth commensurate with the bar depth is chosen automatically.

## **GridAnimationStyle**

---

```
GridAnimationStyle = FADE | NONE
```



Defines how grid backgrounds initially appear in a chart. This parameter is only valid in SVG or SVGweb output formats.

**Example:**

```
GridAnimationStyle = FADE;
```

**Values**

FADE	The grids fade in.
NONE	The grids are immediately visible.

**Default**

NONE

## GridAxis

---

```
GridAxis = (XAxisI, YAxisI), ...;
```

`GridAxis` specifies pairs of X and Y axes for each grid set in the chart, matching the sets in the `Grid` parameter. By default, grids use the bottom and left axes to determine the spacing of the grid lines. The `GridAxis` parameter allows you to specify arbitrary combinations of axes for each grid being displayed. This allows you to easily display grids with different types of lines spaced at various intervals. `GridAxis` is commonly used in conjunction with the `Grid` and `GridLine` parameters.

**Used in These Charts**

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
Grid = (green, white), (black, null);
GridLine = (HORIZONTAL, SOLID, 2), (HORIZONTAL, DOTTED, 1);
GridAxis = (BOTTOM, LEFT), (BOTTOM, RIGHT);
```

**Attributes**

XAxis                      YAxis

## GridBlockActiveLabels

---

```
GridBlockActiveLabels = ("Label1", "URL1", "Target1"), ...;
```

`GridBlockActiveLabels` The `GridBlockActiveLabels` parameter specifies a list of custom active labels to be associated with each grid block. The labels will be displayed whenever the mouse "dwells" over a given grid block being displayed.

**Used in These Charts**

Heat Map

**Attributes**

*Label*                      *URL*                      *Target*

---

**GridBlockBackgroundColor**

---

```
GridBlockBackgroundColor = color...;
```

`GridBlockBackgroundColor` specifies the default grid block background color.

**Used in These Charts**

Heat Map

**Attributes**

*Color*

**Default**

White

---

**GridBlockCellColorType**

---

```
GridBlockCellColorType = type;
```

`GridBlockCellColorType` determines the cell color type.

**Used in These Charts**

Heat Map

**Attributes**

*Type*

**Example:**

```
GridBlockCellColorType=COLORTABLE;
```

**Values**

COLORTABLE  
EXPRESSIONS  
SPECTRUM

---

**GridBlockColors**

---

```
GridBlockColors = (color1, color2,...colorN);
```

If too few colors are specified for the grid blocks the color pattern repeats.

*Used in These Charts*

Heat Map

*Attributes**Color*

## GridBlockColorSpectrum

---

```
GridBlockColorSpectrum = (color1,color2,min,max,gradientstep);
```

`GridBlockColorSpectrum` defines attributes for a color spectrum. A color spectrum is generated from two colors which are the starting and ending colors and from the number of steps which is the number of color buckets. The minimum and maximum values represent the data range for the entire spectrum. Each color bucket in the spectrum represents a certain data range. If a data value of a grid block falls within that range, it will be shown with its associated color. The spectrum colors start at `color1`, go to white or almost white, depending on the number of steps, and then end at `color2`. The number of steps determines the amount each color bucket is changed by and the data values each color represents.

*Used in These Charts*

Heat Map

*Attributes*

*Color1*                      *Color2*                      *Min*                      *Max*      *gradientstep*

*Example*

```
GridBlockColorSpectrum = (black,greyscale,2,45,20);
```

## GridBlockExpressions

---

```
GridBlockExpressions = ("operator",value1,value2,color),...;
```

Each color expression has an operator, value(s) to compare against, and a color to use if the expression is true. The BETWEEN operator is the only operator that is used to compare 2 values.

*Used in These Charts*

Heat Map

*Attributes*

*operator*                      *value1*                      *value2*                      *color*

### *Operator*

---

Specifies the operator used

*Values*

```
">", "<", ">=", "<=", "==", "!=", BETWEEN
```

**Example**

```
GridBlockExpressions = ("BETWEEN", 2, 9, yellow), ("==", 1, , aqua)...
```

**GridBlockLabel**

```
GridBlockLabel = ("mode", color, "font name", font size);
```

```
GridBlockTopLabel = ("mode", color, "font name", font size, angle, interiorAlignment);
```

```
GridBlockLeftLabel = ("mode", color, "font name", font size, angle, interiorAlignment);
```

```
GridBlockRightLabel = ("mode", color, "font name", font size, angle, interiorAlignment);
```

```
GridBlockBottomLabel = ("mode", color, "font name", font size, angle, interiorAlignment);
```

Defines the text style for the grid block labels

**Used in These Charts**

Heat Map

**Attributes**

mode                                  font name                                  font size                                  angle interiorAlignment

**Example:**

```
GridBlockLeftLabel = ("ON", purple, "ARIAL", 12, 0, CENTER);
```

**Default**

ON

**GridBlockLabels**

```
GridBlockLabels = label1,label2,...labelN;
```

```
GridBlockTopLabels = label1,label2,...labelN;
```

```
GridBlockBottomLabels = label1,label2,...labelN;
```

```
GridBlockLeftLabels = label1,label2,...labelN;
```

```
GridBlockRightLabels = label1,label2,...labelN;
```

Defines the list of labels to center over the grid blocks

**Used in These Charts**

Heat Map

**Attributes**

Label

**Example**

```
GridBlockLabels = "Derek", "Joe", "Reggie";
```

## GridBlockLayout

---

```
GridBlockLayout = =(Height,Width);
```

Defines the size of grid. If not defined the grid attempts to layout in a square. The maximum grid size is 50 x 50

### *Used in These Charts*

Heat Map

### *Attributes*

*Height*                      *Width*

### *Example*

```
GridBlockLayout=(2,7);
```

## GridBlockLine

---

```
GridBlockLine =("LineStyle", width, color);
```

Defines the line style to be displayed in the grid.

### *Used in These Charts*

Heat Map

### *Attributes*

*Style*                      *Width*                      *Color*

### *LineStyle*

---

LineStyle tells how to draw the grid line.

### *Example:*

```
GridBlockLine=("Solid", 1, pink);
```

### *Values*

SOLID	solid lines
DOTTED	dotted lines
DASHED	dashed lines
DOTDASH	dot-dash lines

---

## GridBlockSort

---

```
GridBlockSort = (SortType, SortOrder);
```

Determines the sort order for the grid values.

### *Used in These Charts*

Heat Map

### *Attributes*

*SortType*            *SortOrder*

---

### *SortType*

The item that the grid block will be sorted on.

### *Example:*

```
GridBlockSort=(Label, ASCENDING);
```

### *Values*

NONE  
LABEL  
VALUE

---

### *SortOrder*

### *Values*

ASCENDING  
DESCENDING

---

## GridBlockValues

---

```
GridBlockValues = Value1,Value2,Value3 ...;
```

Defines a list values for the heat map dataset

### *Used in These Charts*

Heat Map

### *Attributes*

*Value*

## GridBlockValueFormat

---

```
GridBlockValueFormat = (FormatType, "FormatExpr")
```

The `GridBlockValueFormat` parameter defines the format for the grid block values in the heat map

### *Used in These Charts*

Heat Map

### *Example:*

```
GridBlockValueFormat = (DECIMAL, "%2");
```

### *Attributes*

*FormatType*      *FormatString*

### *FormatType*

---

The type of number to use when formatting the values

### *Values*

FLOAT  
INTEGER  
DECIMAL

## GridBlockValueStyle

---

```
GridBlockValueStyle = ("mode", color, "font name", font size);
```

Defines styles for grid values.

### *Used in These Charts*

Heat Map

### *Attributes*

*mode*                      *font name*                      *font size*

### *Example*

```
GridBlockValueStyle = ("ON",purple,"ARIAL",12);
```

### *Default*

ON

## GridLine

---

```
GridLine = (LineType, LineStyle, LineWidth), ... ;
```

One or more grid line styles can be specified using the `GridLine` parameter. Each set of parameters defines the line properties for the corresponding grid defined in the `Grid` parameter, above.

### Used in These Charts

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

### Attributes

LineType	LineStyle	LineWidth
----------	-----------	-----------

### LineType

---

Tells where to draw the lines, is specified with a `GridLine` parameter, and refers to a `Grid` parameter.

### Example:

```
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

### Values

BOTH	draw both horizontal and vertical lines (default)
VERTICAL	draw vertical lines only
HORIZONTAL	draw horizontal lines only
NONE	draw no grid lines

### Default

BOTH

### LineStyle

---

`LineStyle` tells how to draw the grid lines, is specified with a `GridLine` parameter, and refers to a `Grid` parameter.

### Example:

```
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

### Values

SOLID	solid lines (default)
DOTTED	dotted lines
DASHED	dashed lines
DOTDASH	dot-dash lines
BAR	alternating bars instead of lines

### Default

SOLID



## GroupStackLabels

---

`GroupStackLabels = Label, Label, ...;`

This parameter is used only for grouped stacked BarCharts (GraphType=GROUPSTACK). It defines labels for each stack of data points in a group. If this value is unspecified and ShowGroupStackLabels=ON; the set names defined in the DataSets parameter will be used as the stack labels.

## GroupStackSegmentLabels

---

`GroupStackSegmentLabels = Label, Label, ...;`

This parameter is used only for grouped stacked BarCharts (GraphType=GROUPSTACK). It defines labels for each value in a stack of data points at a single tic location. These labels will be displayed in the legend of the chart. GroupStackSegmentLabels may be overridden by the LegendItems parameter.

## HandActiveLabels

---

`HandActiveLabels[N] = (Name, Label, URL, Target), ...;`

Specifies sets of active labels attached to the hands in a dial chart. Each grouped set in parenthesis, or “tuple,” has a corresponding set within a DataSet parameter.

### Used in These Charts

Dial

### Example:

```
HandActiveLabels = ("Minute", "Minute", "DialChartMin.html", "InfoFrame"),
                  ("Hour", "Hour", "DialChartHr.html", InfoFrame),
                  ("Second", "Second", "DialChartSec.html", InfoFrame);
```

### Attributes

<i>Label</i>	<i>URL</i>	<i>Target</i>
--------------	------------	---------------

## HandBorders

---

`HandBorders = (Name, LineType, LineWidth, Color);`

HandBorders specifies the line style to apply to the borders of dial hands. The default line color is black.

### Used in These Charts

Dial

### Example:

```
HandBorders = ("Hand1", DASHED, 2, DarkGray), ("Hand2", DASHED, 2, DarkGray);
```

---

**Attributes**

HandName	LineType	LineWidth	Color
----------	----------	-----------	-------

---

**LineType**

LineType specifies the style of the border to be drawn on a chart's hands.

**Values**

SOLID	Draws a solid line of LineWidth thickness.
DOTTED	Draws a dotted line of LineWidth thickness.
DASHED	Draws a dashed line of LineWidth thickness.
DOTDASH	Draws a dot-dashed line of LineWidth thickness.

**Default**

SOLID

---

**LineWidth**

LineWidth specifies the width in pixels of the border to be drawn on a chart's hands.

**Values**

1 or greater    Whole number width in pixels

**Default**

1

---

**Color**

Color specifies the color of the hand border.

---

**HandButtonBorder**

---

```
HandButtonBorder = (LineType, LineWidth, Color);
```

HandButtonBorder specifies the line style to apply to the center button of a dial. The default line color is black.

**Used in These Charts**

Dial

**Example:**

```
HandButtonBorder = (SOLID, 2, DarkGray);
```

**Attributes**

LineType                      LineWidth                      Color

**LineType**

---

LineType specifies the style of the border to be drawn on a chart's hands.

**Values**

SOLID                      Draws a solid line of LineWidth thickness.  
DOTTED                      Draws a dotted line of LineWidth thickness.  
DASHED                      Draws a dashed line of LineWidth thickness.  
DOTDASH                      Draws a dot-dashed line of LineWidth thickness.

**Default**

SOLID

**LineWidth**

---

LineWidth specifies the width in pixels of the border to be drawn on a chart's hands.

**Values**

1 or greater                      Whole number width in pixels

**Default**

1

**Color**

---

Color specifies the color of the hand border.

**HandData**

---

---

```
HandData[N] = (Name, Value, Length), ...;
```

HandData identifies and describes hands in dial charts.

**Used in These Charts**

Dial

**Example:**

```
HandData = ("Hour Hand", 3.5, 68), ("Minute Hand", 30, 85), ("Second Hand", 53, 95);
```

**Attributes**

Name	Value	Length
------	-------	--------

**Value**

Value gives the value of the hand relative to the dial's range.

**Example:**

```
HandData = ("Hour Hand", 3.5, 68), ("Minute Hand", 30, 85), ("Second Hand", 53, 95);
```

**Values**

Real numbers

**Default**

None

**Length**

Length gives the length of the hand as a percentage of the dial's radius.

**Example:**

```
HandData = ("Hour Hand", 3.5, 68), ("Minute Hand", 30, 85), ("Second Hand", 53, 95);
```

**Values**

Real numbers

**Default**

None

**HandDelete**

```
HandDelete[N] = (Name|ALL), ...;
```

HandDelete is used to delete a specific hand, or all hands, in a dial chart.

**Used in These Charts**

Dial

**Example:**

```
HandDelete = ("Hour"), ("Minute");
HandDelete = ALL;
```

**Attributes**

(Switch)

**Switch**

This switch allows you to either name the hands that are to be deleted, or to delete all at once.

**Example:**

```
HandDelete = ("Pressure"), ("PressureChange");  
HandDelete = ALL;
```

**Values**

Name	A string that names a hand
ALL	All the hands

**Default**

No defaults

---

## HandDrag

---

```
HandDrag[N] = "ON"/"OFF";
```

The `HandDrag` switch is used to allow or stop the user from dragging the hands of the dial chart with the mouse.

**Used in These Charts**

Dial

**Example:**

```
HandDrag = "ON";  
HandDrag = "OFF";
```

**Attributes**

(Switch)

**Switch**

---

This switch sets the on/off state.

**Example:**

```
HandDrag = "ON";  
HandDrag = "OFF";
```

**Values**

ON	Allows the user to drag the hands on the applet dial
OFF	Stops the user from dragging the hands on the applet dial

**Default**

OFF

---

## HandDropShadow

---

```
HandDropShadow = (color, offsetx, offsety, size);
```

`HandDropShadow` places a shadow on the background of a hand in a dial chart. The color, orientation, and size of the shadow can be defined. The tuple element `color` sets the color of the shadow. That color value is interpolated to complete transparency as it reaches the end of the shadow's blur area. `offsetx` and `offsety` define the center point of the shadow; `offsetx` sets the x-axis offset from the chart's center-point; `offsety` sets the y-axis offset. When an offset attribute is set to a whole number value, the position of the drop shadow is offset from the chart's center point by the number of pixels set by that whole number. When an offset is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow (the set of bars). The center of the drop shadow is repositioned based upon the values or percentages set for `offsetx` and `offsety`. Offset attribute values may be positive or negative. `size` sets the size of the blur area, the region beyond the actual drop shadow shape where the shadow is extended and blurred into transparency. The size of this blurred region is controlled by the `size` attribute. The blurred region becomes larger and more diffuse as the value of `size` is increased. When `size` is set to a whole number value, the size of the blurred area is defined in pixels. When `size` is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow. When using a fractional value, enclose the value in double-quotes to "escape" the decimal point.

### *Used in These Charts*

Dial

### *Example:*

```
HandDropShadow = (color, offsetx, offsety, size);
```

### *Attributes*

Color	Offsetx	offsety	Size
-------	---------	---------	------

---

### *Color*

`Color` specifies the base color of the shadow to be drawn behind a chart's bars.

---

### *Offsetx*

`Offsetx` specifies the x-coordinate offset from center.

---

### *Offsety*

`Offsety` specifies the y-coordinate offset from center.

---

### *Size*

---

`size` specifies the width of the blurred area.

## HandButtonEdgeHighlights

---

```
HandButtonEdgeHighlights = (start,stop,gap,size), ...;
```

The `HandButtonEdgeHighlights` parameter provides a visual pattern fill in a Dial chart which accents the center button. It overlays a ring (annulus) fill pattern over the existing fill patterns in a specified zone along the interior edge of the center button. The gap between the sides of the center button and the fill pattern being applied can be modified. The element `start` sets the beginning color of the gradient, associated with the outside edge; the element `stop` sets the end color of the gradient, associated with the interior of the center button where the color blends to transparency. Color values are interpolated between the two. The element `size` specifies the width of the highlight. The element `gap` specifies the size of the gap between the edge of the highlight and the edge of the center button. When the value for `gap` is specified as a whole number, it sets the distance between the edge of the highlight and the edge of the center button in pixels. When set to a fractional number between 0 and 1, it sets the gap to a percentage of the radius of the pie. Percentage values are written using a decimal point (e.g. "0.05" and "0.50" represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to "escape" the decimal point.

### Used in These Charts

Dial

### Example:

```
HandButtonEdgeHighlights = (blue_25,white_75,1,25), ...;
```

### Attributes

<code>start</code>	<code>stop</code>	<code>gap</code>	<code>size</code>
--------------------	-------------------	------------------	-------------------

## Hands

---

```
Hands [N] = (Name, TipColor, ShaftColor, DialName, HandLabel), ...;
```

The `Hands` parameter names the hands of a dial chart, colors them, and links them to a dial.

### Used in These Charts

Dial

### Example:

```
Hands = ("Hour Hand",black,black,"Hour Dial","Hours"),
        ("Minute Hand",black,black,"Minute Dial"),
        ("Second Hand",xAc0000,xAc0000,"Second Dial");
```

### Attributes

<code>Name</code>	<code>TipColor</code>	<code>ShaftColor</code>	<code>DialName</code>
-------------------	-----------------------	-------------------------	-----------------------

---

## *TipColor*

---

TipColor controls the color of the hand's tip.

### *Example:*

```
Hands = ("Hour Hand",black,black,"Hour Dial"),
        ("Minute Hand",black,black,"Minute Dial"),
        ("Second Hand",xAC0000,xAC0000,"Second Dial");
```

### *Values*

Use values for the regular Color attribute as described in Chapter 8.

### *Default*

black

---

## *ShaftColor*

---

ShaftColor controls the color of the hand's shaft.

### *Example:*

```
Hands = ("Hour Hand",black,black,"Hour Dial"),
        ("Minute Hand",black,black,"Minute Dial"),
        ("Second Hand",xAC0000,xAC0000,"Second Dial");
```

### *Values*

Use values for the regular Color attribute as described in Chapter 8.

### *Default*

black

---

## *DialName*

---

DialName specifies which dial within the DialChart to which the hand is associated. (There can be more than one dial, and they can overlap physically.)

### *Example:*

```
Hands = ("Hour Hand",black,black,"Hour Dial"),
        ("Minute Hand",black,black,"Minute Dial"),
        ("Second Hand",xAC0000,xAC0000,"Second Dial");
```

### *Values*

Use values for the regular Name attribute as described in Chapter 8.

### *Default*

None given

---

## *HandLabel*

---

HandLabel specifies an optional text label to display along with the hand. The label's appearance is controlled by the HandLabels parameter.



## HandLabels

---

```
HandLabels = ("Name", ON|OFF, labelpos, Color, "FontName", FontSize, Angle, interiorAlignment);
```

HandLabels controls the appearance of text labels on the hands of a DialChart.

### Used in These Charts

Dial

### Example:

```
HandLabels = ("Goal", ON, 1.1, red, "Helvetica", 12, 0),
             ("Actual", ON, 1.1, black, "Helvetica", 12, 0);
```

### Attributes

Name	(Switch)	LabelPos	Color	FontName
FontSize	Angle	interiorAlignment		

### Switch

---

This switch turns the hand labels' visibility on and off.

### Values

ON	Show the hand label
OFF	Hide the hand label

### Default

None

### InteriorAlignment

---

The LabelPos attribute specifies the alignment to use in hand labels that contain multiple lines. The legal values for interiorAlignment are LEFT, RIGHT, or CENTER.

### LabelPos

---

The LabelPos attribute in the dial chart works the same way the LabelPos parameter does for the pie chart: it defines the position of the tic mark labels relative to the width of the dial. A value of 1.1 will place each label just outside the dial face, while a value of 0.6 will place each label inside of the dial face. For instance, clocks are often made with their tic mark labels on the outside of the dial face, but gauges tend to have them just on the inside of the dial face along with the tic marks.

### Values

Positive real numbers, generally between 0 and 2.

### Default

None

## HandStyles

---

```
HandStyles[N] = (Name, NEEDLELINE|NEEDLEFILL|NEEDLEBUTTON|SHARP|ROUND|BLOCK|LINE|NONE,
TipWidth, ShaftWidth), ...;
```

The `HandStyles` parameter describes the appearance of the hands in a dial chart.

### Used in These Charts

Dial

#### Example:

```
HandStyles = ("Hour Hand", BLOCK, 8, 4), ("Minute Hand", BLOCK, 6, 3),
("Second Hand", NONE, 4, 2);
```

#### Attributes

<code>Name</code>	(Switch)	<code>TipWidth</code>	<code>ShaftWidth</code>
-------------------	----------	-----------------------	-------------------------

### Switch

---

This switch identifies the style of the tip of a hand in a dial chart.

#### Example:

```
HandStyles = ("Hour Hand", BLOCK, 8, 4), ("Minute Hand", BLOCK, 6, 3),
("Second Hand", NONE, 4, 2);
```

#### Values

SHARP	Triangular tip with sharper corners than BLOCK
ROUND	Circular tip
BLOCK	Triangular tip
LINE	Plain line tip
NONE	Hand will not be drawn

#### Default

SHARP

## Header

---

```
Header = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment, extend);
```

`Header`, which is universally available in Visual Mining chart applications, describes an optional title, or label, that sits at the visual top of a chart, or its “head,” and uses standard attributes for string text, text color, font, font size, and label rotation. Note that the `Header` is displayed as centered across the entire applet or graphic space, not across the grid of the chart. As this can sometimes look awkward, we recommend using the `TopAxisTitle` parameter instead if you want a header centered across the chart instead.

`Header` uses a standard CDL label grouping, or “tuple,” to describe the typography of this label.

***Used in These Charts***

All

***Example:***

```
Header = ("Jets Per Minute", crimson, Helvetica, 12, 0, LEFT, CENTER, OFF);
```

interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
exteriorAlignment	Specifies the alignment for the entire Title object.
extend	Specified background should extend entire length of chart.

The legal values for `interiorAlignment` and `exteriorAlignment` are LEFT, RIGHT, or CENTER. The legal values for `extend` are ON and OFF.

***Attributes***

*Label Color FontName FontSize Angle interiorAlignment exteriorAlignment*

**HeaderActiveLabel**

```
HeaderActiveLabel = ("Label", "URL", "Target");
```

`HeaderActiveLabel` defines a single active label destination for the header title.

***Used in These Charts***

All

***Example:***

```
HeaderActiveLabels = ("Destination", "demo.html", "frame1");
```

***Attributes***

*Label URL Target*

**HeaderBox**

```
HeaderBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `HeaderBox` specifies a background region just for the `Header` title.

***Used in These Charts***

All

***Attributes***

*BorderColor BorderType BorderWidth Color  
ImageFormat ImageURL*

## HistogramBin

---

```
HistogramBin = (HistogramBinType,HistogramBinSize);
```

HistogramBin describes the type and number of bins in a histogram.

### *Used in These Charts*

Histogram

### *Example:*

```
HistogramBin = (BYNUMBER,3);
```

### *Attributes*

*HistogramBinType*      *HistogramBinSize*

### *HistogramBinType*

---

String content describing the distribution type. Legal values are AUTO, BYNUMBER, BYWIDTH. The default value is AUTO.

AUTO allows the histogram to automatically define the bins.

BYNUMBER allows users to specify the number of bins.

BYWIDTH allows the user to specify the width of a bin.

### *HistogramBinSize*

---

Integer specifying the histogram bin size. If HistogramBinType is AUTO, this field is ignored. If HistogramBinType is BYNUMBER, this field specifies the number of bins. If HistogramBinType is BYWIDTH this field specifies the width of each bin.

## HistogramScale

---

```
HistogramScale = (HistogramMinValue,HistogramMaxValue);
```

HistogramScale defines the upper and lower limits of the data range of a histogram.

### *Used in These Charts*

Histogram

### *Example:*

```
HistogramScale = (0,1000);
```

### *Attributes*

*HistogramMinValue*      *HistogramMaxValue*

### *HistogramMinValue*

---

Number specifying the lower value for the data range of the histogram. Data points below the minimum will be placed in the first bin.

### *HistogramMaxValue*

---

Number specifying the upper value for the data range of the histogram. Data points above the maximum will be placed in the last bin.

## HistogramType

---

```
HistogramType = HistogramType;
```

HistogramType describes the type of data distribution in a histogram.

### *Used in These Charts*

Histogram

### *Example:*

```
HistogramType = BYNUMBER;
```

### *Attributes*

*HistogramType*

### *HistogramType*

---

String content describing the distribution type. Legal values are BYNUMBER, PERCENTAGE, and PROBABILITY. The default is BYNUMBER.

BYNUMBER specifies that a bin will hold a count of the number of data points in the bin's range.

PERCENTAGE specifies that a bin will hold the percentage of the total data points that are in the bin's range.

PROBABILITY specifies that a bin will hold the probability that any given point is in the bin's range.

## LabelAnimationStyle

---

```
LabelAnimationStyle = FADE | NONE
```

Defines how axis and data labels initially appear in a chart. This parameter is only valid in SVG and SVGweb output formats.

### *Example:*

```
LabelAnimationStyle = FADE;
```

### Values

FADE	The labels fade in.
NONE	The labels are immediately visible.

### Default

NONE

---

## Labels

---

```
TopLabels[N] = "Label1", "Label2", ...;  
BottomLabels[N] = "Label1", "Label2", ...;  
LeftLabels[N] = "Label1", "Label2", ...;  
RightLabels[N] = "Label1", "Label2", ...;
```

The `Labels` parameter specifies a list of custom tic mark labels that will be used instead of the numeric labels automatically generated by the axis. The `Labels` will be evenly placed along the axis, overriding any tic placement specified by the `StepValue` attribute. If a corresponding `TicLocations` parameter is specified, then the labels will be drawn in order at the locations specified in `TicLocations[N]`.

In a Bar, Combo, Pareto or Stock Chart, the `BarLabels` parameter overrides the `LeftLabels` (for horizontal bars) parameters.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
LeftLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie",  
            "Steve", "Paul";
```

### Attributes

`Label`

---

## LabelPos

---

```
LabelPos = Float;
```

Defines the position of a pie slice label relative to the width of the pie. A value of 1.1 will place each label just outside the pie, while a value of 0.6 will place each label inside of each pie slice.

### Used in These Charts

Pie

### Example:

```
LabelPos = 1.1;
```

### Attributes

`Float`

## Float

---

Float is a number indicating slice label placement, where 1.0 is on the outside edge of the pie.

### Example:

```
LabelPos      = 1.1;
LabelPos      = 0.6;
```

### Values

< 1.0 Places the labels inside the pie border.  
> 1.0 Places the labels outside the pie border

### Default

None

## Layout

---

```
Layout = (LayoutType);
```

The Layout parameter is used to specify the layout format for all of the charts defined in the Charts parameter. The examples below show Layout used in context.

### Used in These Charts

Multi-chart

### Example:

```
Layout = ROWS;                                <!-- charts laid out in rows -->
Charts = ("Sales",PIECHART),
        ("Expense",BARCHART),
        ("Bonus",XYCHART),
        (BREAK),
        ("Growth",COMBOCHART),
        (BREAK),
        ("Budget",XYCHART),
        ("Salary",PIECHART);
```

```
<PARAM NAME=Layout VALUE="(ROWS)">
<PARAM NAME = ChartScript1 VALUE='
Background = (white, NONE);
Header = ("1. Most Requested Pages", black, Helvetica, 12);
DwellLabel = ("", black, "Helvetica", 9);
DwellBox = (xe3e3e3, SHADOW, 2);
ColorTable = xB5D5F0, xBEA9AD, xDACE98, xEBF0F3, xAABAC5, xBFC1A0;
BottomTicks = ("ON", black, "Helvetica", 9);
LeftTicks = ("ON", black, "Helvetica", 9);
LeftScale = (0, 2700);
LeftFormat = (INTEGER);
BarLabels = "Home\nPage", "NetCharts", "Examples", "Products";
GraphType = GROUP;
DataSets = ("Server1", NULL);
```

```

DataSet1      = 2694, 780, 628, 513;
Bar3DDepth   = 3;
'>

```

### Attributes

LayoutType

### LayoutType

---

LayoutType can be either ROWS or COLS, specifying that the charts should be arranged in rows or columns, respectively. The default layout is COLS.

#### Example:

```

Layout = ROWS;                                <!-- charts laid out in rows -->
Charts = ("Sales",PIECHART),
         ("Expense",BARCHART),
         ("Bonus",XYCHART),
         (BREAK),
         ("Growth",COMBOCHART),
         (BREAK),
         ("Budget",XYCHART),
         ("Salary",PIECHART);

<PARAM NAME=Layout VALUE="(ROWS)">
<PARAM NAME = ChartScript1 VALUE='
Background   = (white, NONE);
Header       = ("1. Most Requested Pages", black, Helvetica, 12);
DwellLabel   = ("", black, "Helvetica", 9);
DwellBox     = (xe3e3e3, SHADOW, 2);
ColorTable   = xB5D5F0, xBEA9AD, xDACE98, xEBF0F3, xAABAC5, xBFC1A0;
BottomTicks  = ("ON", black, "Helvetica", 9);
LeftTicks    = ("ON", black, "Helvetica", 9);
LeftScale    = (0, 2700);
LeftFormat   = (INTEGER);
BarLabels    = "Home\nPage", "NetCharts", "Examples", "Products";
GraphType    = GROUP;
DataSets     = ("Server1", NULL);
DataSet1     = 2694, 780, 628, 513;
Bar3DDepth   = 3;
'>

```

#### Values

ROWS	Charts will be arranged in rows
COLS	Charts will be arranged in columns

#### Default

COLS

### LeftActiveLabels

---

```
LeftActiveLabels = ("Label", "URL", "Target"),...;
```

The left axis labels become active labels when LeftActiveLabels is used. Each set in parenthesis has a corresponding set within a DataSet parameter.



**Used in These Charts**

All

**Attributes**

*Label*                      *URL*                      *Target*

**LeftAxis**


---

```
LeftAxis = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
```

If `LeftAxis` is defined for a Combo chart, then the top axis will be used to map the X data values for all line sets, unless otherwise specified using the `LineAxis` parameter. The group sets the typographic characteristics for the data values.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

**Used in These Charts**

Combo

**Example:**

```
LeftAxis = ("", black, "TimesRoman", 16, 0);
```

**Attributes**

*Label*      *Color*              *FontName*      *FontSize*      *Angle*      *interiorAlignment*

**LeftAxisTitle**


---

```
LeftAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);
```

The `LeftAxisTitle` parameter specifies the label attributes for the axis title, which centered along the top axis, just above the grid. When the `Header` parameter, whether because of the use of a legend, or for some other reason, creates a title that seems visually unbalanced, you may find this parameter produces a more pleasing chart title.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.

The legal values for `interiorAlignment` and `exteriorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
LeftAxisTitle = ("Ceres Prototype Project Schedule\n ", black, "Helvetica",
12);
```

**Attributes**

*Label Color FontName FontSize Angle interiorAlignment exteriorAlignment*

**LeftAxisTitleActiveLabel**

```
LeftAxisTitleActiveLabel = ("Label", "URL", "Target");
```

`LeftAxisTitleActiveLabel` defines a single active label destination for the `LeftAxisTitle` parameter.

**Used in These Charts**

All

**Example:**

```
LeftAxisTitleActiveLabel = ("Destination", "demo.html", "frame1");
```

**Attributes**

*Label URL Target*

**LeftAxisTitleBox**

```
LeftAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor,
TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `LeftAxisTitleBox` parameter specifies the region attributes for the axis title centered along the axis. The image-related attributes need not be used unless you want to use an image texture on the box.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
LeftAxisTitleBox = (lightgray, SHADOW, 3,,gray);
```

**Attributes**

*Color BorderType BorderWidth ImageURL ImageFormat  
BorderColor TRCornerStyle BRCornerStyle BLCornerStyle CornerColor*

**XXCornerStyle**

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

---

## LeftColor

---

```
LeftColor = Color;
```

`LeftColor` controls the color of the top axis and the tic marks, but not the tic mark labels. The default axis color is black. If the `NULL` color is specified, the axis color is not changed by this parameter.

### Example:

```
LeftAxisColor = xB5D5F0;
```

### Attributes

`Color`

---

## LeftDrawMinorTics

---

```
LeftDrawMinorTics = ON/OFF;
```

`LeftDrawMinorTics` controls whether or not left tics are drawn. The default value is ON.

### Example:

```
LeftDrawMinorTics = OFF;
```

### Attributes

(Switch)

---

## LeftFormat

---

```
LeftFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
```

`LeftFormat` adjusts the numeric labels that are automatically generated for the top axis, should one be defined.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
LeftFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
LeftFormat = (DATE, "%M/%D", "1 Apr 96", "1d");  
LeftFormat = (INTEGER);  
LeftFormat = (FLOAT, "$%,9.2f", ,);
```

### Attributes

`FormatType`      `FormatExpr`      `TimeBase`      `TimeUnit`

## FormatType

FormatType specifies the type of number being displayed on the top axis.

### Example:

```
LeftFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
LeftFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
LeftFormat = (INTEGER);
LeftFormat = (FLOAT, "$%,9.2f", ,);
```

### Values

DATE	Axis value are shown as date and/or time values. See <b>Appendix A: Date and Time Values</b> for further detail.
FLOAT	Axis values are shown with decimal parts.
INTEGER	Axis values are shown as integers, and are rounded if necessary.
DECIMAL	Axis values are shown as decimals. See <a href="http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html">http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html</a> for more information.

The characters listed here are used in non-localized patterns. Localized patterns use the corresponding characters taken from this formatter's `DecimalFormatSymbols` object instead, and these characters lose their special status. Two exceptions are the currency sign and quote, which are not localized.

Symbol	Location	Localized?	Meaning
0	Number	Y	Digit
#	Number	Y	Digit, zero shows as absent
.	Number	Y	Decimal separator or monetary decimal separator
-	Number	Y	Minus sign
,	Number	Y	Grouping separator
E	Number	Y	Separates mantissa and exponent in scientific notation. <i>Need not be quoted in prefix or suffix.</i>
;	Subpattern boundary	Y	Separates positive and negative subpatterns
%	Prefix or suffix	Y	Multiply by 100 and show as percentage
\u2030	Prefix or suffix	Y	Multiply by 1000 and show as per mille
¤ (\u00A4)	Prefix or suffix	N	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.
'	Prefix or suffix	N	Used to quote special characters in a prefix or suffix, for example, "'#'" formats 123 to "#123". To create a single quote itself, use two in a row: "' o' 'clock".

### Default

INTEGER

## *TimeBase*

---

The `TimeBase` attribute specifies the base date to be used when determining the actual date or time value when using a time unit or numeric value. See **Appendix A: Date and Time Values** for further detail.

### *Example:*

```
LeftFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
LeftFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

### *Values*

String values representing dates or times

### *Default*

None

## *TimeUnit*

---

The `TimeUnit` attribute controls the time multiplier to be used when determining the actual date/time value when using a numeric value. See **Appendix A: Date and Time Values** for further detail.

### *Example:*

```
LeftFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
LeftFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

### *Values*

String values representing dates or times

### *Default*

None

## **LeftLabels**

---

```
LeftLabels = "Label1", "Label2", ...;
```

The `LeftLabels` parameter specifies a list of custom tic mark labels that will be used instead of the numeric labels automatically generated by the axis. The `LeftLabels` will be evenly placed along the axis, overriding any tic placement specified by the `StepValue` attribute.

In a Bar, Combo, Pareto, or Stock Chart, the `BarLabels` parameter overrides the `LeftLabels` (for horizontal bars) parameters.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
LeftLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie",  
            "Steve", "Paul";
```

**Attributes**

Label

---

**LeftMargins**

---

```
LeftMargins = (BottomSideMargin, TopSideMargin);
```

The `LeftMargins` parameter specifies the gap, in pixels, at the beginning and end of the left axis. Most often used to prevent clipping of data points at the extreme ends of the scale.

**Example:**

```
LeftMargins = (20, 20);
```

---

**LeftScroll**

---

```
LeftScroll = (ScrollMin, ScrollMax);
```

The `LeftScroll` parameter specifies a range of values through which the bottom axis can be scrolled. When the `ScrollMin` and `ScrollMax` attributes are defined for the axis, the axis will be displayed as a slider bar, using the axis color defined, with a white background. The relative size of the slider represents the percentage of the entire range currently being displayed. That is, it graphically depicts the size of the current axis range (`MinValue` and `MaxValue` attributes) relative to the scrollable region (`ScrollMin` and `ScrollMax` attributes). See the `LeftScale` parameter for `MinValue` and `MaxValue` definitions.

`LeftScroll` should only be used in conjunction the `LeftScale` parameter.

**Example:**

```
LeftScroll = (0, 98);
```

**Attributes**

ScrollMin                      ScrollMax

---

**ScrollMin**

---

`ScrollMin` sets the lower visible limit for a scrollbar defined with `LeftScroll`

**Example:**

```
TopScroll = (0, 98);
```

**Values**

<MinValue

**Default**

None

## *ScrollMax*

---

ScrollMax sets the upper visible limit for a scrollbar defined with BottomScroll

### **Example:**

```
LeftScroll = (0, 98);
```

### **Values**

> MaxValue

### **Default**

None

## **LeftTicLength**

---

```
LeftTicLength = Number;
```

The LeftTicLength parameter defines the size of axis tic marks which are displayed along the left axis of a chart. The parameter will reset the automatically generated tic length. The value defines the number of pixels to use for the length of the tic mark. By default, the number of pixels used is the width of the character zero (0) as found in the font applied to the label. Setting the LeftTicLength to the value -1 will cause the default size to be used.

### **Attributes**

Number

### **Number**

---

Apparent length of a left axis tic mark in a chart, in pixels.

### **Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### **Example:**

```
LeftTicLength = 10;
```

### **Values**

0	No effect (zero length tics are not drawn).
1 or greater	Whole number length in pixels

### **Default**

-1

## LeftTitle

---

```
LeftTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);
```

`LeftTitle` describes an optional title, or label, that sits on the left side of a chart, and uses standard attributes for string text, text color, font, font size, and label rotation. As with `Header`, `LeftTitle` is universally available in Visual Mining chart applications.

### Used in These Charts

All, but most useful in Pie, Diagram, and Dial, which do not have axes.

### Example:

```
LeftTitle = ("Financial Status", royalblue, Helvetica, 14, 0, LEFT);
```

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.
<code>extend</code>	Specified background should extend entire length of chart.

The legal values for `interiorAlignment` and `exteriorAlignment` are `LEFT`, `RIGHT`, or `CENTER`. The legal values for `extend` are `ON` and `OFF`.

### Attributes

*Label Color FontName FontSize Angle interiorAlignment exteriorAlignment*

## LeftTitleActiveLabel

---

```
LeftTitleActiveLabel = ("Label", "URL", "Target");
```

`LeftTitleActiveLabel` defines a single active label destination for the `LeftTitle` parameter.

### Used in These Charts

All

### Example:

```
LeftActiveLabel = ("Destination", "demo.html", "frame1");
```

### Attributes

*Label URL Target*

## LeftTitleBox

---

```
LeftTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `LeftTitleBox` specifies a background region just for the `LeftTitle` parameter.



**Used in These Charts**

All

**Attributes**

<i>BorderColor</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>Color</i>
<i>ImageFormat</i>	<i>ImageURL</i>	<i>TRCornerStyle</i>	<i>BRCornerStyle</i>
<i>BLCornerStyle</i>	<i>CornerColor</i>		

**XXCornerStyle**

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

**LeftTics**

```
LeftTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);
```

The `LeftTics` parameter specifies the label attributes for the tic marks displayed along the left axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the `LeftTics` parameter. If any attribute is not defined, any previous value of that attribute will be used.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>backgroundColor</code>	Background color of the tic label area
<code>rotationPoint</code>	For rotated axis labels, anchor point for the rotation

The legal values for `interiorAlignment` are LEFT, RIGHT, or CENTER.

The legal values for `rotationPoint` are LEFT, RIGHT.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
LeftTics = ("OFF", black, "Helvetica", 10,LEFT,,RIGHT);
```

**Attributes**

<i>Mode</i>	<i>Color</i>	<i>FontName</i>	<i>FontSize</i>	<i>Angle</i>	<i>interiorAlignment</i>
<i>InteriorAlignment</i>	<i>BackgroundColor</i>	<i>RotationPoint</i>			

**Mode**

`Mode` determines whether or not the tic labels are shown on that axis.



### Attributes

*Label*                      *URL*                      *Target*

## LegendAnimationStyle

---

```
LegendAnimationStyle = FADE | NONE
```

Defines how the legend initially appears in a chart. This parameter is only valid in SVG or SVGWeb output formats.

### Example:

```
LegendAnimationStyle = FADE;
```

### Values

FADE                      The legend fades in.

NONE                      The legend is immediately visible.

### Default

NONE

## LegendAxis

---

```
LegendAxis = (XAxis, Yaxis);
```

The `LegendAxis` parameter is optional and is only used if the `LegendLayout` location parameter is `CENTER` or the `LegendBoxSize` parameter is used. This parameter defines the types of coordinates used for the X and Y attributes in the `LegendLayout` parameter and for the `MaxWidth` and `MaxHeight` attributes in the `LegendBoxSize` parameter.

### Used in These Charts

All

### Example:

```
LegendAxis = (Bottom, Right);
```

### Attributes

XAxis                      Yaxis

### XAxis

---

Specifies which side of a `Legend` becomes the X axis.

### Example:

```
LegendAxis = (Bottom, Right);
```

**Values**

Percent	Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.
Pixel	Raw pixel coordinates
BOTTOM	Map values with the bottom axis of the chart
TOP	Map values with the top axis of the chart

**Default**

PERCENT

**YAxis**

Specifies which side of a `Legend` becomes the Y axis.

**Example:**

```
LegendAxis = (Bottom, Right);
```

**Values**

Percent	Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.
Pixel	Raw pixel coordinates
LEFT	Map values with the left side of the chart
RIGHT	Map values with the right side of the chart

**Default**

PERCENT

**LegendBox**

```
LegendBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle,
BRCornerStyle, BLCornerStyle, CornerColor);
```

The `LegendBox` specifies a background region just for the `Legend` parameter, and is optional, but may only appear with the `Legend` parameter. If specified, it defines a box to be displayed as a background for the `Legend`. The box will be automatically scaled to fit all the legend items, unless otherwise restricted by the `LegendBoxSize` parameter.

**Used in These Charts**

All

**Example:**

```
LegendBox = (yellow, SHADOW, 5);
```

**Attributes**

<i>BorderColor</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>Color</i>	<i>ImageFormat</i>
<i>ImageURL</i>				

---

## LegendBoxSize

---

```
LegendBoxSize = (MaxWidth, MaxHeight);
```

The `LegendBoxSize` parameter is optional, but may only appear with the `Legend` parameter. `LegendBoxSize` is used to specify maximum size values for the legend. This is useful in cases where the legend text must be automatically clipped.

### *Used in These Charts*

All

### *Example:*

```
LegendBoxSize = (100, 150);
```

### *Attributes*

MaxHeight                      MaxWidth

### *MaxHeight*

---

`MaxHeight` only limits the height of the legend if it grows too large; it does not explicitly set the height of the legend. If specified, this attribute is also used when automatically determining the number of columns for the legend.

### *Example:*

```
LegendBoxSize = (100, 150);
```

### *Values*

Maximum allowable legend box height in pixels

### *Default*

0

### *MaxWidth*

---

`MaxWidth` only limits the width of the legend if it grows too large; it does not explicitly set the width of the legend. If specified, this attribute is also used when automatically determining the number of columns for the legend.

### *Example:*

```
LegendBoxSize = (100, 150);
```

### *Values*

Maximum allowable legend box width in pixels

### *Default*

0

---

## LegendDwellAnimationHighlightBorderStyle

---

```
LegendDwellAnimationHighlightBorderStyle = (lineType, LineWidth, lineColor);
```

Defines the border style to be applied to data points in a series when LegendDwellAnimationStyle = HIGHLIGHT.

### *Used in These Charts*

All (except 3D, Heatmap, and Diagram)

### *Example:*

```
LegendDwellAnimationHighlightBorderStyle = (SOLID,1,BLACK);
```

### *Attributes*

<i>LineType</i>	<i>LineWidth</i>	<i>LineColor</i>
-----------------	------------------	------------------

---

### *LineType*

#### *Values*

SOLID	A solid line is displayed.
DOTTED	A dotted line is displayed.
DASHED	A dashed line is displayed.
DOTDASH	Alternating dots and dashes are displayed.

#### *Default*

SOLID

---

### *LineWidth*

#### *Values*

Width of the line in pixels

#### *Default*

1

---

### *LineColor*

#### *Default*

Black

---

## LegendDwellAnimationHighlightFill

---

```
LegendDwellAnimationHighlightFill = Color;
```

Defines the color used to fill data points in a series when `LegendDwellAnimationStyle = HIGHLIGHT`.

### *Used in These Charts*

All (except 3D, Heatmap, and Diagram)

### *Example:*

```
LegendDwellAnimationHighlightFill = BLUE);
```

### *Attributes*

Color

### *Default*

NONE

## LegendDwellAnimationStyle

---

```
LegendDwellAnimationStyle = HIGHLIGHT | NONE
```

Defines how the chart behaves when the mouse dwells over a legend entry.

### *Example:*

```
LegendDwellAnimationStyle = HIGHLIGHT;
```

### *Values*

**HIGHLIGHT** All data points in the related series are highlighted using values specified in `LegendDwellAnimationHighlightFill` and `LegendDwellAnimationHighlightBorderStyle`.

**NONE** No highlight is applied to the data points.

### *Default*

NONE

## LegendItems

---

```
LegendItems = ("Label", Color1, SymType1, SymSize1, SymStyle1, LineType1, LineWidth1, LineColor1, PatternFill, color1, color2, imageURL, shadowwidth), . . . ;
```

The `LegendItems` parameter is optional, but may only appear with the `Legend` parameter. If specified, it defines one or more items to be included in the `Legend`, each with its own attributes within the parenthesized set.

### *Used in These Charts*

All

**Example:**

```
LegendItems = ("1st Data Set", peachpuff, DIAMOND, 5, FILLED, SOLID, 2,
               peachpuff),
              ("2nd Data Set", mintcream, DIAMOND, 5, FILLED, SOLID, 2,
               mintcream),
              ("3rd Data Set", plum, DIAMOND, 5, FILLED, SOLID, 2, plum);
```

**Attributes**

<i>Label</i>	<i>Color</i>	<i>SymType</i>	<i>SymSize</i>
<i>SymStyle</i>	<i>LineStyle</i>	<i>LineWidth</i>	<i>LineColor</i>

**SymSize**

*SymSize* indicates the size, in pixels, of the iconic symbol to use for this legend item.

**Example:**

```
LegendItems = ("1st Data Set", peachpuff, DIAMOND, 5, FILLED, SOLID, 2,
               peachpuff),
```

**Values**

Integer values of pixels

**Default**

0

**SymStyle**

*SymStyle* indicates the style, either filled or outlined, of the iconic symbol to use for this legend item.

**Example:**

```
LegendItems = ("1st Data Set", peachpuff, DIAMOND, 5, FILLED, SOLID, 2,
               peachpuff),
```

**Values**

FILLED  
OUTLINED

**Default**

FILLED

**SymType**

*SymType* indicates the kind of iconic symbol to use for this legend item.

**Example:**

```
LegendItems = ("1st Data Set", peachpuff, DIAMOND, 5, FILLED, SOLID, 2,
               peachpuff),
```



**Values**

NONE	No symbol is displayed.
CIRCLE	Displays circles
SQUARE	Displays squares
DIAMOND	Displays diamonds
CROSS	Displays crosses
TARGET	Displays targets (bulls-eye)
TRIANGLEDOWN	Displays downward pointing triangles
TRIANGLEUP	Displays upward pointing triangles

**Default**

SQUARE

**PatternFill**

The `PatternFill` attribute provides a visual pattern fill for a legend item. The `Color1` and `Color2` attributes provide colors to use in creating the pattern fill.

Type	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional <code>imageURL</code> element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

## LegendLayout

---

`LegendLayout = (Type, Location, X, Y, Justify, Columns);`

The `LegendLayout` parameter is optional, but may only appear with the `Legend` parameter. If specified, it defines the placement of the legend within the chart.

### Used in These Charts

All

### Example:

```
LegendLayout = (HORIZONTAL, BOTTOM, 0, 0, LEFT, 2);
```

### Attributes

Columns	Justify	Location	Type
X	Y		

### Columns

---

The `Type` and `Columns` attributes specify the way legend items are displayed within the legend box. In general, the `Columns` attribute specifies the desired number of text columns.

### Example:

```
LegendLayout = (HORIZONTAL, BOTTOM, 0, 0, LEFT, 2);
```

### Values

- < 0      The number of columns will be automatically determined in order to prevent the legend height from exceeding the height of the graph or the maximum height specified by the `LegendBoxSize` parameter.
- 0 or 1    A single column will be displayed.
- > 1      The specified number of columns will be displayed.

### Default

-1

### Justify

---

When using a `Location` value of `CENTER`, the `Justify` attribute determines where the legend box is located relative to the legend control point specified by `X` and `Y`, as shown below:

TOPLEFT	TOP	TOPRIGHT
LEFT	CENTER	RIGHT
BOTTOMLEFT	BOTTOM	BOTTOMRIGHT

For example, if `Justify` is set to `TOP`, then the legend will be displayed *below* the control point. That is, the top of the legend will be positioned at the control point. If, on the other hand, `BOTTOMRIGHT` is chosen, then the bottom right corner of the legend will be positioned at the control point.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

TOPLEFT	Displayed at the top of the panel and is left justified
TOP	Displayed at the top of the panel and is center justified
TOPRIGHT	Displayed at the top of the panel and is right justified
LEFT	Displayed at the left of the panel and is left justified
CENTER	X and Y attributes used to specify legend control point.
RIGHT	Displayed at the right of the panel and is right justified
BOTTOMLEFT	Displayed at the bottom of the panel and is left justified
BOTTOM	Displayed at the bottom of the panel and is center justified
BOTTOMRIGHT	Displayed at the bottom of the panel and is right justified

**Default**

CENTER

**Location**

The `Location` attribute defines the location of the legend relative to the graph. All of the location values, except for `CENTER`, specify fixed areas outside of the graph, and the size of the legend will affect the size of the graph. In those cases, the exact placement of the legend is completely controlled by the chart, with appropriate justification. For example, a `TOP` chart legend is displayed at the top of the panel and is center justified, while a `BOTTOMRIGHT` legend is displayed below the graph and is right justified.

If the `Location` is set to `CENTER`, then the `x`, `y` attributes are used to specify the location of the legend *control point*. The control point can be specified using a number of different coordinate types (see `LegendAxis` parameter for details) but in all cases, it represents an anchor position *within* the confines of the chart itself. That is, the control point is a location that lies inside of the axes. As such, any legend displayed using the `CENTER` location will be clipped to the borders of the chart.

When using the `CENTER` location, the `Justify` attribute determines where the legend box is located relative to the legend control point specified by `x` and `y`.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

TOPLEFT	Displayed at the top of the panel and is left justified
TOP	Displayed at the top of the panel and is center justified
TOPRIGHT	Displayed at the top of the panel and is right justified
LEFT	Displayed at the left of the panel and is left justified
CENTER	X and Y attributes used to specify legend control point.
RIGHT	Displayed at the right of the panel and is right justified
BOTTOMLEFT	Displayed at the bottom of the panel and is left justified
BOTTOM	Displayed at the bottom of the panel and is center justified
BOTTOMRIGHT	Displayed at the bottom of the panel and is right justified

**Default**

RIGHT

---

**Type**

---

The `Type` and `Columns` attributes specify the way legend items are displayed within the legend box. In general, the `Type` attribute specifies the desired orientation and the `Columns` attribute specifies the desired number of columns.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

**HORIZONTAL** The number of columns will be automatically determined in order to prevent the legend width from exceeding the width of the graph or the maximum width specified by the `LegendBoxSize` parameter.

If `Columns` is 0 or 1, a single column will be displayed.

**VERTICAL** The number of columns will be automatically determined in order to prevent the legend height from exceeding the height of the graph or the maximum height specified by the `LegendBoxSize` parameter.

If `Columns` is 0 or 1, a single row will be displayed.

**Default**

VERTICAL

---

**X**

---

Specifies the X position of the control point of the Legend.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

**Percent** Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.  
**Pixel** Raw pixel coordinates  
**BOTTOM** Map values with the bottom axis of the chart  
**TOP** Map values with the top axis of the chart

**Default**

50

---

**Y**

---

Specifies the Y position of the control point of the Legend

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

Percent	Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.
Pixel	Raw pixel coordinates
LEFT	Map values with the left side of the chart
RIGHT	Map values with the right side of the chart

**Default**

50

## Line3Ddepth

---

```
Line3Ddepth[N] = depth;
```

If `Line3DDepth` is specified, then the lines drawn on the chart can be “3D.” This parameter can be used with the `LineWidth` parameter to achieve visible separation between 3D line sets.

**Used in These Charts**

Combo, Line, Pareto, Stock, X-Y

**Example:**

```
Line3DDepth = 17;
```

**Attributes**

Depth

**Depth**

---

Depth indicates the width of the line along a Z-axis.

**Example:**

```
Line3DDepth = 17;
```

**Values**

>1	If depth is greater than 1, then all line sets will be displayed as 3D lines. The max amount of space allocated to each line in the Z dimension will be "depth" pixels.
1	Lines will not be seen as 3D

**Default**

1

## LineAnimationStyle

---

```
LineAnimationStyle = BEND | FADE | NONE
```

Specifies how lines initially appear in a line chart. This parameter is only valid in SVG or SVGWeb output formats.

## Attributes

Style

## Style

---

Style refers to the manner in which lines are first rendered in a line chart.

### Example:

```
LineStyle = BEND;
```

### Values

**BEND** The lines start as a straight line at zero and each point bends to its actual value.

**FADE** The lines fade in.

**NONE** The lines are immediately visible.

### Default

NONE

## LineAxis

---

```
LineAxis[N] = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
```

The `LineAxis` parameter defines a set of one or more axes for a line chart, and must correspond to a `DataSet`. The attributes define the specific axis to be used when mapping the X and Y values respectively for each data set defined. The `XAxis` attribute may be set to `BOTTOM` or `TOP`, while the `YAxis` may be set to `LEFT` or `RIGHT`.

### Used in These Charts

Combo, Line, Stock, X-Y

### Example:

```
LineAxis = (BOTTOM, LEFT), (BOTTOM, LEFT);
```

### Attributes

`XAxis`                      `YAxis`

## LineColorTable

---

```
LineColorTable[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

`LineColorTable` defines a set of colors for dataset N that overrides all other color specifications for that set. The parameters used for specifying the color of data points in a chart are (in ascending order of precedence) `ColorTable`, `LinesSets`, `LineStyle`, and `LineColorTable`. `LineColorTable` is used most frequently to color some specific data point.

For example

```
LineColorTable2 = ,,blue;
```

will change the third data point in the second series to blue, while all other datapoints in the chart continue to be colored by one of the other color related parameters.

The colors you can use are defined in the common `Color` attribute (Chapter 4) .

### *Used in These Charts*

Combo, Line, Stock, XY

### *Example:*

```
BarColorTable2 = , , red;
```

### *Attributes*

None

## LineDropShadow

---

```
LineDropShadow = (color, offsetx, offsety, size);
```

`LineDropShadow` places a shadow on the background field of the Combo, Line, Stock or XY chart. The color, orientation, and size of the shadow can be defined. The tuple element `color` sets the color of the shadow. That color value is interpolated to complete transparency as it reaches the end of the shadow's blur area. `offsetx` and `offsety` define the center point of the shadow; `offsetx` sets the x-axis offset from the chart's center-point; `offsety` sets the y-axis offset. When an offset attribute is set to a whole number value, the position of the drop shadow is offset from the chart's center point by the number of pixels set by that whole number. When an offset is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow. The center of the drop shadow is repositioned based upon the values or percentages set for `offsetx` and `offsety`. Offset attribute values may be positive or negative. `size` sets the size of the blur area, the region beyond the actual drop shadow shape where the shadow is extended and blurred into transparency. The size of this blurred region is controlled by the `size` attribute. The blurred region becomes larger and more diffuse as the value of `size` is increased. When `size` is set to a whole number value, the size of the blurred area is defined in pixels. When `size` is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow. When using a fractional value, enclose the value in double-quotes to "escape" the decimal point.

### *Used in These Charts*

Bubble, Combo, Line, Stock, X-Y

### *Example:*

```
LineDropShadow = (Black,5,5,"0.05");
```

### *Attributes*

Color	Offsetx	offsety	Size
-------	---------	---------	------

---

## Color

---

`Color` specifies the base color of the shadow to be drawn behind the chart.

### Example:

```
LineDropShadow = (black, "-.05", "-0.05", 55);
```

---

## Offsetx

---

`Offsetx` specifies the x-coordinate offset from center.

### Example:

```
LineDropShadow = (black, 25 -10, 25);
```

---

## Offsety

---

`Offsety` specifies the y-coordinate offset from center.

### Example:

```
LineDropShadow = (black, 25 -10, 25);
```

---

## Size

---

`Size` specifies the width of the blurred area.

### Example:

```
LineDropShadow = (black, "-.05", "-.05", 55);
```

---

## LineFillPattern

---

`LineFillPattern[N]` = (*type*, *color1*, *color2*, *imageUrl*), ...;

The `LineFillPattern` parameter provides a visual pattern fill for the line area of a chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type



	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENTRADIAL	radial style gradient
	GRADIENTCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENTCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

### Used in These Charts

Line

#### Example:

```
LineFillPattern = (VERTICAL,blue,white);
```

#### Attributes

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

## LineLabels[n]

```
LineLabels[1-50] = ("Label", "URL", "Target"), ...;
```

LineLabels defines sets of sets of active label destinations for the lines in a line graph, and maps to LineSets, which must also be specified.

### Used in These Charts

Combo, Stock

**Example:**

```
LineLabels1 = ("OUTLINE", "fr71h.html", "frame1"), ("OUTLINE", "fr72h.html",
"frame1"), ("OUTLINE", "fr73h.html", " frame1");
```

**Attributes**

Label	URL	Target
-------	-----	--------

**LineSet[n]**

```
LineSet[1-50] = y1, y2, y3, ...;
```

Defines a list of from 1 up to 50 Y values for each line set in a line chart, and corresponds to the `LineSets` parameter, with which it must be used. X values are not specified because the line chart style assumes automatic uniform spacing along the X Axis of the graph.

**Used in These Charts**

Combo, Line, Radar, Stock, X-Y

**Example:**

```
LineSets = ("Hammers", black), ("Clamps", black), ("Wrenches", black),
("Pliers", black);
LineSet1 = 40, -100, 80, 50, 40;
LineSet2 = 60, 50, 10, 40, 30;
LineSet3 = -50, 20, 49, 10, 89;
LineSet4 = 40, 50, 150, 150, 200;
```

**Attributes**

Y

**Y**

A real value, corresponding to a set within the `LineSet` parameter. Null values, which consist of no space between delimiting commas, may be assigned and are displayed differently depending on the `GraphType`.

**Example:**

```
LineSet1 = 10, 20, 30, 40;
LineSet2 = 60,,10,40,30;
```

**Values**

>= 0

**Default**

None

---

## LineSets

---

```
LineSets[N] = (Name1, SymColor1), (Name2, SymColor2), ...;
```

By default, the `Name` and `SymColor` assigned to each data set will be used in the chart legend. At most 50 line sets may be displayed.

### *Used in These Charts*

Combo, Line, Radar, Stock, X-Y

### *Example:*

```
LineSets = ("Sprocket", black),
           ("Actuator", black),
           ("Do-Hicky", black),
           ("Thingy", black);
```

### *Attributes*

Name	SymColor
------	----------

---

### *Name*

The `Name` of a `LineSet` is an identifier, which will only appear in the chart if a legend is used in the chart.

### *Example:*

```
LineSets = ("Sprocket", black),
           ("Actuator", black),
           ("Do-Hicky", black),
           ("Thingy", black);
```

### *Value*

Any legal string value

### *Default*

None

---

### *SymColor*

If a `SymColor` is not specified in the vector, then the previously specified color will be used. If the color is specified as `NULL`, then a default color will be chosen from the color table. `SymColor` is used in the definitions of a legend, and may be overridden by the specification of a color in the `LineStyle` parameter.

### *Example:*

```
LineSets = ("Sprocket", black),
           ("Actuator", purple),
           ("Do-Hicky", red),
           ("Thingy", orange);
```

### *Value*

Any legal color value (see `Color` attribute in Chapter 4).

**Default**

black

**LineStyle**

---

```
LineStyle = (Type, LineWidth, Color, FillColor, LineType, LineType, FillType), ... ;
```

This parameter specifies the line style to be displayed for one or more line sets.

**Used in These Charts**

Bubble, Combo, Line, Radar, Stock, Strip, X-Y

**Example:**

```
LineStyle =(SOLID, 3, blue, blue, NORMAL, OFF),
           (SOLID, 2, red, red, FIT, COLOR);
```

**Attributes**

Color	FillColor	LineType	LineWidth	Type
-------	-----------	----------	-----------	------

**FillColor**

---

If this attribute is not NULL, then the area under the line set will be filled with the given color.

**Example:**

```
LineStyle =(SOLID, 3, skyblue, lightskyblue),
           (SOLID, 3, red, pink),           <!-- red line with pink fill -->
           (SOLID, 3, green, lightgreen),
           (SOLID, 3, orange, peachpuff);
```

**Values**

NULL	Also, value left unspecified: No color fills the area under the line.
Any legal color	Area under the line is filled. See Chapter 4 for the <code>Color</code> attribute.

**Default**

None

**Type**

---

The style of line to draw

**Values**

NONE  
SOLID  
DOTTED  
DASHED  
DOTDASH

**Default**

SOLID

## LineType

---

The type of line to use to connect the points in the series

### Values

NORMAL  
FIT  
BOTH

### Default

NORMAL

## FillType

---

The type of fill to use for the series. ON will force a fill, using a color from the Color Table if a fill color is not defined. OFF will not fill, even if a fill color is specified, COLOR will fill only if a fill color is specified.

### Values

ON  
OFF  
COLOR

### Default

COLOR

## LineSymbol

---

```
LineSymbol[N] = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor,ShadowWidth), ...;
```

LineSymbol specifies the symbols to be displayed for one or more line sets. That is, the first parenthesized group defines the symbol for the first line set, and so on.

### Used in These Charts

Combo, Line, Radar, Stock, Strip, X-Y

### Example:

```
LineSymbol = (IMAGE,,,,,"$SYMBOLS/save.gif"),
              (IMAGE,,,,,"$SYMBOLS/cut.gif"),
              (IMAGE,,,,,"$SYMBOLS/paste.gif"),
              (IMAGE,,,,,"$SYMBOLS/pinwheel.gif");

LineSymbol = (CIRCLE, 6, BOTH, white, 1, grey, 0),
              (SQUARE, 6, BOTH, cyan, 1, red, 2),
              (DIAMOND, 6, BOTH, firebrick, 1),
              (CROSS, 6, BOTH, green, 1);
```

### Attributes

BorderColor	BorderWidth	ImageURL	ShadowWidth
Size	Style	SymbolColor	Type

---

## Size

---

`Size` specifies the size of the symbol in pixels. This attribute is ignored for `IMAGE` symbols.

### Example:

```
LineSymbol = (CIRCLE, 6, BOTH, white, 1),
             (SQUARE, 7, BOTH, cyan, 1),
             (DIAMOND, 9, BOTH, firebrick, 1),
             (CROSS, 8, BOTH, green, 1);
```

### Values

Any integer value in pixels

### Default

None

---

## SymbolColor

---

`SymbolColor` specifies for foreground color of the `LineSymbol`.

---

## ShadowThickness

---

`ShadowThickness` specifies for size of the shadow behind a `LineSymbol`. Any number other than 0 causes NetCharts to choose a shadow size based on the size of the symbol. Specifying a value of 0 suppresses the shadow.

---

## Style

---

`Style` specifies how the `LineSymbol` should be drawn, including `FILLED`, `OUTLINED`, or `BOTH`. If `FILLED` is specified, the symbol is filled with the line set color. If `OUTLINED` is specified, only the outline is drawn, using the line set color. If `BOTH` is specified, then the symbol is filled with the line set color and the outline is drawn using the `borderColor`.

### Example:

```
LineSymbol = (CIRCLE, 6, BOTH, white, red),
             (SQUARE, 6, OUTLINE, orchid),
             (DIAMOND, 6, FILLED),
             (CROSS, 6, BOTH, white, darkcyan);
```

### Values

<code>FILLED</code>	Symbol is filled with the <code>LineSet</code> color.
<code>OUTLINED</code>	Only the outline is drawn, using the <code>LineSet</code> color
<code>BOTH</code>	Symbol is filled with the <code>LineSet</code> and the outline is drawn using the <code>BorderColor</code>

### Default

None

---

## Type

---

Type specifies the type of symbol to be displayed on the line set.

**Example:**

```
LineSymbol = (CIRCLE, 6, BOTH, white, 1),
             (SQUARE, 6, BOTH, cyan, 1),
             (DIAMOND, 6, BOTH, firebrick, 1),
             (CROSS, 6, BOTH, green, 1);
```

**Values**

NONE	No symbol is displayed.
CIRCLE	Displays circles
SQUARE	Displays squares
DIAMOND	Displays diamonds
CROSS	Displays crosses
TARGET	Displays targets (bulls-eye)
TRIANGLEDOWN	Displays downward pointing triangles
TRIANGLEUP	Displays upward pointing triangles
IMAGE	If specified, the <code>ImageURL</code> attribute is required and will be used to load a GIF image for the symbol.

**Default**

None

## LineSymbolSpotlights

```
LineSymbolSpotlights = (start,stop,center,centeroffsetx,centeroffsety, focusoffsetx, focusoffsety,radius),...;
```

Adds or overlays a color fill over one or more existing fill patterns to produce multiple layered color effects. The spotlight “illuminates” the line symbols. The center of the spotlight and its focus can be adjusted independently by adjusting offsets from the symbol center point. The elements *centeroffsetx* and *centeroffsety* set the x and y-coordinates of the center of the spotlight as an offset of the chart center point. When set to whole numbers, *centeroffsetx* and *centeroffsety* specify the number of pixels to offset from the chart center point. When set to fractional values (between 0 and 1), they are interpreted as percentages of the width of the symbols. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point. The element *focusoffsetx* is the offset from the chart center which defines the x-coordinate of the focus point of the spotlight. The element *focusoffsety* is the offset from the chart center which defines the y-coordinate of the focus point of the spotlight. When set to whole numbers, *focusoffsetx* and *focusoffsety* specify the offset from the center in pixels. When set to fractional values (between 0 and 1), they are interpreted as percentages of the width of the bars. The element *radius* sets the size of the spotlight, from its center to its edge. When set to a whole number, it sets the size of the *radius* in pixels; when set to a fractional value, it sets the radial diameter of the spotlight based upon that percentage of the minimum height and width of the symbols on the chart.

**Used in These Charts**

Bar, Bubble, Line, Combo, Pareto, Stock, XY

**Example:**

```
LineSymbolSpotlights = (purple_40,blue_155,RIGHT,50,-50,100,150,250);
```

**Attributes**

start	stop	center
centeroffsetx	centeroffsety	focusoffsetx
focusoffsety	radius	

**Start**

`start` specifies the first of the two colors which will be interpolated to produce a gradient spotlight.

**Example:**

```
LineStyleSpotlights = (purple_40, blue_155, LEFT, 20,10,120,-120,250);
```

**Stop**

`stop` specifies the second of two colors which will be interpolated to produce a gradient spotlight.

**Center**

`center` specifies the position around of the center of the chart where the spotlight center will be placed.

**Values**

RIGHT	Offsets the center point of the spotlight to the right of the center point of the chart.
LEFT	Offsets the center point to the left.
TOP	Offsets the center point to the top.
BOTTOM	Offsets the center point to the bottom.
CENTER	Uses the chart center point for the spotlight center point.
TOPRIGHT	Offsets the center point of the spotlight to the top right.
TOPLEFT	Offsets the center point of the spotlight to the top left.
BOTTOMRIGHT	Offsets the center point of the spotlight to the bottom right.
BOTTOMLEFT	Offsets the center point of the spotlight to the bottom left.

**Default**

CENTER

**Centeroffsetx**

`Centeroffsetx` specifies the x-coordinate offset for the spotlight center.

**Centeroffsety**

`Centeroffsety` specifies the y-coordinate offset for the spotlight center.



### ***Focusoffsetx***

---

`Focusoffsetx` specifies the x-coordinate offset for the center of the spotlight's focus.

### ***Focusoffsety***

---

`Focusoffsety` specifies the y-coordinate offset for the center of the spotlight's focus.

### ***Radius***

---

`Radius` specifies the length of the radius of the spotlight from the center of the spotlight.

## **LineValueLabel**

---

---

`LineValueLabel[N]` = (*mode, color, font name, width*);

Defines the label value to be displayed for each point in a lineset.

### ***Used in These Charts***

Bar, Combo, Line, Radar, Stock, Strip, X-Y

### ***Example:***

```
LineValueLabel = ("ON", black, "Helvetica", 18);
```

### ***Attributes***

Mode    Color    Font Name    Width

## **LineValueLabelBox**

---

---

`LineValueLabelBox[N]` = (*color, mode, depth*);

Defines the line label box to be displayed with each point on a lineset.

### ***Used in These Charts***

Bar, Combo, Line, Radar, Stock, Strip, X-Y

### ***Example:***

```
LineValueLabelBox = (grey, RAISED, 3);
```

### ***Attributes***

Color    Mode    Depth

---

## LineValueLabelStyle

---

LineValueLabelStyle[N] = *labelposition1, labelposition2, ... labelpositionN*;

Defines where the LineValueLabel text will display for each point of the lineset.

### *Used in These Charts*

Bar, Combo, Line, Radar, Stock, Strip, X-Y

### *Example:*

```
LineValueLabelStyle = TOP;
```

### *Values*

TOPLEFT	Displayed at the top left point in a lineset
TOP	Displayed at the top of the point in a lineset
TOPRIGHT	Displayed at the top right point in a lineset
LEFT	Displayed at the left of the point in a lineset
CENTER	Displayed at the center point in a lineset
RIGHT	Displayed at the right of the point in a lineset
BOTTOMLEFT	Displayed at the bottom left point in a lineset
BOTTOM	Displayed at the bottom of the point in a lineset
BOTTOMRIGHT	Displayed at the bottom right point in a lineset

### *Default*

TOP

### *Attributes*

Label Position

---

## LineWidth

---

LineWidth[N] = *PercentDepth*;

LineWidth determines the width of a 3D line in a line chart, with reference to the amount of space specified by the z-dimension in the Line3DDepth parameter.

### *Used in These Charts*

Combo, Line, Pareto, Stock, X-Y

### *Example:*

```
LineWidth = 60;
```

### *Attributes*

PercentDepth

---

## *PercentDepth*

---

This percentage value, 0-100 or 0.0-1.0, determines the amount of space actually used to display 3D lines when Line3DDepth is greater than 1.

### *Example:*

```
LineWidth = 0.6;
```

### *Values*

0 to 100 or 0.0 to 1.0 If 100 percent is specified, then each line will completely fill the amount of space specified by Line3DDepth in the Z dimension. If 50 percent is specified, then each line will occupy only 50 percent of that space, which will result in a visible separation between each line set.

### *Default*

100

---

## **Locale**

---

Locale = "language\_country\_variant";

The `Locale` parameter can be used to specify the locale from which to derive the decimal symbol and group symbol used when formatting numeric values, and month and day names and abbreviations used when formatting date values. Specifying this parameter causes the decimal symbol and group symbol from the `NumberFormat` parameter to be ignored.

### *Used in These Charts*

All

### *Examples:*

```
Locale = "fr";  
Locale = "en_US";  
Locale = "no_NO_NY";  
Locale = "es_ES"
```

---

## **MeanActiveLabels**

---

```
MeanActiveLabels = ("Label1", "URL1", "Target1"),...;
```

MeanActiveLabels define the active labels associated with mean values.

### *Used in These Charts*

Box Chart

### *Example:*

```
MeanActiveLabels = (lightgray, SHADOW, 3,,gray);
```

### *Attributes*

*Label, Target, URL*

## MeanColor

---

```
MeanColor = Color;
```

MeanColor allows users to specify the color to be used to display the mean value. This value is used if no color is specified in the MeanSymbol parameter. The default value is the value of the MedianColor

### *Used in These Charts*

Box Chart

#### *Example:*

```
MeanColor = red;
```

### *Attributes*

*Color*

## MeanLine

---

```
MeanLine = (type, width, color);
```

MeanLine allows the users to draw a line that connects the mean of each data series in a Box Chart.

### *Used in These Charts*

Box Chart

#### *Example:*

```
MeanLine = (DASHED, 3, green);
```

type	type of line to draw. Legal values are SOLID, DASHED, DOTTED and DOTDASH
width	width in pixels of the line, the default is 1.
color	color of the line, the default is the MeanColor

### *Attributes*

*Color, Type, Width*

## MeanSymbol

---

MeanSymbol = (type1, size1, style1, bordercolor1, borderwidth1, image1, color1),... ;

MeanSymbol is used to define the style in which to draw the mean value.

### Used in These Charts

Box Chart

### Example:

```
MeanSymbol = (CIRCLE, 6, BOTH, white, 1);
```

typeN	the type of symbol to use for the mean in data series N. Legal values are NONE, CIRCLE, SQUARE, DIAMOND, CROSS, TARGET, TRIANGLEDOWN, TRIANGLEUP, IMAGE
sizeN	the size in pixels of the symbols for the mean in data series N
styleN	the drawing style for the mean in data series N. Legal values are FILLED, OUTLINED or BOTH
borderColorN	the color of the border for the mean in data series N
borderWidthN	the width in pixels of the border for the mean in data series N
imageN	the image to use for displaying for the mean in data series N
colorN	the color for the mean in data series N

### Attributes

Type, Size, Style, borderColor, borderWidth, Image, Color

## MedianColor

---

MedianColor[N] = Color;

MedianColor determines the color to be used when drawing the median. If MedianColor is not defined, the default color is white.

### Used in These Charts

Box

### Example:

```
MedianColor = xE3E3E3;
MedianColor = silver;
```

### Attributes

Color

---

## MetaData

---

```
MetaData = ("name", "value"), ("name", "value"), ...;
```

The `MetaData` parameter allows a chart writer to embed useful information about the chart within its definition.

### *Used in These Charts*

ALL

### *Example:*

The parameter allows the setting of any number of name/value pairs. For example,

```
MetaData=("Author", "John Doe"),  
("Creation Date", "January 21, 2001"),  
("Department", "Sales");
```

This information, once embedded in the chart through the `MetaData` parameter, can be searched via a NetCharts Server. For example, let's say you want to search your server for all charts created for the sales department. A possible solution would be to institute a company-wide policy mandating that all charts have a `MetaData` parameter containing the department for which the chart was developed. Searching for the word `Department` and `Sales` in all charts would give the desired results.

### *Attributes*

*Name*            *Value*

---

## MinimumDataPoints

---

```
MinimumDataPoints = int_val;
```

`MinimumDataPoints` defines the number of data points the must be present in a series in order for the chart to draw in summary mode. IF `ShowDataPoints` is OFF, and the minimum number of data points necessary to calculate a summary display are not present, the raw data will be displayed.

### *Used in These Charts*

Box Chart

### *Example:*

```
MinimumDataPoints = 12;
```

### *Attributes*

*Value*

## NaturalDisplayOrder

---

NaturalDisplayOrder = ON / OFF;

The new Box Chart provides control over the left to right order of data set displays. The default behavior for a BoxChart in VERTICAL mode is to display the series from right to left. Most charts make more sense when they are laid out left to right. The NaturalDisplayOrder parameter provides control over this behavior.

### Used in These Charts

Box Chart

### Example:

```
NaturalDisplayOrder = ON;
NaturalDisplayOrder = OFF;
```

### Default

ON

### Attributes

Mode

## NodeBox

---

NodeBox[N] = (Color, BorderType, BorderWidth, ImageURL, ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor), ...;

The NodeBox parameter controls the appearance of the box that forms the nodes in diagram charts. If only one 'tuple is used, all the boxes will appear the same. Otherwise, the order of the 'tuples maps to the order in which the nodes were defined in the Nodes parameter.

Notice that in the example, a one-pixel transparent GIF image is used to invisibly anchor two edges so that the edges may look as though they turn a corner. This is done because the Edge parameter only allows one single straight line per edge.

### Used in These Charts

Diagram

### Example:

```
NodeBox =
  (white, BOX, 1, ".../images/blonde-woman.gif",,gray)
  (white, BOX, 1, ".../images/susan.gif",,gray),
  (white, BOX, 1, ".../images/glasses-girl.gif",,gray),
  (white, BOX, 1, ".../images/tie-guy.gif",,gray),
  (white, BOX, 1, ".../images/coffeeshopguy.gif",,gray),
  (white, NONE, 0, ".../images/pixel-clear.gif"),
  (white, NONE, 0, ".../images/pixel-clear.gif"),
  (white, NONE, 0, ".../images/pixel-clear.gif");
```

### Attributes

*Color*      *BorderType*    *BorderWidth*    *ImageURL*      *ImageFormat*  
*BorderColor*   *TRCornerStyle* *BRCornerStyle* *BLCornerStyle* *CornerColor*

### XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

## NodeDrag

---

```
NodeDrag[N] = "ON"/"OFF";
```

The `NodeDrag` switch is used to allow or stop the user from dragging the nodes of the diagram chart with the mouse.

### Used in These Charts

Diagram

#### Example:

```
NodeDrag = "ON";  
NodeDrag = "OFF";
```

### Attributes

(Switch)

### Switch

---

This switch sets the on/off state.

#### Example:

```
NodeDrag = "ON";  
NodeDrag = "OFF";
```

### Values

ON	Allows the user to drag the hands on the applet dial
OFF	Stops the user from dragging the hands on the applet dial

### Default

ON

## NodeLabel

---

```
NodeLabel[N] = (Color, FontName, FontSize, Angle), ...;
```



`NodeLabel` is an optional parameter that controls the specific appearance and style of the labels for named nodes defined with the `Nodes` parameter. The order of the node labels maps to the order in which the nodes were defined. If any attribute is not specified in the vector, then the previously specified attribute will be used.

Note that if any attribute is not specified in the 'tuple, then the previously specified attribute will be used. If a color is specified as `NULL` or omitted, then a color will be chosen from a `ColorTable` parameter.

### Used in These Charts

#### Diagram

#### Example:

```
NodeLabel = (black, "Helvetica", 8);
NodeLabel = (yellow, "TimesRoman", 8, 0), (yellow, "TimesRoman", 8, 0);
```

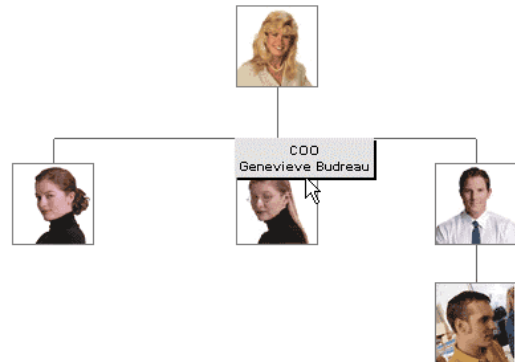
#### Attributes

*Label*      *Color*            *FontName*      *FontSize*      *Angle*      *interiorAlignment*

## Nodes

```
Nodes [N] = (Name, Label, X, Y), ...;
```

`Nodes` defines all the nodes in a diagram chart, by naming, labeling, and positioning them. The example below was used in the diagram chart on the right.



### Used in These Charts

#### Diagram

#### Example:

```
Nodes =
("CEO", "", 200, 31),
("VP Sales", "", 31, 150),
("COO", "", 200, 150),
("VP Marketing", "", 350, 150),
("Webmaster", "", 350, 240)
("sales-node", "", 31, 100), <!-- Invisible node to turn a corner -->
("coo-node", "", 200, 100), <!-- Invisible node to turn a corner -->
("marketing-node", "", 350, 100); <!-- Invisible node to turn a corner -->
```

#### Attributes

*Name*            *Label*            *X*            *Y*

## X

The `x` value maps the location of the center of a node along the chart's X-axis, and is given in pixels.

#### Example:

```
Nodes =
("CEO", "", 200, 31),
```

```

("VP Sales", "", 31, 150),
("COO", "", 200, 150),
("VP Marketing", "", 350, 150),
("Webmaster", "", 350, 240)
("sales-node", "", 31, 100), <!-- Invisible node to turn a corner -->
("coo-node", "", 200, 100), <!-- Invisible node to turn a corner -->
("marketing-node", "", 350, 100); <!-- Invisible node to turn a corner -->

```

**Values**

Positive numbers greater than or equal to 0.

**Default**

No defaults

**Y**

The Y value maps the location of the center of a node along the chart's Y-axis, and is given in pixels.

**Example:**

```

Nodes =
("CEO", "", 200, 31),
("VP Sales", "", 31, 150),
("COO", "", 200, 150),
("VP Marketing", "", 350, 150),
("Webmaster", "", 350, 240)
("sales-node", "", 31, 100), <!-- Invisible node to turn a corner -->
("coo-node", "", 200, 100), <!-- Invisible node to turn a corner -->
("marketing-node", "", 350, 100); <!-- Invisible node to turn a corner -->

```

**Values**

Positive numbers greater than or equal to 0.

**Default**

No defaults

**NoteActiveLabels[n]**

```
NoteActiveLabels[1-20] = ("Label", "URL", "Target"), ...;
```

NoteActiveLabels[n] defines a list of custom active labels to be associated with each note in a particular NoteSet. That is, these labels will be displayed whenever the mouse "dwells" over a given note. This can be used to provide the user with additional information about that particular note or to "drilldown" to another document. See ActiveLabels for a detailed explanation of active label capabilities.

**Used in These Charts**

All

**Example:**

```
NoteActiveLabels1 = ("$13,422", "../daily/stats.html", "statwin"), ("$27,002"),
("$33,812"), ("$12,799");
```

**Attributes**

<i>Label</i>	<i>URL</i>	<i>Target</i>
--------------	------------	---------------

**NoteArrow**


---

```
NoteArrow[N] = (LineType1, LineWidth1, LineColor1, ArrowType1, ArrowStyle1), (LineType2, LineWidth2, LineColor2, ArrowType2, ArrowStyle2), ...;
```

`NoteArrow` defines a list of list of line and arrow definitions, with the first in the list referring to `NoteSet1`, the second in the list to `NoteSet2`, and so on.

**Used in These Charts**

All

**Example:**

```
NoteArrow = (SOLID, 2, red, TOFROM, BLOCK);
```

**Attributes**

<i>LineType</i>	<i>LineWidth</i>	<i>LineColor</i>	<i>ArrowType</i>	<i>ArrowStyle</i>
-----------------	------------------	------------------	------------------	-------------------

**LineColor**


---

`LineColor` controls the color of the line in the `NoteArrow`. This parameter operates in all respects as the typical `Color` parameter. See Chapter 4 for specifics.

**ArrowType**


---

`ArrowType` controls the direction of the arrowhead terminating the `NoteArrow`.

**Example:**

```
NoteArrow = (SOLID, 2, red, TOFROM, BLOCK);
```

**Values**

NONE	No arrowhead is shown
FROMTO	Arrowhead is oriented and placed with last point on the line
TOFROM	Arrowhead is oriented and placed with first point on the line
BOTH	Two arrowheads are used, one oriented and placed with the last point on the line, the second oriented and placed with the first point on the line.

**Default**

NONE

**ArrowStyle**


---

`ArrowStyle` controls the shape of the arrowhead terminating the `NoteArrow`.

**Example:**

```
NoteArrow = (SOLID, 2, red, TOFROM, BLOCK);
```

**Values**

ROUND	Round arrowhead
SHARP	Conventional triangular pointed arrowhead
BLOCK	Square arrowhead

**Default**

SHARP

**NoteAxis**

```
NoteAxis[N] = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
```

`NoteAxis` defines a list of axis mapping pairs, with the first in the list referring to `NoteSet1`, the second in the list to `NoteSet2`, and so on.

When an X and Y value must be mapped to screen pixels, the axis mapping pair for that `NoteSet` is used to determine the mapping strategy.

The `XAxis` and `YAxis` parameters may be of different types.

**Used in These Charts**

All

**Example:**

```
NoteAxis = (PERCENT, LEFT), (BOTTOM, LEFT), (BOTTOM, LEFT);
```

**Attributes**

<code>XAxis</code>	<code>YAxis</code>
--------------------	--------------------

**XAxis**

`XAxis` controls to which location the Notes' X values are relative.

**Example:**

```
NoteAxis = (PERCENT, PERCENT), (BOTTOM, LEFT), (PIXEL, LEFT);
```

**Values**

BOTTOM	X value is relative to bottom axis values
TOP	X value is relative to top axis values
PERCENT	X value is a percentage of window width (0-100%)
PIXEL	X value is an absolute pixel location relative to the left edge of the applet

**Default**

BOTTOM

---

## YAxis

---

`YAxis` controls to which location the Notes' Y values are relative.

### Example:

```
NoteAxis = (PERCENT, PERCENT), (BOTTOM, LEFT), (PIXEL, PIXEL);
```

### Values

LEFT            Y value is relative to left axis values  
 RIGHT           Y value is relative to right axis values  
 PERCENT        Y value is a percentage of window height (0-100%)  
 PIXEL           Y value is an absolute pixel location relative to the top edge of the applet

### Default

LEFT

---

## NoteBox

---

```
NoteBox[N] = (Color1, BorderType1, BorderWidth1, ImageURL1, ImageFormat1, TRCornerStyle, BRCornerStyle,
  BLCornerStyle, CornerColor), ...;
```

The `NoteBox` parameter is a list of region definitions, with the first in the list referring to `NoteSet1`, the second in the list to `NoteSet2`, and so on. The `NoteBox`, if defined, is drawn underneath the note label and can be used to highlight the note.

### Used in These Charts

All

### Example:

```
NoteBox = (white, NONE), (white, RAISED);
```

### Attributes

<i>Color</i>	<i>BorderType</i>	<i>BorderWidth</i>	<i>ImageURL</i>	<i>ImageFormat</i>
<i>TRCornerStyle</i>	<i>BRCornerStyle</i>	<i>BLCornerStyle</i>	<i>CornerColor</i>	

---

## XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

---

## NoteDrag

---

```
NoteDrag[N] = ON / OFF;
```

The `NoteDrag` feature allows `NoteSets` to be individually configured to allow, or dis-allow dragging.

***Used in These Charts***

Diagram

***Example:***

```
NoteDrag = "ON";
```

***Attributes***

Mode

***Values***

ON                Allows the user to drag NoteSets  
 OFF              Stops the user from dragging NoteSets

***Default***

ON

**NoteLabel**

```
NoteLabel[N] = ("Label1", Color1, "FontName1", FontSize1, Angle1), ...;
```

The `NoteLabel` parameter contains a list of label definitions, with the first parenthesized group in the list referring to `NoteSet1`, the second parenthesized group defining `NoteSet2`, and so on.

If `Mode` is `OFF`, then the note text for that `NoteSet` will not be shown. In any other case, the `NoteSet` text will be drawn using the given label attributes in the `NoteLabel` parameter.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

***Used in These Charts***

All

***Example:***

```
NoteLabel = ("ON",black, Helvetica, 10), ("ON",black, Helvetica, 10),
            ("ON",black, Helvetica, 10);

NoteLabel = ("", black, "Dialog", 12), ("", x5E8109, "Helvetica", 12), ("",
            x1A4C8F, "Helvetica", 12, 270);
```

***Attributes***

*Label*    *Color*        *FontName*    *FontSize*    *Angle*    *interiorAlignment*

**NoteSet[n]**

```
NoteSet1 = ("Text1", X, Y, X1, Y1, X2, Y2, X3, Y3), ...;
```

For each defined `NoteSet` defined in the `NoteSets` parameter, there should be a corresponding `NoteSet[n]` where `[n]` refers to the appropriate `NoteSet` number.

Each `NoteSet` may have an unlimited number of notes, each with its own text string, X-Y position vector, and optional point vectors to draw an arrow segment. While the X-Y vector must be defined, the X1-Y1, X2-Y2, and X3-Y3 vectors are optional and are used to draw a line segment, if desired.

### *Used in These Charts*

All

### *Example:*

```
NoteSet1 = ("x", 95, 35, 5, 35), ("y", 50, 90, 50, 25);
NoteSet2 = ("L", 48, 65), ("a", 65, 33), ("0", 47, 29), ("f(x) = L", 65, 25),
("lim\ nx -> a", 38, 25);
```

### *Attributes*

Text	X	Y	X1	Y1	X2	Y2	X3	Y3
------	---	---	----	----	----	----	----	----

### *Text*

---

`Text` displays the text of the `NoteSet`. The text may contain multiple lines, delimited by the `\n` (new line) symbol.

### *Example:*

```
NoteSet1 = ("The Amplifier clearly led\nall components in overall\nsales. Bob
McIness and \nstaff did a great job.",82,35,68,70);
```

### *Values*

Any alphanumeric text. The text may contain multiple lines, delimited by the `\n` (new line) symbol. `Text` may also be a null, or empty string, "".

### *Default*

None

### *X, Y*

---

`x` and `y` are required. These first two attributes control the note's relative position. This location vector is also used as the first point for the arrow line if X1-Y1 are defined.

### *Example:*

```
NoteSet1 = ("The Amplifier clearly led\nall components in overall\nsales. Bob
McIness and \nstaff did a great job.",82,35,68,70);
```

### *Values*

Any real numbers. What you use also depends on what you defined with the `NoteAxis` parameter.

**Default**

None

**X1, Y1**

---

X1 and Y1 are optional, and define either a corner or an endpoint for an arrow line for their note.

**Example:**

```
NoteSet1 = ("The Amplifier clearly led\nall components in overall\nsales. Bob  
McIness and \nstaff did a great job.", 82, 35, 68, 70);
```

**Values**

Any real numbers. What you use also depends on what you defined with the `NoteAxis` parameter.

**Default**

None

**X2, Y2**

---

X2 and Y2 are optional, and define either a corner or an endpoint for an arrow line for their note.

**Example:**

```
NoteSet4 = ("Sudden Gains", .5, 45, 0, 55, 10, 55, 10, 65);
```

**Values**

Any real numbers. What you use also depends on what you defined with the `NoteAxis` parameter.

**Default**

None

**X3, Y3**

---

X3 and Y3 are optional, and define the endpoint for an arrow line for their note.

**Example:**

```
NoteSet4 = ("Sudden Gains", .5, 45, 0, 55, 10, 55, 10, 65);
```

**Values**

Any real numbers. What you use also depends on what you defined with the `NoteAxis` parameter.

**Default**

None



---

## NoteSets

---

```
NoteSets[N] = ("Name1", Justify1), ("Name2", Justify2), ...;
```

The `NoteSets` parameter is a required parameter for displaying notes, and contains a collection of `NoteSet` names and justification types. The number of defined `NoteSets[n]` is based on the number of parenthesized groups, or tuples, defined herein.

### *Used in These Charts*

All

### *Example:*

```
NoteSets = ("note1"), ("note2");
```

```
NoteSets = ("Axes", BOTTOM), ("TextOnly", CENTER), ("Text2Only", TOP);
```

### *Attributes*

Name                      Justify

### *Name*

---

Name is the identifying name for the `NoteSet`.

### *Example:*

```
NoteSets = ("Axes", BOTTOM), ("TextOnly", CENTER), ("Text2Only", TOP);
```

### *Values*

Simple alphanumeric text, which will not be seen by the user.

### *Default*

None. Must have a name.

### *Justify*

---

`Justify` defines where each note is placed relative to the note's X-Y location, which is defined in the `NoteSet[n]` parameter.

### *Example:*

```
NoteSets = ("Axes", BOTTOM), ("TextOnly", CENTER), ("Text2Only", TOP);
```

**Values**

TOPLEFT	The note's top left corner becomes its origin
TOP	A point horizontally centered on the note's top edge becomes its origin
TOPRIGHT	The note's top right corner becomes its origin
LEFT	A point vertically centered on the left side becomes its origin
CENTER	The center of the note becomes its origin
RIGHT	A point vertically centered on the right side becomes its origin
BOTTOMLEFT	The note's bottom left corner becomes its origin
BOTTOM	A point horizontally centered on the note's bottom edge becomes its origin
BOTTOMRIGHT	The note's bottom right corner becomes its origin

**Default**

CENTER

**NotesDrawnBeforeData**


---



---

```
NotesDrawnBeforeData=ON|OFF;
```

The `NotesDrawnBeforeData` parameter supports the values ON (draw Notes first, behind the data points) and OFF (draw Notes last, over top of the data points, the default value).

**Used in These Charts**

All

**Example:**

```
NotesDrawnBeforeData=ON;
```

Would draw Notes first, behind the data points.

**NumberFormat**


---



---

```
NumberFormat = ("decimalSymbol", "groupSymbol", groupSize);
```

The `NumberFormat` parameter can be used to specify the symbols and group size used when formatting numeric values. The default `decimalSymbol` is ".", the default `groupSymbol` is "," and the default `groupSize` is 3.

**Used in These Charts**

All

**Example:**

```
NumberFormat = (",", " , ".");
```

would generate the following results for the given format for the decimal value 1234.456:

%f	1234,46
%.f	1.234,46
%d	1234

<code>%d</code>	1.234
<code>%.8,1f</code>	1.234,5
<code>%08,1f</code>	001234,5

### Attributes

`decimalSymbol`      `groupSize`      `groupSymbol`

## OutlierActiveLabels

---

```
OutlierActiveLabels = ("Label1", "URL1", "Target1"),...;
```

OutlierActiveLabels define the active labels associated with mean values.

### Used in These Charts

Box Chart

### Example:

```
OutlierActiveLabels = ("Exception condition", "javascript:doException()");
```

### Default

### Attributes

`Label`, `URL`, `Target`

## OutlierColor

---

```
OutlierColor[N] = Color;
```

`OutlierColor` controls the color to be used when drawing outliers in a box chart. This value is used if no color is specified in the `OutlierSymbol` parameter. The default value is the value of `MedianColor`.

### Used in These Charts

Box

### Example:

```
OutlierColor = xE3E3E3;  
OutlierColor = silver;
```

### Attributes

`Color`

## OutlierSymbol

---

OutlierSymbol = (type1, size1, style1, bordercolor1, borderwidth1, image1, color1),...;

OutlierSymbol is used to define the style in which to draw the mean value.

### Used in These Charts

Box Chart

### Example:

```
OutlierSymbol = (NONE, 3, FILLED, green,2, , blue);
```

### Default

ON

typeN	the type of symbol to use for outliers in data series N. Legal values are NONE, CIRCLE, SQUARE, DIAMOND, CROSS, TARGET, TRIANGLEDOWN, TRIANGLEUP, IMAGE
sizeN	the size in pixels of the symbols for outliers in data series N
styleN	the drawing style for outliers in data series N. Legal values are FILLED, OUTLINED or BOTH
borderColorN	the color of the border for outliers in data series N
borderWidthN	the width in pixels of the border for outliers in data series N
imageN	the image to use for displaying for outliers in data series N
colorN	the color for outliers in data series N

### Attributes

Type, Size, Style, BorderColor, BorderWidth, Image, Color

## PercentileN

---

PercentileN = Integer;

PercentileN defines the value of N when PlotType=PERCENTN. N specifies that for a given data series the chart will show the Nth percentile, median and 100-Nth percentile as a box, and draws whiskers to the minimum and maximum data points.

### Used in These Charts

Box

### Example:

```
PercentileN = 5;
```

---

## PieAngle

---

`PieAngle[N] = Integer;`

`PieAngle` specifies the starting angle for the first pie slice. The angle is measured in degrees from 0 to 360, in a counterclockwise direction, with 0 degrees pointing to the right (i.e., the 3 o'clock position).

### *Used in These Charts*

Pie

### *Example:*

```
PieAngle = 180;
```

### *Attributes*

None

---

## PieAngles

---

`PieAngles = Value1, Value2,...;`

`PieAngles` specifies the starting angle for each pie. The angle is measured in degrees from 0 to 360, in a counterclockwise direction, with 0 degrees pointing to the right (i.e., the 3 o'clock position)

### *Used in These Charts*

MultiPie

### *Example:*

```
PieAngles 15,0,10,10....;
```

### *Attributes*

None

---

## PieBackgrounds

---

`PieBackgrounds = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor), ...;`

`PieBackground` specifies the background for each pie.

### *Used in These Charts*

MultiPie

### *Example:*

```
PieBackgrounds = (pink,RAISED,10,,,),(lightgray,BOX,1,,,);
```

### *Attributes*

*Color BorderType BorderWidth ImageURL ImageFormat BorderColor*

---

## PieDropShadow

---

```
PieDropShadow = (color, offsetx, offsety, size);
```

`PieDropShadow` places a shadow on the background field of the pie chart. The color, orientation, and size of the shadow can be defined. The tuple element `color` sets the color of the shadow. That color value is interpolated to complete transparency as it reaches the end of the shadow's blur area. `offsetx` and `offsety` define the center point of the shadow; `offsetx` sets the x-axis offset from the chart's center-point; `offsety` sets the y-axis offset. When an offset attribute is set to a whole number value, the position of the drop shadow is offset from the chart's center point by the number of pixels set by that whole number. When an offset is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow (the pie). The center of the drop shadow is repositioned based upon the values or percentages set for `offsetx` and `offsety`. Offset attribute values may be positive or negative. `size` sets the size of the blur area, the region beyond the actual drop shadow shape where the shadow is extended and blurred into transparency. The size of this blurred region is controlled by the `size` attribute. The blurred region becomes larger and more diffuse as the value of `size` is increased. When `size` is set to a whole number value, the size of the blurred area is defined in pixels. When `size` is set to a fractional value (between 0 and 1), the value is interpreted as a percentage of the width of the object casting the shadow. When using a fractional value, enclose the value in double-quotes to "escape" the decimal point.

### *Used in These Charts*

Pie, Multipie

### *Example:*

```
PieDropShadow = (Black_100,5,5,"0.05");
```

### *Attributes*

Color	Offsetx	offsety	Size
-------	---------	---------	------

### *Color*

---

`Color` specifies the base color of the shadow to be drawn behind a chart's bars.

### *Example:*

```
PieDropShadow = (black_100, "-.05", "-0.05", 55);
```

### *Offsetx*

---

`Offsetx` specifies the x-coordinate offset from center.

**Example:**

```
PieDropShadow = (black_100, 25 -10, 25);
```

**Offsety**

---

`Offsety` specifies the y-coordinate offset from center.

**Example:**

```
PieDropShadow = (black_100, 25 -10, 25);
```

**Size**

---

`Size` specifies the width of the blurred area.

**Example:**

```
PieDropShadow = (black, "-.05","-.05", 55);
```

**PieEdgeHighlights**

---

```
PieEdgeHighlights = (start,stop,gap,size), ...;
```

The `PieEdgeHighlights` parameter provides a visual pattern fill in a Pie or MultiPie chart which accents the inner pie border. It overlays a ring (annulus) fill pattern over the existing fill patterns in a specified zone along the interior edge of the pie. The gap between the sides of the pie and the fill pattern being applied can be modified. The element `start` sets the beginning color of the gradient, associated with the outside edge; the element `stop` sets the end color of the gradient, associated with the interior of the pie where the color blends to transparency. Color values are interpolated between the two. The element `size` specifies the width of the highlight. The element `gap` specifies the size of the gap between the edge of the highlight and the edge of the pie. When the value for `gap` is specified as a whole number, it sets the distance between the edge of the highlight and the edge of the pie in pixels. When set to a fractional number between 0 and 1, it sets the gap to a percentage of the radius of the pie. Percentage values are written using a decimal point (e.g. "0.05" and "0.50" represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to "escape" the decimal point.

**Used in These Charts**

Pie, Multipie

**Example:**

```
PieEdgeHighlights = (blue_25,white_75,1,25), ...;
```

**Attributes**

<code>start</code>	<code>stop</code>	<code>gap</code>	<code>size</code>
--------------------	-------------------	------------------	-------------------

## PieHighlights

`PieHighlights = (type,start,stop,angle,gap,extent), ...;`

The `PieHighlights` parameter provides a visual pattern fill in a Pie or MultiPie chart. It adds or overlays a fill pattern over one or more existing fill patterns to produce multiple color effects. The *angle* of origin of the gradient pattern can be modified. The gap between the sides of the pie and the fill pattern being applied can be modified. Gradient patterns can be set using the *type* attribute. Only gradient patterns may be used. A *type* value of NONE suppresses the highlights. The element *start* sets the beginning color of the gradient; the element *stop* sets the end color of the gradient. Color values are interpolated between the two. The element *angle* specifies the number of degrees from zero from which the initial gradient color begins at the edge of the pie. The element *angle* can be set to values greater than 360 degrees. The element *gap* specifies the size of the gap between the edge of the highlight and the associated edge of the pie. When the value for *gap* is specified as a whole number, it sets the distance between the edge of the highlight and the edge of the pie in pixels. When set to a fractional number between 0 and 1, it sets the gap to a percentage of the diameter of the pie. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point. The element *extent* controls the width of the highlight. When the value for *extent* is specified using a whole number, it sets the diameter of the highlight in pixels. If *extent* is set to -1, `PieHighlights` fills the diameter available after taking into account the value of the *gap* attribute specified previously. If *extent* is set to a fractional number between 0 and 1, `PieHighlights` sets the extent of the highlight to that percentage of the diameter of the pie available.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENTBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENANGLED	top left to bottom right style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
<b>start</b>	This color is used in the following ways: - <i>Starting color for gradients</i>	
<b>stop</b>	This color is used in the following ways: - <i>Stopping color for gradients</i>	

### Used in These Charts

Pie, MultiPie

### Example:

```
PieHighlights = (GRADIENRADIAL, yellow, white, 270, 15, -1);
```



**Attributes**

<i>type</i>	<i>start</i>	<i>stop</i>
<i>angle</i>	<i>gap</i>	<i>extent</i>

---

**PieLabel**

---

```
PieLabel = (State, Color, FontName, FontSize, Angle, InteriorAlignment ),...;
```

PieLabel controls the appearance of the text in the pie labels.

**Used in These Charts**

MultiPie

**Example:**

```
PieLabel = ("ON", teal, "Sansserif Bold", 12, 0, null),...;
```

**Attributes**

*State Color FontName FontSize Angle InteriorAlignment*

---

**PieLabelBox**

---

```
PieLabelBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor),...;
```

PieLabelBox controls the appearance of the optional box surrounding each pie label.

**Used in These Charts**

MultiPie

**Example:**

```
PieLabelBox = ( teal, "SOLID", 1);
```

**Attributes**

*Color BorderType BorderWidth ImageURL ImageFormat BorderColor*

---

**PieLayout**

---

```
PieLayout = (Orientation,Row,Columns);
```

PieLayout defines the layout for the pie series. Orientation is either horizontal or vertical.

**Used in These Charts**

MultiPie

**Example:**

```
PieLabelLayout = (VERTICAL,3,1);
```

**Default**

HORIZONTAL

**Attributes***Orientation*      *Number of Rows*      *Number Of Columns***PieLabelLocation**

---

```
PieLabelLocation = Location;
```

`PieLabelLocation` defines the location of the pie labels

Location can be set to TOP, BOTTOM, LEFT or RIGHT.

**Used in These Charts**

MultiPie

**Example:**

```
PieLabelLocation = LEFT
```

**Default**

TOP

**Attributes***Location***PieLabels**

---

```
PieLabels = Label1, Label2, Label3, ... LabelN;
```

Specifies the text for the labels associated with each pie

**Used in These Charts**

MultiPie

**Example:**

```
PieLabels = "North", "South"...;
```

**Default****Attributes***Label*

---

## PieMargin

---

`PieMargin = Integer;`

`PieMargin` defines the margin between pies in the pie layout.

Used in These Charts

MultiPie

### *Example:*

```
PieMargin = 1
```

### *Attributes*

NONE

---

## PieSize

---

`PieSize = (minWidth, minHeight, maxWidth, maxHeight) ;`

The `PieSize` parameter can be used to set minimum and maximum sizes for the actual pie in a pie chart. This allows programmers to guarantee that the pie portion will always be the same size regardless of the length of the strings in the legend or slice labels. `PieSize` has the following interaction with `PieSquare`; if the minimum or maximum dimensions specified are not square, and `PieSquare` is ON, then the pie will be inscribed in a square with a dimension that ranges from the smallest of the minimum values to the smallest of the maximum values.

*Used in These Charts*

Pie

### *Example:*

```
PieSize = (5,25);
```

### *Attributes*

*minWidth, minHeight, maxWidth, maxHeight*

---

## PieSpotlights

---

`PieSpotlights = (start,stop,center,centeroffsetx,centeroffsety, focusoffsetx, focusoffsety,radius),... ;`

Adds or overlays a color fill over one or more existing fill patterns to produce multiple layered color effects. The spotlight “illuminates” the pie in the Pie or MultiPie chart. The center of the spotlight and its focus can be adjusted independently by adjusting offsets from the pie chart center point. The elements *centeroffsetx* and *centeroffsety* set the x and y-coordinates of the center of the spotlight as an offset of the pie chart center point. When set to whole numbers, *centeroffsetx* and *centeroffsety* specify the number of pixels to offset from the chart center point. When set to fractional values (between 0 and 1), they are interpreted as percentages of the diameter of the pie. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point. The element *focusoffsetx* is the offset from the pie chart

center which defines the x-coordinate of the focus point of the spotlight. The element *focusoffsety* is the offset from the pie chart center which defines the y-coordinate of the focus point of the spotlight. When set to whole numbers, *focusoffsetx* and *focusoffsety* specify the offset from the center in pixels. When set to fractional values (between 0 and 1), they are interpreted as percentages of the diameter of the pie. The element *radius* sets the size of the spotlight, from its center to its edge. When set to a whole number, it sets the size of the *radius* in pixels; when set to a fractional value, it sets the radial diameter of the spotlight based upon that percentage of the diameter of the pie.

### Used in These Charts

Pie, MultiPie

#### Example:

```
PieSpotlights = (blue_0,blue_105,RIGHT,1,1,120,-120,440);
```

#### Attributes

start	stop	center
centeroffsetx	centeroffsety	focusoffsetx
focusoffsety	radius	

### Start

---

*start* specifies the first of the two colors which will be interpolated to produce a gradient spotlight.

#### Example:

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

### Stop

---

*stop* specifies the second of two colors which will be interpolated to produce a gradient spotlight.

#### Example:

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

### Center

---

*center* specifies the position around of the center of the chart where the spotlight center will be placed.

#### Example:

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Values**

RIGHT	Offsets the center point of the spotlight to the right of the center point of the chart.
LEFT	Offsets the center point to the left.
TOP	Offsets the center point to the top.
BOTTOM	Offsets the center point to the bottom.
CENTER	Uses the chart center point for the spotlight center point.
TOPRIGHT	Offsets the center point of the spotlight to the top right.
TOPLEFT	Offsets the center point of the spotlight to the top left.
BOTTOMRIGHT	Offsets the center point of the spotlight to the bottom right.
BOTTOMLEFT	Offsets the center point of the spotlight to the bottom left.

**Default**

CENTER

**Centeroffsetx**

---

Centeroffsetx specifies the x-coordinate offset for the spotlight center.

**Example:**

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Centeroffsety**

---

Centeroffsety specifies the y-coordinate offset for the spotlight center.

**Example:**

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Focusoffsetx**

---

Focusoffsetx specifies the x-coordinate offset for the center of the spotlight's focus.

**Example:**

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

**Focusoffsety**

---

Focusoffsety specifies the y-coordinate offset for the center of the spotlight's focus.

**Example:**

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

---

## *Radius*

---

`Radius` specifies the length of the radius of the spotlight from the center of the spotlight.

### *Example:*

```
PieSpotlights = (purple_40,blue_155,LEFT,20,10,120,-120,250);
```

---

## **PieSquare**

---

`PieSquare = Switch;`

`PieSquare`, when turned on, tells the pie chart that the appearance of the pie chart should be kept as high as it is wide. This prevents the pie shape from becoming too wide or too narrow because of titles, labels, legends, or other chart features.

### *Used in These Charts*

Pie, MultiPie

### *Example:*

```
PieSquare = ON;  
PieSquare = OFF;
```

### *Attributes*

None

---

## **Pie3Ddepth**

---

`Pie3Ddepth[N] = Pixels;`

`Pie3DDepth` specifies the apparent 3D depth of the pie in pixels (whole number).

### *Used in These Charts*

Pie MultiPie

### *Example:*

```
Pie3DDepth = 12;
```

### *Attributes*

None

---

## PlotArea

---

```
PlotArea = (xlocation, ylocation, width, height);
```

PlotArea allows chart designers to define and bound the position of the plot area within the chart. The plot area is defined as the area where the data points and axes are displayed. By default, NetCharts charts are laid out with respect to the text elements to the left, right, top, and bottom of the chart while assigning the remaining real estate to the PlotArea. Specifying PlotArea changes this behavior and causes the chart to be laid out with respect to the PlotArea while assigning the remaining real estate to text elements in the top, bottom, left and right of the chart.

If PlotArea is defined in absolute terms (in pixels), the plot area will remain a fixed size, and will be clipped if it does not fit within the chart. If the chart size should change, the extra or eliminated space will affect only the space to the top, bottom, left and right of the plot area. Text in the top, bottom, left or right will be clipped if it does not fit in the space.

If PlotArea is defined in relative terms (as a percentage of the chart size) the PlotArea will grow and shrink proportionally with changes to the chart size. Text in the top, bottom, left or right will be clipped if it does not fit in the space outside the plot area.

*xlocation* - x coordinate of the upper left corner of the plot area within the chart. If *xlocation* is a number between 0 and 1, it is interpreted as a percentage of the total available chart width. If *xlocation* is  $\geq 1$ , it is interpreted as an absolute location in pixels.

*ylocation* - y coordinate of the upper left corner of plot area within the chart. If *ylocation* is a number between 0 and 1, it is interpreted as a percentage of the total available chart height. If *ylocation*  $\geq 1$  it is interpreted as an absolute location in pixels.

*width* - width of plot area. If *width* is a number between 0 and 1, it is interpreted as a percentage of the total available chart area. If *width*  $\geq 1$  it is interpreted as an absolute size in pixels.

*height* - width of plot area. If *height* is a number between 0 and 1, it is interpreted as a percentage of the total available chart height. If *height*  $\geq 1$  it is interpreted as an absolute size in pixels.

### *Used in These Charts*

Bar, Box, Bubble, Combo, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
PlotArea = (1,1,1,1);
```

### *Attributes*

*xlocation*      *ylocation*      *width*      *height*

---

## PlotType

---

```
PlotType = Type;
```

`PlotType` defines the type of plot to be displayed for a `BoxChart`.

### Used in These Charts

Box

#### Example:

```
PlotType = STANDARD | EDA | GAUSSIAN | PERCENTN | TENNINETY;
```

#### Attributes

Type

### Type

---

Type refers to the manner in which lines are stacked in a line chart, allowing the creation of area graph.

#### Example:

```
PlotType = STANDARD;
```

#### Values

STANDARD	In the Standard type of box plot, the box represents the IQR. The Inter-Quartile Range is the difference between the upper hinge (the 75 <sup>th</sup> percentile) and the lower hinge (the 25 <sup>th</sup> percentile).
EDA	The Exploratory Data Analysis (EDA) box chart is similar to the Standard box chart in that the box is based on the IQR, and the median is plotted as a line through the box. (The lower part of the box is also referred to as the <i>lower quartile</i> , and the upper part of the box is then referred to as the <i>upper quartile</i> .) However, this type of plot features additional elements called fences. The fences are defined as follows: <b>Lower Outer Fence</b> = lower quartile - 3 * IQR <b>Lower Inner Fence</b> = lower quartile - 1.5 * IQR <b>Upper Inner Fence</b> = upper quartile + 1.5 * IQR <b>Upper Outer Fence</b> = upper quartile + 3 * IQR
GAUSSIAN	The Gaussian box chart is different from the other options in terms of which statistics is represents. The lower portion of the box is the minimum value, and the upper portion of the box is the maximum value. The mean value is shown as a line through the box. The lower whisker is equal to the mean value minus three standard deviations, and the upper whisker is equal to the mean value plus three standard deviations.
PERCENTN	This represents the minimum, N <sup>th</sup> percentile, median, 100-N <sup>th</sup> percentile, and the maximum. In this plot, there are no outside values, fences, adjacent values, etc. The minimum value is the end of one whisker, and the maximum value is the end of the other whisker. When N=10 this mode is identical to TENNINETY.
TENNINETY	This represents the minimum, 10 <sup>th</sup> percentile, median, 90 <sup>th</sup> percentile, and the maximum. In this plot, there are no outside values, fences, adjacent values, etc. The minimum value is the end of one whisker, and the maximum value is the end of the other whisker.

#### Default

STANDARD



### *Used in These Charts*

Box

## **PolarLabel**

---

```
PolarLabel = (mode, Color, FontName, FontSize, Angle, interiorAlignment);
```

`PolarLabel` defines the presentation format for optional numeric labels for the exterior of a polar chart.

### *Used in These Charts*

Polar

### *Example:*

```
PolarAxis = (ON, black, "TimesRoman", 16, 0, LEFT);
```

### *Attributes*

<i>Mode</i>	<i>Color</i>	<i>FontName</i>	<i>FontSize</i>	<i>Angle</i>	<i>interiorAlignment</i>
-------------	--------------	-----------------	-----------------	--------------	--------------------------

### *Mode*

---

`Mode` determines whether or not the labels are shown.

### *Values*

ON     Show tic labels for this axis  
OFF    Don't show tic labels on this axis

### *Default*

ON

### *interiorAlignment*

---

`interiorAlignment` dictates the alignment of text lines in multi-line labels.

### *Values*

LEFT  
RIGHT  
CENTER

### *Default*

CENTER

## **PolarLabelFormat**

---

```
PolarLabelFormat = (dataType, formatString);
```

The `PolarLabelFormat` parameter defines the format for displaying exterior labels on the polar chart

### *Used in These Charts*

Polar

### *Example:*

```
PolarLabelFormat = (INTEGER, "%3d\u00B0");
```

### *Attributes*

*DataType*      *FormatString*

### *DataType*

---

The type of number to use when formatting the labels

### *Values*

FLOAT  
INTEGER  
DECIMAL

## **PolarLabelStep**

---

```
PolarLabelStep = stepsize;
```

`PolarLabelStep` defines the step size for the exterior polar chart labels. For example, a value of 90 would cause labels to be drawn at 0, 90, 180 and 270 degrees around the exterior of the polar chart.

### *Used in These Charts*

Polar

### *Example:*

```
PolarLabelStep = 90;
```

## **PolarScale**

---

```
PolarScale = (minValue, maxValue, stepSize) ...;
```

The `PolarScale` parameter defines the scales for each of the axes in the `PolarChart`. All axes in a polar chart have the same scale. If `PolarScale` is not specified, or only certain attributes are specified, the axes will automatically choose values for the unspecified attributes.

### *Used in These Charts*

Polar

**Example:**

```
PolarScale = (0,10,1);
```

**Attributes**

*MaxValue*      *MinValue*      *StepSize*

---

**PolarSquare**

---

```
PolarSquare = ON|OFF;
```

The `PolarSquare` parameter tells the polar chart that the data area of the chart should be kept as high as it is wide.

**Used in These Charts**

Polar

**Example:**

```
RadarSquare = ON;
```

---

**PolarSize**

---

```
PolarSize = (minWidth, minHeight, maxWidth, maxHeight) ;
```

The `PolarSize` parameter can be used to set minimum and maximum sizes for the axes area in a polar chart. This allows programmers to guarantee that the axes area will always be the same size regardless of the length of the strings in the legend or tic labels. `PolarSize` has the following interaction with `PolarSquare`; if the minimum or maximum dimensions specified are not square, and `RadarSquare` is ON, then the axes area will be inscribed in a square with a dimension that ranges from the smallest of the minimum values to the smallest of the maximum values.

**Used in These Charts**

Polar

**Example:**

```
PolarSize = (100,100,200,200);
```

**Attributes**

*minWidth*, *minHeight*, *maxWidth*, *maxHeight*

---

**PolyActiveLabels**

---

```
PolyActiveLabels = ("Label1", "URL1", "Target1"),...;
```

`PolyActiveLabels` define the active labels associated with defined polygons.

**Used in These Charts**

Diagram Chart

**Example:**

```
PolyActiveLabels = ("west region","javascript:doSelection()");
```

**Default**

Use tag name as label

**Attributes***Label, URL, Target*

---

**PolyColor**

---

```
PolyColor = ("Tag", "Color"),...;
```

PolyColor associates a color with the named polygon, causing that polygon to be filled with the given color.

**Used in These Charts**

Diagram Chart

**Example:**

```
PolyColor = ("MD","green_130"), ("TX","green_200");
```

**Default**

None

**Attributes***TagName, Color*

---

**PolySet**

---

```
PolySet = ("Tag", x1,y1,x2,y2,...),...;
```

PolySet defines a polygon with a "tag" name and a series of X/Y points.

**Used in These Charts**

Diagram Chart

**Example:**

```
PolySet = ("A", 120,120,180,120,180,160,120,160,120,120),  
("B", 220,220,280,220,280,260,220,260,220,220);
```

**Default**

None

**Attributes***TagName, X/Y Pairs*

---

## RadarSize

---

```
RadarSize = (minWidth, minHeight, maxWidth, maxHeight) ;
```

The RadarSize parameter can be used to set minimum and maximum sizes for the axes area in a radar chart. This allows programmers to guarantee that the axes area will always be the same size regardless of the length of the strings in the legend or tic labels. RadarSize has the following interaction with RadarSquare; if the minimum or maximum dimensions specified are not square, and RadarSquare is ON, then the axes area will be inscribed in a square with a dimension that ranges from the smallest of the minimum values to the smallest of the maximum values.

### *Used in These Charts*

Radar

### *Example:*

```
RadarSize = (100,100,200,200) ;
```

### *Attributes*

*minWidth, minHeight, maxWidth, maxHeight*

---

## RadarSquare

---

```
RadarSquare = ON|OFF;
```

The RadarSquare parameter tells the radar chart that the appearance of the CenterRadius should be kept as high as it is wide.

### *Used in These Charts*

Radar

### *Example:*

```
RadarSquare = ON;
```

---

## RadialAxes

---

```
RadialAxes = ("axisTitle", minValue, maxValue, stepSize)...;
```

The RadialAxes parameter defines each of the axes in the chart.

### *Used in These Charts*

Radar

### *Example:*

```
RadialAxes = RadialAxes = ("Metric1", 0, 100, 25, black),  
    ("Metric2", 50, 100, 25, black), ("Metric3", 0, 200, 50, black),
```

```
("Metric4", 60, 90, 10, black), ("Metric5"0, 95, 15, black);
```

### *Attributes*

*AxisTitle*      *MaxValue*      *MinValue*      *StepSize*

---

## RadialAxesAngles

---

```
RadialAxesAngles = angle1, angle2 ...;
```

The `RadialAxesColors` parameter controls the drawing angle of the axes on a Polar chart. An angle of 0 degrees draws an axis straight up from the center of the chart.

### *Used in These Charts*

Polar

### *Example:*

```
RadialAxesAngles = 0, 90, 180, 270;
```

---

## RadialAxesColors

---

```
RadialAxes = color1, color2 ...;
```

The `RadialAxesColors` parameter defines the colors for each of the axes in the chart.

### *Used in These Charts*

Radar, Polar

### *Example:*

```
RadialAxes = black, black, red, black;
```

---

## RadialAxesFormat

---

```
RadialAxesFormat = (dataType, formatString),...;
```

The `RadialAxesFormat` parameter defines the format for displaying radial axis tic labels.

### *Used in These Charts*

Radar, Polar

### *Example:*

```
RadialAxesFormat = (INTEGER, "%3d"), (FLOAT, "%f"), (INTEGER, "%3d");
```

**Attributes**

*DataType*      *FormatString*

**DataType**

Value that specifies the type of numeric format to use to label the axes tics. Legal values include INTEGER, FLOAT and DECIMAL.

**FormatString**

If the `dataType` attribute is INTEGER or FLOAT, the input data value is expected to be of type integer or float and will be parsed as such (if string conversion is necessary). The format itself is a C-language style `sprintf` format. Some examples:

Data	Type	Format	Output
1000	INTEGER	%d	1000
1000	INTEGER	%, %d	\$1,000
1000	INTEGER	%d%	1000%
1000	FLOAT	%f	1000.0
1000	FLOAT	%.2f	1000.00
1000	FLOAT	%, .2f	\$1,000.00

If the format type is DECIMAL, the format syntax is consistent with those defined in the Java `DecimalFormat` spec.

**RadialAxesLabel**

```
RadialAxesLabel = ("Mode", Color, Font, Label), ...;
```

`RadialAxesLabel` controls the appearance of all radial axes labels in the chart. One radial axis label may be displayed per axis. The label is drawn at the end of the axis.

**Used in These Charts**

Radar

**Example:**

```
RadialAxesLabel = ("ON", black, "sansserif", 10, 0), ("ON", red, "sansserif", 12, 0);
```

**Attributes**

*Font*      *Color*      *Label*      *Mode*

**RadialAxesLabels**

```
RadialAxesLabels = ("Axis1Label1", Axis1Label2...), (Axis2Label1, Axis2Label2...) ...;
```

`RadialAxesLabels` specifies the text to use as axis tic labels for each axis.

### *Used in These Charts*

Radar

#### **Example:**

```
RadialAxesLabels = ("Jan", "Feb", "Mar"), ("inbound", "outbound", "returned");
```

## RadialAxesScales

---

```
RadialAxesScales = (minValue, maxValue, stepSize) ...;
```

The `RadialAxesScales` parameter defines the scales for each of the axes in the chart. This parameter was introduced in NetCharts 4.6

### *Used in These Charts*

Radar

#### **Example:**

```
RadialAxesScales = (25, 40, 5), (0, 100, 10);
```

#### **Attributes**

*MaxValue*      *MinValue*      *StepSize*

## RadialAxesTics

---

```
RadialAxesTics = ("axisTicLabelMode", axisTicLabelColor, "axisTicLabelFont", axisTicLabelFontAngle) ...;
```

The `RadialAxesTics` parameter specifies the label attributes for the tic marks displayed along the defined axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the `RadialAxesTics` parameter. If any attribute is not defined, any previous value of the attribute will be used.

### *Used in These Charts*

Radar, Polar

#### **Example:**

```
RadialAxesTics = ("ON", black, "sansserif", 9, 0), ("ON", red, "sansserif", 12, 0);
```

#### **Attributes**

*Font*      *FontAngle*      *Color*      *Mode*

## RadialAxesTitles

---



```
RadialAxesTitles = "Title1","Title2","Title3"...;
```

The `RadialAxesTitles` parameter defines the title for each of the axes in the chart. This parameter was introduced in NetCharts 4.6

### *Used in These Charts*

Radar

### *Example:*

```
RadialAxesTitles = "Inbound","Outbound","Undelivered";
```

---

## RadialAxesTitleActiveLabels

---

```
RadialAxisTitleActiveLabels = ("Label","URL","Target"), ...;
```

The `RadialAxesTitleActiveLabels` parameter specifies a custom active label to be associated with the radial axis title. That is, these labels will be displayed whenever the mouse “dwells” over the specified radial axis title.

### *Used in These Charts*

Radar

### *Example:*

```
RadialAxesTitleActiveLabels = ("Metric 1",,);
```

### *Attributes*

*Label*      *Target*      *URL*

---

## RadialGrids

---

```
RadialGrids = (gridRadius, gridLineType, gridLineWidth, gridLineColor, gridAreaColor), ...;
```

The `RadialGrids` parameter supports the display of one or more circular grids behind the data.

### *Used in These Charts*

Radar, Polar

### *Example:*

```
RadialGrids = (25,SOLID,1,black,white), (50,SOLID,1,black,white);
```

### *Attributes*

*GridRadius*, *GridLineType*, *GridLineWidth*, *GridLineColor*, *GridAreaColor*



**Used in These Charts**

Combo

**Example:**

```
RightAxis = ("Milliseconds", black, "TimesRoman", 16, 0);
```

**Attributes**

*Label*    *Color*    *FontName*    *FontSize*    *Angle*    *interiorAlignment*

**RightAxisTitle**

```
RightAxisTitle[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);
```

The `RightAxisTitle` parameter specifies the label attributes for the axis title, which centered along the top axis, just above the grid. When the `Header` parameter, whether because of the use of a legend, or for some other reason, creates a title that seems visually unbalanced, you may find this parameter produces a more pleasing chart title.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.

The legal values for `interiorAlignment` and `exteriorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
RightAxisTitle = ("Ceres Prototype Project Schedule\n ", black, "Helvetica", 12);
```

**Attributes**

*Label*    *Color*    *FontName*    *FontSize*    *Angle*    *interiorAlignment*    *exteriorAlignment*

**RightAxisTitleActiveLabel**

```
RightAxisTitleActiveLabel = ("Label", "URL", "Target");
```

`RightAxisTitleActiveLabel` defines a single active label destination for the `RightAxisTitle` parameter.

**Used in These Charts**

All

**Example:**

```
RightAxisTitleActiveLabel = ("Destination", "demo.html", "frame1");
```



**Example:**

```
RightDrawMinorTics = OFF;
```

**Attributes**

(Switch)

**RightFormat**

```
RightFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
```

`RightFormat` adjusts the numeric labels that are automatically generated for the top axis, should one be defined.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
RightFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
RightFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
RightFormat = (INTEGER);
RightFormat = (FLOAT, "$%,9.2f", ,);
```

**Attributes**

FormatType	FormatExpr	TimeBase	TimeUnit
------------	------------	----------	----------

**FormatType**

`FormatType` specifies the type of number being displayed on the top axis.

**Example:**

```
RightFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
RightFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
RightFormat = (INTEGER);
RightFormat = (FLOAT, "$%,9.2f", ,);
```

**Values**

DATE	Axis value are shown as date and/or time values. See <b>Appendix A: Date and Time Values</b> for further detail.
FLOAT	Axis values are shown with decimal parts.
INTEGER	Axis values are shown as integers, and are rounded if necessary.

**Default**

INTEGER

**TimeBase**

The `TimeBase` attribute specifies the base date to be used when determining the actual date or time value when using a time unit or numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
RightFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
RightFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

**Values**

String values representing dates or times

**Default**

None

**TimeUnit**

---

The `TimeUnit` attribute controls the time multiplier to be used when determining the actual data/time value when using a numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
RightFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
RightFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

**Values**

String values representing dates or times

**Default**

None

## RightLabels

---

```
RightLabels = "Label1", "Label2", ...;
```

The `RightLabels` parameter specifies a list of custom tic mark labels that will be used instead of the numeric labels automatically generated by the axis. The `RightLabels` will be evenly placed along the axis, overriding any tic placement specified by the `StepValue` attribute.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
RightLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie",  
             "Steve", "Paul";
```

**Attributes**

`Label`

---

## RightMargins

---

```
RightMargins = (BottomSideMargin, TopSideMargin);
```

The `RightMargins` parameter specifies the gap, in pixels, at the beginning and end of the right axis. Most often used to prevent clipping of data points at the extreme ends of the scale.

**Example:**

```
RightMargins = (20, 20);
```

---

## RightScroll

---

```
RightScroll = (ScrollMin, ScrollMax);
```

The `RightScroll` parameter specifies a range of values through which an axis can be scrolled. When the `ScrollMin` and `ScrollMax` attributes are defined for the axis, the axis will be displayed as a slider bar, using the axis color defined, with a white background. The relative size of the slider represents the percentage of the entire range currently being displayed. That is, it graphically depicts the size of the current axis range (`MinValue` and `MaxValue` attributes) relative to the scrollable region (`ScrollMin` and `ScrollMax` attributes). See the `RightScale` parameter for `MinValue` and `MaxValue` definitions.

`RightScroll` should only be used in conjunction with the `RightScale` parameter.

**Example:**

```
RightScroll = (0, 98);
```

**Attributes**

```
ScrollMin          ScrollMax
```

**ScrollMin**

`ScrollMin` sets the lower visible limit for a scrollbar defined with `RightScroll`

**Example:**

```
RightScroll = (0, 98);
```

**Values**

```
<MinValue
```

**Default**

```
None
```

**ScrollMax**

---

`ScrollMax` sets the upper visible limit for a scrollbar defined with `RightScroll`

**Example:**

```
RightScroll = (0, 98);
```

**Values**

>MaxValue

**Default**

None

---

**RightTicLength**

---

```
RightTicLength = Number;
```

The `RightTicLength` parameter defines the size of axis tic marks which are displayed along the right axis of a chart. The parameter will reset the automatically generated tic length. The value defines the number of pixels to use for the length of the tic mark. By default, the number of pixels used is the width of the character zero (0) as found in the font applied to the label. Setting the `RightTicLength` to the value -1 will cause the default size to be used.

**Attributes**

Number

**Number**

---

Apparent length of a right axis tic mark in a chart, in pixels.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
RightTicLength = 10;
```

**Values**

0	No effect (zero length tics are not drawn).
1 or greater	Whole number length in pixels

**Default**

-1

---

**RightTics**

---

```
RightTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);
```

The `RightTics` parameter specifies the label attributes for the tic marks displayed along the right axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the `RightTics` parameter. If any attribute is not defined, any previous value of that attribute will be used.



interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
backgroundColor	Background color of the tic label area
rotationPoint	For rotated axis labels, anchor point for the rotation

The legal values for `interiorAlignment` are LEFT, RIGHT, or CENTER.

The legal values for `rotationPoint` are LEFT, RIGHT.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
RightTics = ("OFF", black, "Helvetica", 10);
```

### *Attributes*

```
Mode      Color      FontName  FontSize  Angle  interiorAlignment
InteriorAlignment  BackgroundColor  RotationPoint
```

### *Mode*

Mode determines whether or not the tic labels are shown on that axis.

### *Example:*

```
RightTics = ("OFF", black, "Helvetica", 10);
```

### *Values*

```
ON      Show tic labels for this axis
OFF     Don't show tic labels on this axis
```

### *Default*

```
ON
```

## **RightTitle**

```
RightTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment);
```

`RightTitle` describes an optional title, or label, that sits on the left side of a chart, and uses standard attributes for string text, text color, font, font size, and label rotation. As with `Header`, `RightTitle` is universally available in Visual Mining chart applications.

### *Used in These Charts*

All, but most useful in Pie, Diagram, and Dial, which do not have axes.

### *Example:*

```
RightTitle = ("Financial Status", royalblue, Helvetica, 14, 0, LEFT);
```

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.

The legal values for `interiorAlignment` and `exteriorAlignment` are LEFT, RIGHT, or CENTER.

### Attributes

`Label`      `Color`              `FontName`      `FontSize`      `Angle`      `interiorAlignment`  
`exteriorAlignment`

## RightTitleActiveLabel

---

```
RightTitleActiveLabel = ("Label", "URL", "Target");
```

`RightTitleActiveLabel` defines a single active label destination for the `RightTitle` parameter.

### Used in These Charts

All

### Example:

```
RightActiveLabel = ("Destination", "demo.html", "frame1");
```

### Attributes

`Label`                      `URL`                      `Target`

## RightTitleBox

---

```
RightTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `RightTitleBox` specifies a background region just for the `RightTitle` parameter.

### Used in These Charts

All

### Attributes

`BorderColor`              `BorderType`              `BorderWidth`              `Color`  
`ImageFormat`              `ImageURL`              `TRCornerStyle`              `BRCornerStyle`  
`BLCornerStyle`              `CornerColor`

### XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

---

## RubberbandBorderStyle

---

RubberbandBorderStyle = (*linetype*, *linewidth*, *linecolor*);

Defines the borderstyle to apply to the rubberbanded box created by clicking and dragging to create zoom rectangle.

### *Used in These Charts*

All

### *Example:*

```
RubberbandBorderStyle = (SOLID,1,YELLOW);
```

### *Attributes*

*LineType*                      *LineWidth*                      *LineColor*

---

### *LineType*

#### *Values*

SOLID	A solid line is displayed (default).
DOTTED	A dotted line is displayed.
DASHED	A dashed line is displayed.
DOTDASH	Alternating dots and dashes are displayed.

#### *Default*

SOLID

---

### *LineWidth*

The width of the line defaults to 1 pixel.

---

### *LineColor*

The LineColor defaults to black.

---

## RubberbandFill

---

RubberbandFill = *Color*;

Defines the color of the rubberbanded box created by clicking and dragging a zoom rectangle.

### *Used in These Charts*

All



```
RightScale[N] = mode, ...;
```

The `Scale` parameter specifies the minimum and maximum data values which will be displayed along the axis. If the `Scale` parameter is not defined then the tic mark locations will be automatically determined based on the actual data values being displayed. That is, the axis will be “autoscaled” using the current data values to determine “reasonable” values for `MinValue`, `MaxValue` and `StepValue`.

Any combination of `MinValue`, `MaxValue` and `StepValue` may be defined. Those attributes that are not defined will have “reasonable” values chosen for them.

If the `StepValue` is defined but not as an even multiple of the difference between the `MinValue` and `MaxValue`, then no tic mark will be displayed at the `Max Value`.

---

## ScaleFactor

---

```
ScaleFactor = Number;
```

The number can be -1 or any value, including fractional values, greater than or equal to 0.

Specify a scale factor of -1 to have the chart autoscale. Autoscaling causes the chart's text, line widths, borders, and images to grow in proportion to the size of the chart. For instance, if you increased the size of the chart by 2, then the font sizes would all increase by 2.

The NetCharts Designer product refers to the `ScaleFactor` parameter as `ChartScaleFactor`.

A scale factor of 0 indicates that the chart should disable scaling. All other positive values indicate the percentage of their original size to which the components of the chart (text, line width, borders, etc.) should be scaled. That is, a value of 1.2 indicates that the components of the chart should be 120% their normal size, whereas a value of .3 means that the components should assume a size 30% that of their normal size.

If no `ScaleFactor` parameter is set in the CDL definition, the chart defaults to scaling disabled (`ScaleFactor = 0`).

### *Attributes*

*Number*

---

## ScaleMode

---

```
TopScaleMode[N] = (mode,logBase), (mode,logBase)...;  
BottomScaleMode[N] = (mode, logBase), (mode,logBase), ...;  
LeftScaleMode[N] = (mode, logBase), (mode, logBase), ...;  
RightScaleMode[N] = (mode,logBase), (mode, logBase), ...;
```

The `ScaleMode` parameter is used to specify which scale mode should be used on a given axis. The default value for the `ScaleMode` parameter is `LINEAR`.

### *Example:*

```
TopScaleMode = (LOG,16);
```

### Values

LINEAR    Linear (no logBase is used)  
LOG        Logarithmic (Use a Logarithmic scale mode with log base = logBase)

### Attributes

Mode       logBase

## ScaleSet

---

TopScaleSet[N] = (Min, Max, Step, Percentage), ...;  
BottomScaleSet[N] = (Min, Max, Step, Percentage), ...;  
LeftScaleSet[N] = (Min, Max, Step, Percentage), ...;  
RightScaleSet[N] = (Min, Max, Step, Percentage), ...;

The `ScaleSet` defines the minimum and maximum values of a scale and percentage of total space allocated to the axis.

### Example:

```
ScaleSet = (Min, Max, Step, Percentage);
```

### Attributes

## Scroll

---

TopScroll[N] = (ScrollMin, ScrollMax);  
BottomScroll[N] = (ScrollMin, ScrollMax);  
LeftScroll[N] = (ScrollMin, ScrollMax);  
RightScroll[N] = (ScrollMin, ScrollMax);

The `Scroll` parameter specifies a range of values through which an axis can be scrolled. When the `ScrollMin` and `ScrollMax` attributes are defined for the axis, the axis will be displayed as a slider bar, using the axis color defined, with a white background. The relative size of the slider represents the percentage of the entire range currently being displayed. That is, it graphically depicts the size of the current axis range (`MinValue` and `MaxValue` attributes) relative to the scrollable region (`ScrollMin` and `ScrollMax` attributes). See the `Scale` parameter for `MinValue` and `MaxValue` definitions.

`Scroll` should only be used in conjunction with the `Scale` parameter.

### Example:

```
TopScroll = (0, 98);
```

### Attributes

ScrollMin                      ScrollMax

### ***ScrollMin***

`ScrollMin` sets the lower visible limit for a scrollbar defined with `TopScroll`

**Example:**

```
TopScroll = (0, 98);
```

**Values**

<MinValue

**Default**

None

### ***ScrollMax***

---

`ScrollMax` sets the upper visible limit for a scrollbar defined with `TopScroll`

**Example:**

```
TopScroll = (0, 98);
```

**Values**

>MaxValue

**Default**

None

## **SectorActiveLabels[n]**

---

```
SectorActiveLabels[N] = (Name, Label, URL, Target), ...;
```

`SectorActiveLabels` defines a list of up to 50 active label destinations for named sectors within a dial, in a dial chart. These labels map to the named sectors that were defined using the `Sectors` parameter.

**Used in These Charts**

Dial

**Example:**

```
SectorActiveLabels = ("Danger", "", "dailysales.html", "infoframe"),  
                    ("Warning", "", "dailysales.html", "infoframe"),  
                    ("On Track", "", "dailysales.html", "infoframe");
```

**Attributes**

<i>Name</i>	<i>Label</i>	<i>URL</i>	<i>Target</i>
-------------	--------------	------------	---------------

## SectorBorders

---

```
SectorBorders = (Name, Type, LineWidth, Color), ...;
```

`SectorBorders` controls the appearance of the borders around a named sector within a dial. Note that the fill color for the sector is designated in the `Sectors` parameter.

### Used in These Charts

Dial

#### Example:

```
SectorBorders = ("Danger\nZone", NONE, 0, null);
```

#### Attributes

Name	Type	LineWidth	Color
------	------	-----------	-------

## SectorColors

---

```
SectorColors = (Name, Color), ...;
```

`SectorColors` controls the background color of a sector.

### Used in These Charts

Dial

#### Example:

```
SectorColors = ("Sector1", green), ("Sector2", yellow);
```

## SectorData

---

```
SectorData = (Name, StartValue, StopValue), ...;
```

`SectorData` controls the point at which named sectors in a dial start and stop, *relative to the measurements on the dial*.

### Used in These Charts

Dial

#### Example:

```
SectorData = ("Danger", 0, 5),
             ("Warning", 5, 8),
             ("On Track", 8, 10),
             ("Low", 0, 3.25),
             ("Medium", 3.25, 6.75),
             ("High", 6.75, 10);
```

#### Attributes

Name	StartValue	StopValue
------	------------	-----------



## *StartValue*

---

StartValue shows the angular place where the sector begins.

### *Example:*

```
SectorData = ("Danger", 0, 5),  
             ("Warning", 5, 8),  
             ("On Track", 8, 10),  
             ("Low", 0, 3.25),  
             ("Medium", 3.25, 6.75),  
             ("High", 6.75, 10);
```

### *Values*

Numeric, but depends on how the dial setting are set in the DialScale parameter.

### *Default*

None

## *StopValue*

---

StopValue shows the angular place where the sector ends.

### *Example:*

```
SectorData = ("Danger", 0, 5),  
             ("Warning", 5, 8),  
             ("On Track", 8, 10),  
             ("Low", 0, 3.25),  
             ("Medium", 3.25, 6.75),  
             ("High", 6.75, 10);
```

### *Values*

Numeric, but depends on how the dial setting are set in the DialScale parameter.

### *Default*

None

## **SectorDelete**

---

```
SectorDelete = (Name|ALL), ...;
```

SectorDelete is used to delete a specific sector, or all sector, in a dial.

### *Used in These Charts*

Dial

### *Example:*

```
SectorDelete = ("Low"), ("Medium");  
SectorDelete = ALL;
```

### *Attributes*

(Switch)

### *Switch*

---

This switch allows you to either name the sectors that are to be deleted, or to delete all at once.

#### *Example:*

```
SectorDelete = ("Low"), ("Medium");  
SectorDelete = ALL;
```

#### *Values*

Name	A string that names a hand
ALL	All the hands

#### *Default*

No defaults

## SectorDrag

---

```
SectorDrag = "ON"/"OFF";
```

The `SectorDrag` switch is used to allow or stop the user from dragging the sectors of a dial with the mouse.

#### *Used in These Charts*

Dial

#### *Example:*

```
SectorDrag = "ON";  
SectorDrag = "OFF";
```

### *Attributes*

(Switch)

### *Switch*

---

This switch sets the on/off state.

#### *Example:*

```
SectorDrag = "ON";  
SectorDrag = "OFF";
```

#### *Values*

ON	Allows the user to drag the sector around the dial
OFF	Stops the user from dragging the sector around the dial

**Default**

OFF

**SectorEdgeHighlights**

```
SectorEdgeHighlights = (start,stop,gap,size), ...;
```

The `SectorEdgeHighlights` parameter provides a visual pattern fill in a Dial chart which accents the dial sectors. It overlays a ring (annulus) fill pattern over the existing fill patterns in a specified zone along the interior edge of the sectors. The gap between the sides of the center button and the fill pattern being applied can be modified. The element `start` sets the beginning color of the gradient, associated with the outside edge; the element `stop` sets the end color of the gradient, associated with the interior of the sectors where the color blends to transparency. Color values are interpolated between the two. The element `size` specifies the width of the highlight. The element `gap` specifies the size of the gap between the edge of the highlight and the edge of the sectors. When the value for `gap` is specified as a whole number, it sets the distance between the edge of the highlight and the edge of the center button in pixels. When set to a fractional number between 0 and 1, it sets the gap to a percentage of the radius of the pie. Percentage values are written using a decimal point (e.g. “0.05” and “0.50” represent 5% and 50%, respectively). When using a fractional value, enclose the tuple element in double-quotes to “escape” the decimal point.

**Used in These Charts**

Dial

**Example:**

```
SectorEdgeHighlights = (blue_25,white_75,1,25), ...;
```

**Attributes**

```
start                stop                gap                size
```

**SectorFillPattern**

```
SectorFillPattern = (type, color1, color2, imageURL), ...;
```

The `SectorFillPattern` parameter provides a visual pattern fill for the sector area of a dial chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient

	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
<b>Images</b>		
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

**Used in These Charts**

Dial

**Example:**

```
SectorFillPattern = (GRADIENVERTICAL,blue,white);
```

**Attributes**

Type	Color1	Color2
ImageFormat	ImageURL	

**SectorLabels**

```
SectorLabels = (Name, ON/OFF, LabelPos, Color, FontName, FontSize, Angle,interiorAlignment) , ...;
```

SectorLabels works identically to the DialTicLabelStyles parameter, and controls the specific appearance and style of the labels for named sectors defined with the Sectors parameter.

interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
-------------------	---

The legal values for interiorAlignment are LEFT, RIGHT, or CENTER.

**Used in These Charts**

Dial

**Example:**

```
SectorLabels = ("Danger", "ON", 0.6, white, "Helvetica", 10, 0),
               ("Warning", "ON", 0.6, black, "Helvetica", 10, 0),
               ("On Track", "ON", 0.6, white, "Helvetica", 10, 0),
               ("Low", "ON", 1.2, black, "Helvetica", 10, 0),
               ("Medium", "ON", 1.2, black, "Helvetica", 10, 0),
               ("High", "ON", 1.2, black, "Helvetica", 10, 0);
```

**Attributes**

<i>Name</i>	(Switch)	<i>LabelPos</i>	<i>Color</i>	<i>FontName</i>
<i>FontSize</i>	<i>Angle</i>	<i>interiorAlignment</i>		

**Switch**

This switch turns the tic labels' visibility on and off.

**Example:**

```
DialTicLabelStyles = ("Hour Dial", "ON", 1.1, black, "Helvetica", 14, 0),
                    ("Minute Dial", "ON", 1.1, black, "Helvetica", 14, 0);
```

**Values**

ON      Show the tic labels  
OFF     Hide the tic labels

**Default**

None

**LabelPos**

The `LabelPos` attribute in the dial chart works the same way the `LabelPos` parameter does for the pie chart: it defines the position of the tic mark labels relative to the width of the dial. A value of 1.1 will place each label just outside the dial face, while a value of 0.6 will place each label inside of the dial face. For instance, clocks are often made with their tic mark labels on the outside of the dial face, but gauges tend to have them just on the inside of the dial face along with the tic marks.

**Example:**

```
DialTicLabelStyles = ("Hour Dial", "ON", 1.1, black, "Helvetica", 14, 0),
                    ("Minute Dial", "ON", 1.1, black, "Helvetica", 14, 0);
```

**Values**

Positive real numbers, generally between 0 and 2.

**Default**

None

## Sectors

Sectors = (Name, Color, DialName, OuterRadius, InnerRadius, SectorLabel), ...;

Dial charts can not only have multiple dials, but these dials can be divided up into sectors. We are used to sectors in gauges, where one typically finds green-yellow-red sectors indicating danger or quality levels. The performance scorecard dial chart (see figure below) incorporates them in several ways.

The `Sectors` parameter identifies and colors the sector within a named dial, and places the sector radially in that dial.

### Used in These Charts

Dial

#### Example:

```
Sectors = ("Danger", red, "Internal", 80, 20),
("Warning", yellow, "Internal", 80, 20),
("On Track", green, "Internal", 80, 20),
("Low", black, "External", 100, 80),
("Medium", gray, "External", 100, 80),
("High", white, "External", 100, 80);
```

### Attributes

Name	Color	DialName	OuterRadius
InnerRadius			

### DialName

DialName is the name of the dial in which the sector resides.

#### Example:

```
Sectors = ("Danger", red, "Internal", 80, 20),
("Warning", yellow, "Internal", 80, 20),
("On Track", green, "Internal", 80, 20),
("Low", black, "External", 100, 80),
("Medium", gray, "External", 100, 80),
("High", white, "External", 100, 80);
```

### Values

Any string value

### Default

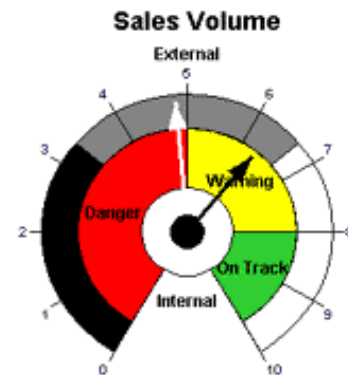
None

### OuterRadius

OuterRadius controls the outer limit of the sector as a percentage of the dial's radius.

#### Example:

```
Sectors = ("Danger", red, "Internal", 80, 20),
("Warning", yellow, "Internal", 80, 20),
("On Track", green, "Internal", 80, 20),
("Low", black, "External", 100, 80),
("Medium", gray, "External", 100, 80),
("High", white, "External", 100, 80);
```



**Values**

Numerical value from 0 to 100

**Default**

None

**InnerRadius**

---

InnerRadius controls the inner limit of the sector as a percentage of the dial's radius.

**Example:**

```
Sectors = ("Danger", red, "Internal", 80, 20),
          ("Warning", yellow, "Internal", 80, 20),
          ("On Track", green, "Internal", 80, 20),
          ("Low", black, "External", 100, 80),
          ("Medium", gray, "External", 100, 80),
          ("High", white, "External", 100, 80);
```

**Values**

Numerical value from 0 to 100

**Default**

None

**SectorLabel**

---

SectorLabel specifies an optional text label to display along with the sector. The label's appearance is controlled by the SectorLabels parameter.

**ShowEightyTwentyLines**

---

```
ShowEightyTwentyLines = ON / OFF;
```

This parameter defines whether 80/20 lines will be drawn on the Pareto chart. The default value is OFF.

**Used in These Charts**

Pareto

**Example:**

```
ShowEightyTwentyLines = ON
```

**ShowGroupStackLabels**

---

```
ShowGroupStackLabels = ON / OFF;
```

This parameter is used only for grouped stacked BarCharts (GraphType=GROUPSTACK). It defines whether stack labels will be drawn at each tic. The default value is OFF.

---

## ShowDataPoints

---

```
ShowDataPoints = ON | OFF;
```

ShowDataPoints is used to display the raw data in addition to the summary data for each series. The default is OFF

### *Used in These Charts*

Box Chart

### *Example:*

```
ShowDataPoints = ON;  
ShowDataPoints = OFF;
```

### *Default*

OFF

### *Attributes*

Mode

---

## SliceAnimationStyle

---

```
SliceAnimationStyle = GROW | FADE | NONE
```

Specifies how slices initially appear in a pie chart. This parameter is only valid in SVG or SVGWeb output formats.

### *Attributes*

Style

### *Style*

---

style refers to the manner in which lines are first rendered in a line chart.

### *Example:*

```
LineAnimationStyle = BEND;
```

### *Values*

- |      |  |
|------|--|
| GROW | The slices grow and spin their actual value. |
| FADE | The slices fade in.                          |
| NONE | The slices are immediately visible.          |

### *Default*

NONE



---

## SliceBorder

---

```
SliceBorder[N] = (LineType, Width, Color);
```

In pie charts, this parameter specifies the line style to be used for the border of all pie slices.

### *Used in These Charts*

Pie MultPie

### *Example:*

```
SliceBorder = (DOTTED, 2, blue);
```

### *Attributes*

<i>LineType</i>	<i>Width</i>	<i>Color</i>
-----------------	--------------	--------------

---

## SliceColors

---

```
SliceColors[N] = Value1, Value2, Value3, Value4, ..., Valuen;
```

In pie charts, this parameter specifies the line style to be used for the border of all pie slices.

### *Used in These Charts*

Pie

### *Example:*

```
SliceColor = red,white,blue;
```

### *Attributes*

*Color*

---

## SliceData

---

```
SliceData[N] = Value1, Value2, Value3, Value4, ..., Valuen;
```

`SliceData` provides a method for specifying pie chart slice data without using the `Slices` parameter, and consists of a vector of data which may represent either percentages or data values.

### *Used in These Charts*

Pie

### *Example:*

```
SliceData = 15, 33, 44, 102, 19.45, 88, 47.9;
```

**Attributes**

None

**SliceFillPattern**

`SliceFillPattern = (type, color1, color2, imageURL), ...;`

The `SliceFillPattern` parameter provides a visual pattern fill for pie slices in a chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient
	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
	<b>Images</b>	
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

**Used in These Charts**

Pie MultPie

**Example:**

```
SliceFillPattern = (GRADIENTVERTICAL,blue,white);
```

**Attributes**

<i>Type</i>	<i>Color1</i>	<i>Color2</i>
<i>ImageFormat</i>	<i>ImageURL</i>	

---

## SliceFormat

---

```
SliceFormat = (FormatType, "FormatExpr");
```

In pie charts, SliceFormat affects the numeric labels that are automatically generated for each slice.

### *Used in These Charts*

Pie MultPie

### *Example:*

```
SliceFormat = (FLOAT, "$%,9.2f");
```

### *Attributes*

FormatType                      FormatExpr

### *FormatType*

---

The FormatType attribute specifies the type of number being displayed by that axis

### *Example:*

```
SliceFormat = (FLOAT, "$%,9.2f");
```

### *Values*

FLOAT            Display numeric values with decimal points  
INTEGER         Display numeric values only as integers, and will round if necessary

### *Default*

FLOAT

### *FormatExpr*

---

The FormatExpr attribute specifies a numeric display format to be used for each slice label and each active label generated by default. The format allows a developer to specify numeric formats using expressions similar to those provided in the C/Unix printf function.

### *Example:*

```
SliceFormat = (FLOAT, "$%,9.2f");
```

### *Values*

For the specifics of numeric formats, see FormatExpr in **Chapter 4: Common CDL Attributes**.

### *Default*

%.2f

---

## SliceLabel

---

```
SliceLabel = (State, Color, FontName, FontSize, Angle, interiorAlignment);
```

`SliceLabel` controls the appearance of all the slice labels in a pie chart. This parameter is a standard “tuple.”

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

### *Used in These Charts*

Pie MultPie

### *Example:*

```
SliceLabel = (ON, Black, Helvetica, 12, 270, LEFT);
```

### *Attributes*

```
State    Color    FontName    FontSize    Angle    interiorAlignment
```

### *State*

---

`State` indicates if this parameter is in effect or not.

### *Example:*

```
SliceLabel = (ON, Black, Helvetica, 12, 270);
```

### *Values*

ON or null

### *Default*

ON

---

## SliceLabelBox

---

```
SliceLabelBox = (Color, BorderType, BorderWidth);
```

`SliceLabelBox` controls the appearance of the area under the slice label in a pie chart.

### *Used in These Charts*

Pie MultPie

### *Example:*

```
SliceLabelBox = (fuschia, RECESS, 4);
```

## Attributes

*Color*      *BorderType*      *BorderWidth*

## SliceLabelContent

---

```
SliceLabelContent = [Value1],[Value2],[Value3];
```

`SliceLabelContent` allows you to specify the contents of RADIAL or EXTERIOR pie labels. (LEGEND labels will only show the text label.) The `SliceLabelContent` is a comma-delimited list of up to three items that should appear in these labels.

### Used in These Charts

Pie MultPie

### Example:

```
SliceLabelContent = DATA,LABEL;
```

### Values

PERCENTAGE	Displays the slice percentage as nn.nn% (for backward compatibility)
PERCENTAGE_FLOAT	Displays the slice percentage as nn.nn%
PERCENTAGE_INT	Displays the slice percentage as nn%
DATA	The data value, as formatted in the <code>SliceFormat</code> parameter
LABEL	The label passed as part of the <code>Slices</code> or <code>SliceLabels</code> parameter.

### Default

LABEL

## SliceLabelContentDelimiter

---

```
SliceLabelContentDelimiter = "delimiter";
```

`SliceContentDelimiter` allows specification of the delimiter to use between content items. These delimiters can be any text, but will usually be commas, newlines (\n), or spaces.

### Used in These Charts

Pie MultPie

### Example:

```
SliceLabelContentDelimiter = ",";  
SliceLabelContentDelimiter = "rrr";  
SliceLabelContentDelimiter = "\n";  
SliceLabelContentDelimiter = " ";
```

### *Attributes*

None

## SliceLabelLine

---

```
SliceLabelLine = (LineStyle, LineWidth, Color);
```

`SliceLabelLine` controls the appearance of the lines connecting the exterior labels to the pie slice in a pie chart. If the color of the line is set to `NULL`, or left undefined, the color of each line will be the same as the slice to which it connects.

### *Used in These Charts*

Pie, MultiPie

### *Example:*

```
SliceLabelLine = (SOLID, 2, antiquewhite);
```

### *Attributes*

`LineStyle`    `LineWidth`    `Color`

## SliceLabels

---

```
SliceLabels = Label1, Label2, Label3, Label4, ..., Labeln;
```

`SliceLabels` provides a method for specifying pie chart slice labels without using the `Slices` parameter, and consists of a vector of labels that will be applied to each of the pie chart slices.

### *Used in These Charts*

Pie MultiPie

### *Example:*

```
SliceLabels = "Monday", "Tuesday", "Wednesday", "Thursday", "Friday";
```

### *Attributes*

`Label`

## SliceLabelStyle

---

```
SliceLabelStyle = Style;
```

`SliceLabelStyle` allows explicit specification of the label style for pie charts.

***Used in These Charts***

Pie MultiPie

***Example:***

```

SliceLabelStyle = EXTERIOR;
SliceLabelStyle = LEGEND;
SliceLabelStyle = RADIAL;

```

***Attributes***

Style

***Style***

---

Only one `Style` attribute is allowed for `SliceLabelStyle`.

***Example:***

```

SliceLabelStyle = EXTERIOR;
SliceLabelStyle = LEGEND;
SliceLabelStyle = RADIAL;

```

***Values***

EXTERIOR	Labels are put on the left and right sides of the pie.
LEGEND	Labels are placed in a Legend
RADIAL	Labels are put along the radius of each slice

If `SliceLabelStyle` isn't defined, but a `Legend` is, the style will be a `LEGEND`.

If the chart doesn't have `SliceLabelStyle`, but `LabelPos` is defined, the style will be `RADIAL`.

If no `SliceLabelStyle` is defined, but both `LabelPos` and `Legend` are, then the style shown will be `LEGEND`.

***Default***

EXTERIOR

**SlicePos**

---

```

SlicePos[N] = Position1, Position2, Position3, ...;

```

`SlicePos` defines the position of each pie slice relative to the width of the pie. Can be used to “pre-explode” slices from the rest of the pie.

***Used in These Charts***

Pie MultiPie

***Example:***

```

SlicePos = 0, 1.2, 1.5, 0, 0, 0, 0;

```

**Attributes**

Position

**Position**

The `Position` attribute may be expressed as a fractional number in the range 0.0 to 2.0, or as a percentage value in the range of 2 to 100. A value of 0.5 or 50.0 will move the slice center radially out 50 percent of the width of the pie. That is, any nonzero value will result in the slice being *exploded*.

**Example:**

```
SlicePos = 0, 1.2, 1.5, 2.0, 0, 0, 0;
SlicePos = 0, 80, 75, 100, 0, 0, 0;
```

**Values**

0 Slices all lay together in the pie  
 >0 and <= 2 Slices explode from pie  
 Or, 2 to 100 % amount slices explode from pie

**Default**

0

**Slices**

```
Slices[N] = (Value, SliceColor, Label, LabelColor, FontName, FontSize, LabelAngle, LabelBgColor,
LabelBgBorder), ...;
```

`Slices` has been deprecated. Use `SliceData`, `SliceLabels` and `SliceColors`.

`Slices` defines the set of slices for a pie chart, specifically the value and appearances for a slice. Each grouping of attributes addresses one of the pie's slices. You can define each slice individually, or slices will take their attributes from the last grouping defined before them.

Only the first two attributes, `Value` and `SliceColor`, need to be specified; all others will be assigned default values. For the first slice, the defaults will be based on system defaults. For all other slices, the value assigned to the previous slice for that attribute will be used as the default. In that way, you need only specify attributes for the first slice in order to control the attributes of all slices. The value `NULL` may be passed as a pie slice value, but has the same effect as a 0 slice value.

**Used in These Charts**

Pie

**Example:**

```
Slices = (2300, NULL, "Mon", black, "TimesRoman", 14),
(3700,, "Tue"),
(1200,, "Wed"),
(2500,, "Thu"),
(4300,, "Fri"),
(1900,, "Sat"),
(2700,, "Sun");
```



**Attributes**

Value	SliceColor	Label	LabelColor
FontName	FontSize	LabelAngle	LabelBgColor
LabelBgBorder			

**Value**

Value is the raw numeric data associated with the pie slice described in its group. Visual Mining pie charts automatically calculate percentages, so you do not need to figure this out for yourself. The percentage will be computed based on the total of all slice values. Using NULL as a Value creates a slice of zero value, and is equivalent to a 0 data value.

**Example:**

```
Slices = (2300, NULL, "Mon", black, "TimesRoman", 14),
          (3700,, "Tue"),
          (1200,, "Wed"),
          (2500,, "Thu"),
          (4300,, "Fri"),
          (1900,, "Sat"),
          (2700,, "Sun");
```

**Values**

Any real number.

**Default**

None

**SliceColor**

SliceColor determines the color of the pie slice, and is in all respects specified as any other Color attribute. If NULL is used as a value, the color is taken from either a specified ColorTable parameter, or the default system color table.

**Example:**

```
Slices = (2300, NULL, "Mon", black, "TimesRoman", 14),
          (3700,, "Tue"),
          (1200,, "Wed"),
          (2500,, "Thu"),
          (4300,, "Fri"),
          (1900,, "Sat"),
          (2700,, "Sun");

Slices = (2300, x6C5D94, "Mon", x6C5D94, "Helvetica", 12, 0, white, NONE, 0),
          (3700, x999966, "Tue", x999966, "Helvetica", 12, 0, white, NONE, 0),
          (1200, x315394, "Wed", x315394, "Helvetica", 12, 0, white, NONE, 0),
          (2500, x213321, "Thu", x213321, "Helvetica", 12, 0, white, NONE, 0),
          (4300, x00566F, "Fri", x00566F, "Helvetica", 12, 0, white, NONE, 0),
          (1900, x690931, "Sat", x690931, "Helvetica", 12, 0, xe3e3e3, NONE, 0),
          (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, xe3e3e3, NONE, 0);
```

**Values**

See the Color attribute in **Chapter 4: Common CDL Attributes** for details.

**Default**

blue

---

## LabelColor

---

`LabelColor` determines the color of the pie slice's label font, and is in all respects specified as any other `Color` attribute.

### Example:

```
Slices = (2300, NULL, "Mon", black, "TimesRoman", 14),
          (3700,, "Tue"),
          (1200,, "Wed"),
          (2500,, "Thu"),
          (4300,, "Fri"),
          (1900,, "Sat"),
          (2700,, "Sun");

Slices = (2300, x6C5D94, "Mon", x6C5D94, "Helvetica", 12, 0, white, NONE, 0),
          (3700, x999966, "Tue", x999966, "Helvetica", 12, 0, white, NONE, 0),
          (1200, x315394, "Wed", x315394, "Helvetica", 12, 0, white, NONE, 0),
          (2500, x213321, "Thu", x213321, "Helvetica", 12, 0, white, NONE, 0),
          (4300, x00566F, "Fri", x00566F, "Helvetica", 12, 0, white, NONE, 0),
          (1900, x690931, "Sat", x690931, "Helvetica", 12, 0, xe3e3e3, NONE, 0),
          (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, xe3e3e3, NONE, 0);
```

### Values

See the `Color` attribute in **Chapter 4: Common CDL Attributes** for details.

### Default

blue

---

## LabelAngle

---

`LabelAngle` determines the display angle of the pie slice's label font. In all other respects, it is identical with the common `Angle` attribute

### Example:

```
Slices = (2300, x6C5D94, "Mon", x6C5D94, "Helvetica", 12, 0, white, NONE, 0),
          (3700, x999966, "Tue", x999966, "Helvetica", 12, 90, white, NONE, 0),
          (1200, x315394, "Wed", x315394, "Helvetica", 12, 90, white, NONE, 0),
          (2500, x213321, "Thu", x213321, "Helvetica", 12, 90, white, NONE, 0),
          (4300, x00566F, "Fri", x00566F, "Helvetica", 12, 270, white, NONE, 0),
          (1900, x690931, "Sat", x690931, "Helvetica", 12, 270, gray, NONE, 0),
          (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, xe3e3e3, NONE, 0);
```

### Values

0 degrees counterclockwise from horizontal  
 90 degrees counterclockwise from horizontal  
 180 degrees counterclockwise from horizontal  
 270 degrees counterclockwise from horizontal

### Default

0

---

## LabelBgColor

---

`LabelBgColor` determines the color behind the pie slice's label font, and is in all respects specified as any other `Color` attribute.

**Example:**

```
Slices = (2300, x6C5D94, "Mon", x6C5D94, "Helvetica", 12, 0, white, NONE, 0),
          (3700, x999966, "Tue", x999966, "Helvetica", 12, 0, white, NONE, 0),
          (1200, x315394, "Wed", x315394, "Helvetica", 12, 0, white, NONE, 0),
          (2500, x213321, "Thu", x213321, "Helvetica", 12, 0, white, NONE, 0),
          (4300, x00566F, "Fri", x00566F, "Helvetica", 12, 0, white, NONE, 0),
          (1900, x690931, "Sat", x690931, "Helvetica", 12, 0, xe3e3e3, NONE, 0),
          (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, xe3e3e3, NONE, 0);
```

**Values**

See the `Color` attribute in **Chapter 4: Common CDL Attributes** for details.

**Default**

blue

---

**LabelBgBorder**

`LabelBgBorder` determines the width of the border of the pie slice's label, and is in all respects specified as any other common `BorderStyle` attribute.

**Example:**

```
Slices = (2300, x6C5D94, "Mon", x6C5D94, "Helvetica", 12, 0, white, NONE, 0),
          (3700, x999966, "Tue", x999966, "Helvetica", 12, 0, white, NONE, 0),
          (1200, x315394, "Wed", x315394, "Helvetica", 12, 0, white, NONE, 0),
          (2500, x213321, "Thu", x213321, "Helvetica", 12, 0, white, NONE, 0),
          (4300, x00566F, "Fri", x00566F, "Helvetica", 12, 0, white, NONE, 0),
          (1900, x690931, "Sat", x690931, "Helvetica", 12, 0, xe3e3e3, RAISED,
          0),
          (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, xe3e3e3, RAISED,
          0);
```

**Values**

NONE	No border
BOX	Simple Box outline
SHADOW	Shadow border
RAISED	Raised border
RECESS	Recessed border

**Default**

NONE

---

**SliceSet**

```
SliceSet = Value1, Value2, Value3, Value4, ..., Valuen;
```

`SliceSet` defines the numeric values for each slice set.

**Used in These Charts**

MultiPie

## SliceSets

---

```
SliceSets = ("Name", Color, "State");
```

`SliceSet` defines a list of `SliceSet` tuples with the following attributes in each tuple:

Name	Name assigned to this SliceSet.
Color	Slice Color.
State	Labels ON or OFF. Default is ON

### *Used in These Charts*

MultiPie

### *Example:*

```
SliceSets = ("Over 18", "x7996A1"), ("Under 18", "xADD6E6");
```

### *Attributes*

*Name*            *Color*            *State*.

## SliceSlide

---

```
SliceSlide = ON/OFF;
```

When `SliceSlide = ON`, clicking on a pie slice will cause the slice to slide in/out of the pie. This parameter is only valid in SVG or SVGweb output formats.

### *Example:*

```
SliceSlide = ON;
```

### *Attributes*

(Switch)

## StackDisplayOrder

---

```
StackDisplayOrder = BOTTOMUP/TOPDOWN;
```

`StackDisplayOrder` defines the ordering of legend items in a chart with multiple bar series. The default is `BOTTOMUP`, which specifies that the legend items will be displayed in the order in which the data sets are specified. For example `DataSet1`, will appear in the legend first. `TOPDOWN` is useful when multiple series of bars are stacked and the legend is displayed vertically. In this mode, the stack of legend items will be "stacked" in the same order as the bars in the chart.

### *Used in These Charts*

Bar, Combo, Pareto, Stock

**Example:**

```
StackDisplayOrder = TOPDOWN;
```

## StackedBarConnectors

---

```
StackedBarConnectors = OFF | LINE | FILL;
```

`StackedBarConnectors` defines the bars in a particular series can be connected together with lines. This improves the user's ability to track the values of a single series of values across a multiseries stacked BarChart. This is valid on for multiseries BarCharts or ComboCharts when `GraphType` is `STACK`. The lines are drawn using the drawing attributes specified in the `BarBorder` parameter. If `BarBorder` not defined, one pixel solid black lines are drawn.

OFF specifies that no connectors are drawn.

LINE specifies that lines are drawn between the top and bottom of bars in the same series.

FILL specifies that lines are drawn between the top and bottom of bars in the same series and the area between those lines is filled with the bar set color.

**Used in These Charts**

Bar, Combo

## StackLabel

---

```
StackLabel[N] = Type;
```

In the context of stacked bar or line charts, `StackLabel` defines how the default active labels should be generated for each line symbol when `GraphType` is `STACK` or `PERCENT`.

**Used in These Charts**

Bar, Combo, Line, Pareto, Stock

**Example:**

```
StackLabel = TOTAL;
```

**Attributes**

Type

**Type**

---

Type, in context of the `StackLabel` parameter, determines how the numeric value of the active label is shown.

**Example:**

```
StackLabel = TOTAL;
```

### Values

TOTAL	Uses the accumulated numeric total for the label
ITEM	Uses the individual item's numeric value for the label
PERCENT	Uses the accumulated numeric total for the label, displayed as a percentage of the total of all values.

### Default

TOTAL

## StockAnimationStyle

---

```
StockAnimationStyle = GROW | FADE | NONE
```

Specifies how the stock series (high, low, open, close) initially appear in a Stock chart. This parameter is only valid in SVG or SVGWeb output formats.

### Attributes

Style

### Style

---

`style` refers to the manner in which stock data series are first rendered in a stock chart.

### Example:

```
StockAnimationStyle = GROW;
```

### Values

GROW	The stock series data start as a straight line at 0 and each point bends to its actual value.
FADE	The stock series fade in.
NONE	The series are immediately visible.

### Default

NONE

## StockAxis

---

```
StockAxis[N] = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;
```

`StockAxis` defines which side to be used when mapping the X and Y axes, respectively.

### Used in These Charts

Stock

### Example:

```
StockAxis = (BOTTOM, LEFT), (RIGHT, TOP);
```

**Attributes**

XAxis            YAxis

**XAxis**

XAxis, in context of the StockAxis parameter, determines whether the X-axis for a pair of axes will be on the bottom or the top of the chart.

**Example:**

```
StockAxis = (TOP, LEFT), (BOTTM, RIGHT);
```

**Values**

BOTTOM        X values will be plotted along the chart's bottom axis  
TOP            X values will be plotted along the chart's top axis

**Default**

BOTTOM

**YAxis**

YXAxis, in context of the StockAxis parameter, determines whether the Y-axis for a pair of axes will be on the left or the right of the chart.

**Example:**

```
StockAxis = (TOP, LEFT), (BOTTM, RIGHT);
```

**Values**

LEFT           X values will be plotted along the chart's left axis  
RIGHT          X values will be plotted along the chart's right axis

**Default**

LEFT

**StockColorTable**

```
StockColorTable[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

StockColorTable defines a set of colors for dataset N that overrides all other color specifications for that set. The parameters used for specifying the color of data points in a chart are (in ascending order of precedence) ColorTable, StockSets, StockFillPattern and StockColorTable. StockColorTable is used most frequently to color some specific stock data point.

For example

```
StockColorTable2 = ,,blue;
```

will change the third stock data point in the second series to blue, while all other data points in the chart continue to be colored by one of the other color related parameters.

The colors you can use are defined in the common Color attribute (Chapter 4).

*Used in These Charts*

Stock

*Example:*

```
StockColorTable2 = , , red;
```

*Attributes*

None

**StockData[n]**


---

```
StockData[1-50] = (High1, Low1, Open1, Close1), (High21, Low2, Open2, Close2), ...;
```

StockData is used to define stock values.

*Used in These Charts*

Stock

*Example:*

```
StockData = (120,119,120,121), (32,30.25, 29.75, 31.5);
```

*Attributes*

High	Low	Open	Close
------	-----	------	-------

**High**


---

High defines the highest stock value for that data point.

*Example:*

```
StockData = (120,119,120,121), (32,30.25, 29.75, 31.5);
```

*Values*

Any stock number

null	No symbol will be displayed for this data point
------	---

*Default*

none

**Low**


---

Low defines the lowest stock value for that data point.

*Example:*

```
StockData = (120,119,120,121), (32,30.25, 29.75, 31.5);
```

*Values*

Any stock number

null	The value of High will be used in place of Low
------	--



**Default**

none

**Open**

`Open` defines the opening stock value for that data point.

**Example:**

```
StockData = (120,119,120,121), (32,30.25, 29.75, 31.5);
```

**Values**

Any stock number

`null` No tic mark will be drawn for this data point

**Default**

none

**Close**

`Close` defines the closing stock value for that data point.

**Example:**

```
StockData = (120,119,120,121), (32,30.25, 29.75, 31.5);
```

**Values**

Any stock number

`null` No tic mark will be drawn for this data point

**Default**

none

**StockFillPattern**

`StockFillPattern[N]` = (*type, color1, color2, imageURL, ...*);

The `StockFillPattern` parameter provides a visual pattern fill for stock sets in a stock chart.

<b>Type</b>	NONE	no pattern, do default fill, if any
	<b>Built-In Patterns</b>	
	FSLASH	front slash type
	BSLASH	back slash type
	DGRID	diagonal grid lines, (front and back slash lines)
	HORIZONTAL	horizontal lines
	VERTICAL	vertical lines
	GRID	grid lines, (horizontal and vertical lines)
	<b>Gradients</b>	
	GRADIENTVERTICAL	bottom to top style gradient

	GRADIENTHORIZONTAL	left to right style gradient
	GRADIENTFDIAG	top right to bottom left style gradient
	GRADIENBDBIAG	top left to bottom right style gradient
	GRADIENRADIAL	radial style gradient
	GRADIENCENTERHORIZONTAL	center out horizontal style gradient
	GRADIENCENTERVERTICAL	center out vertical style gradient
<b>Images</b>		
	IMAGE	use an image specified in the optional imageURL element
<b>color1</b>	This color is used in the following ways: - <i>Foreground color for patterns</i> - <i>Starting color for gradients</i> - <i>Ignored in images</i>	
<b>color 2</b>	This color is used in the following ways: - <i>Background color for patterns</i> - <i>Stopping color for gradients</i> - <i>Ignored in images</i>	
<b>imageURL</b>	The URL to an image to use as the fill	

**Used in These Charts**

Stock

**Example:**

```
StockFillPattern = (DGRID,blue,white);
```

**Attributes**

Type	Color1	Color2
ImageFormat	ImageURL	

**StockLabels[n]**

```
StockLabels[1-50] = ("Label", "URL", "Target"), ...;
```

StockLabels can be used to override the default active labels generated for each data value and/or to specify a hyper-link to another document, in a given target window or frame.

**Used in These Charts**

Stock

**Example:**

```
StockLabels1 = ("DCX", "http://www.daimlerchrysler.com", "infoframe"),
("MSFT", "http://www.microsoft.com", "infoframe"),
("SEBL", "http:// http://www.siebel.com", "infoframe");
```

**Attributes**

Label	URL	Target
-------	-----	--------

---

## StockSets

---

```
StockSets[N] = (Label1, Color1, Width1, TicLen1), (Label2, Color2, Width2, TicLen2), ...;
```

`StockSets` defines the display attributes for one or more stock data sets.

### *Used in These Charts*

Stock

### *Example:*

```
StockSets = ("DCX", azure, 1, 3),  
            ("MSFT", darkblue, 1, 3),  
            ("SEBL", moccasin, 1,3);
```

### *Attributes*

<i>Label</i>	<i>Color</i>	<i>Width</i>	<i>TicLen</i>
--------------	--------------	--------------	---------------

---

### *Width*

`Width` defines the width of the vertical bar (hi/lo) in pixels. These pixel values override any `StockWidth` attributes for the current stock set.

### *Example:*

```
StockSets = ("DCX", azure, 1, 3),  
            ("MSFT", darkblue, 1, 3),  
            ("SEBL", moccasin, 1,3);
```

### *Values*

Any pixel value

0 `StockWidth` attributes are used to determine the size of the bar

### *Default*

0

---

### *TicLen*

`TicLen` defines length of the open/close tic marks in pixels. These pixel values override any `StockWidth` attributes for the current stock set.

### *Example:*

```
StockSets = ("DCX", azure, 1, 3),  
            ("MSFT", darkblue, 1, 4),  
            ("SEBL", moccasin, 1,5);
```

### *Values*

Any pixel value

0 `StockWidth` attributes are used to determine the width of the tic mark

**Default**

0

**StockWidth**

---

```
StockWidth[N] = (Width, TicLen);
```

`StockWidth` specifies the relative size of every bar and tic mark used when displaying stock values.

**Used in These Charts**

Stock

**Example:**

```
StockWidth = (5, 80);  
StockWidth = (.2, .5);
```

**Attributes**

Width	TicLen
-------	--------

**Width**

---

`Width` controls the width of the bars for stock data points.

**Example:**

```
StockWidth = (5, 80);  
StockWidth = (.2, .5);
```

**Values**

0 to 100 or 0.0 to 1.0	Width of bar in percentage of space allocated for the stock symbol
100 or 1.0	The bar will occupy all of the space allocated for each stock symbol, leaving no room for the tic marks

**Default**

33

**TicLen**

---

`TicLen` controls the width of the tic marks indicating open & close values for stock data points.

**Example:**

```
StockWidth = (5, 80);  
StockWidth = (.2, .5);
```

**Values**

0 to 100 or    Width of bar in percentage of space allocated for the stock symbol  
 0.0 to 1.0  
 0                The tic mark will fill up the remaining space not used by the bar

**Default**

0

**StripLayout**


---

```
StripLayout[N] = (NumSlots, InitialFill, MaxFill, UndefinedString) ;
```

StripLayout defines basic strip chart information, including how many "steps" there are in the chart's X axis; which side of the chart the data should begin fill from; the maximum number of datapoints that can be simultaneously loaded without loss; and the string used when a tic is shown but no data for the tic is available.

**Used in These Charts**

Strip

**Example:**

```
StripLayout = (30, RIGHT, 45, "none");
```

**Attributes**

NumSlots	InitialFill	MaxFill	UndefinedString
----------	-------------	---------	-----------------

**NumSlots**


---

NumSlots controls how many "steps" there are in the chart's X axis.

**Example:**

```
StripLayout = (30, RIGHT, 45, "none");
```

**Values**

Whole number greater than 0

**Default**

None

**InitialFill**


---

InitialFill controls which side of the chart the data begins to fill in from.

**Example:**

```
StripLayout = (21, RIGHT, 20, "*");  

StripLayout = (22, LEFT, 100, "*");
```

**Values**

LEFT  
RIGHT

**Default**

RIGHT

**MaxFill**

---

MaxFill controls the maximum number of datapoints that can be simultaneously loaded without loss.

**Example:**

```
StripLayout = (22, RIGHT, 100, "*");
```

**Values**

Whole number

**Default**

None

**UndefinedString**

---

UndefinedString controls the string used when a tic is shown but no data for the tic is available.

**Example:**

```
StripLayout = (8, RIGHT, 100, "*");
```

**Values**

String value, generally a single symbol.

**Default**

None

**TaskColorTable**

---

```
TaskColorTable[1-50] = Color1, Color2, Color3, Color4, Color5, ...;
```

TaskColorTable defines a set of colors for dataset N that overrides all other color specifications for that set. The parameters used for specifying the color of tasks in a chart are (in ascending order of precedence) ColorTable, DataSets, and TaskColorTable. TaskColorTable is used most frequently to color some specific task.

For example

```
TaskColorTable2 = ,,blue;
```

will change the third task in the second series to blue, while all other tasks in the chart continue to be colored by one of the other color related parameters.

The colors you can use are defined in the common `Color` attribute (Chapter 4).

### *Used in These Charts*

Time

#### **Example:**

```
TaskColorTable2 = , , red;
```

#### **Attributes**

None

## TaskHeight

---

```
TaskHeight = value;
```

The `TaskHeight` parameter specifies the height of a task bar in a `TimeChart`.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

#### **Example:**

```
TopLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie", "Steve",  
"Paul";
```

#### **Attributes**

*Label*

## TicLocations

---

```
TopTicLocations[N] = value, value, value, ...;  
BottomTicLocations[N] = value, value, value, ...;  
LeftTicLocations[N] = value, value, value, ...;  
RightTicLocations[N] = value, value, value, ...;
```

The `TicLocations` parameter can be used with a corresponding `Label` parameter for complete control of tic drawing and tic labeling. The labels specified in the `Labels` parameter are drawn in order at the locations specified in this parameter. For best results, this parameter should be used in conjunction with an explicitly set axis scale.

### *Used in These Charts*

#### **Example:**

```
TopTicLocations = 10, 20, 30, 40;
```

#### **Attributes**

*value*

## Tics

```
TopTics[N] = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment,backgroundColor,rotationPoint);
BottomTics[N] = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment,
backgroundColor,rotationPoint);
LeftTics[N] = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment,backgroundColor,rotationPoint);
RightTics[N] = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment,
backgroundColor,rotationPoint);
```

The `Tics` parameter specifies the label attributes for the tic marks displayed along a given axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the `Tics` parameter. If any attribute is not defined, any previous value of that attribute will be used.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>backgroundColor</code>	Background color of the tic label area
<code>rotationPoint</code>	For rotated axis labels, anchor point for the rotation

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

The legal values for `rotationPoint` are `LEFT`, `RIGHT`.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
TopTics = ("ON", black, "Helvetica", 10, LEFT);
```

### Attributes

<code>Mode</code>	<code>Color</code>	<code>FontName</code>	<code>FontSize</code>	<code>Angle</code>
<code>InteriorAlignment</code>	<code>BackgroundColor</code>	<code>RotationPoint</code>		

### Mode

`Mode` determines whether or not the tic labels are shown on that axis.

### Example:

```
TopTics = ("ON", black, "Helvetica", 10);
```

### Values

<code>ON</code>	Show tic labels for this axis
<code>OFF</code>	Don't show tic labels on this axis

### Default

`ON`



## TicLayout

---

```
TopTicLayout[N] = (Mode, SkipCount, StaggerLevels);
BottomTicLayout[N] = (Mode, SkipCount, StaggerLevels);
LeftTicLayout[N] = (Mode, SkipCount, StaggerLevels);
RightLayout[N] = (Mode, SkipCount, StaggerLevels);
```

The `TicLayout` parameter is normally used in cases where the tic axis labels may overlap if not adjusted. This allows one to adjust the visual data densities for your chart.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
TopTicLayout = (AUTOSTAGGER, 5, 3);
```

### Attributes

Mode	SkipCount	StaggerLevels
------	-----------	---------------

### Mode

---

`Mode` controls the type of layout mode that should be used with the axis labels.

### Example:

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

### Values

NORMAL	No explicit layout processing should occur
AUTO	Insures that labels never overlap. If labels are overlapping, it automatically staggers labels up to the number of levels defined in the <code>StaggerLevels</code> parameter (default = 2). If the labels still overlap, even after staggering, labels will be automatically skipped at a constant interval until none overlap.
AUTOSKIP	The axis labels should be automatically skipped at a constant interval if (and only if) they overlap
AUTOSTAGGER	The axis labels should be automatically staggered up to the number of levels defined in the <code>StaggerLevels</code> parameter, if (and only if) they overlap
SKIP	a certain number of axis tics should be skipped, and uses the <code>SkipCount</code> parameter to determine that number. (Default = 1.)
STAGGER	Axis labels should be staggered, using the number of levels defined in the <code>StaggerLevels</code> parameter. (Default = 2.)
SKIPSTAGGER	Axis labels should be skipped and staggered, using the <code>SkipCount</code> and <code>StaggerLevels</code> parameters. (Defaults = 1 & 2, respectively.)

### Default

NORMAL

## *StaggerLevels*

---

`StaggerLevels` the number of visual "levels" to which axis labels can or should be staggered for good visibility.

### *Example:*

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

### *Values*

0 or 1            No stagger occurs  
>=2             Integer Number of text lines staggered

### *Default*

2

## *SkipCount*

---

`SkipCount` controls the number of tics that should be skipped should the `Mode` be set to `SKIP`.

### *Example:*

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

### *Values*

0                No skipping occurs  
>0              Integer Number of axis tics skipped

### *Default*

1

## **TitleSpacing**

---

```
TitleSpacing = Number;
```

The `TitleSpacing` parameter is specified in pixels and defines the amount of space between an extended title and the border of the chart. The default value is 5. A value of 0 will allow the title background to extend to the border of the chart. The parameter is only relevant when a title has its `extend` attribute set ON.

### *Used in These Charts*

All

## TopActiveLabels

---

```
TopActiveLabels = ("Label", "URL", "Target"), ...;
```

The top axis labels become active labels when `TopActiveLabels` parameter is used. Each set in parenthesis has a corresponding set within a `DataSet` parameter.

### Used in These Charts

All

### Attributes

*Label*            *URL*            *Target*

## ToggleDataVisibility

---

```
ToggleDataVisibility = ON/OFF;
```

When `ToggleDataVisibility = ON`, clicking on a legend item will temporarily hide or show the associated data series. This parameter is only valid in SVG or SVGweb output formats.

### Example:

```
ToggleDataVisibility = OFF;
```

### Attributes

(Switch)

## TopAxis

---

```
TopAxis = (Label, Color, FontName, FontSize, Angle, interiorAlignment);
```

If `TopAxis` is defined for a Combo chart, then the top axis will be used to map the X data values for all line sets, unless otherwise specified using the `LineAxis` parameter. The group sets the typographic characteristics for the data values.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
--------------------------------	---

The legal values for `interiorAlignment` are `LEFT`, `RIGHT`, or `CENTER`.

### Used in These Charts

Combo

### Example:

```
TopAxis = ("", black, "TimesRoman", 16, 0);
```

**Attributes**

*Label Color FontName FontSize Angle interiorAlignment*

**TopAxisTitle**

```
TopAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);
```

The `TopAxisTitle` parameter specifies the label attributes for the axis title, which centered along the top axis, just above the grid. When the `Header` parameter, whether because of the use of a legend, or for some other reason, creates a title that seems visually unbalanced, you may find this parameter produces a more pleasing chart title.

<code>interiorAlignment</code>	Specifies the alignment to use in text strings that contain multiple lines.
<code>exteriorAlignment</code>	Specifies the alignment for the entire Title object.

The legal values for `interiorAlignment` and `exteriorAlignment` are LEFT, RIGHT, or CENTER.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
TopAxisTitle = ("Ceres Prototype Project Schedule\n ", black, "Helvetica", 12);
```

**Attributes**

*Label Color FontName FontSize Angle interiorAlignment exteriorAlignment*

**TopAxisTitleActiveLabel**

```
TopAxisTitleActiveLabel = ("Label", "URL", "Target");
```

`TopAxisTitleActiveLabel` defines a single active label destination for the `TopAxisTitle` parameter.

**Used in These Charts**

All

**Example:**

```
TopAxisTitleActiveLabel = ("Destination", "demo.html", "frame1");
```

**Attributes**

*Label URL Target*

---

## TopAxisTitleBox

---

```
TopAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor,
TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```

The `TopAxisTitleBox` parameter specifies the region attributes for the axis title centered along the axis. The image-related attributes need not be used unless you want to use an image texture on the box.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
TopAxisTitleBox = (lightgray, SHADOW, 3,,,gray);
```

### *Attributes*

```
Color           BorderType   BorderWidth   ImageURL      ImageFormat
BorderColor TRCornerStyle BRCornerStyle BLCornerStyle CornerColor
```

---

## XXCornerStyle

---

The drawing style for each of the four corners of the region. Styles are specified in a clockwise fashion starting in the upper left - Top Left, Top Right, Bottom Right, and Bottom Left. Legal values are SQUARE, SNIP and ROUND. The default is SQUARE.

---

## TopColor

---

```
TopColor = Color;
```

`TopColor` controls the color of the top axis and the tic marks, but not the tic mark labels. The default axis color is black. If the `NULL` color is specified, the axis color is not changed by this parameter.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
TopAxisColor = xB5D5F0;
```

### *Attributes*

```
Color
```

---

## TopDrawMinorTics

---

```
TopDrawMinorTics = ON/OFF;
```

`TopDrawMinorTics` controls whether or not Top tics are drawn. The default value is ON.

**Example:**

```
TopDrawMinorTics = OFF;
```

**Attributes**

(Switch)

**TopFormat**

```
TopFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");
```

`TopFormat` adjusts the numeric labels that are automatically generated for the top axis, should one be defined.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
TopFormat = (INTEGER);
TopFormat = (FLOAT, "$%,9.2f", ,);
```

**Attributes**

<code>FormatType</code>	<code>FormatExpr</code>	<code>TimeBase</code>	<code>TimeUnit</code>
-------------------------	-------------------------	-----------------------	-----------------------

**FormatType**

`FormatType` specifies the type of number being displayed on the top axis.

**Example:**

```
TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
TopFormat = (INTEGER);
TopFormat = (FLOAT, "$%,9.2f", ,);
```

**Values**

DATE	Axis value are shown as date and/or time values. See <b>Appendix A: Date and Time Values</b> for further detail.
FLOAT	Axis values are shown with decimal parts.
INTEGER	Axis values are shown as integers, and are rounded if necessary.

**Default**

INTEGER

**TimeBase**

The `TimeBase` attribute specifies the base date to be used when determining the actual date or time value when using a time unit or numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

**Values**

String values representing dates or times

**Default**

None

**TimeUnit**

---

The `TimeUnit` attribute controls the time multiplier to be used when determining the actual data/time value when using a numeric value. See **Appendix A: Date and Time Values** for further detail.

**Example:**

```
TopFormat = (DATE, "%w\n%M/%D", "1 Jan 2000", "1h");  
TopFormat = (DATE, "%M/%D", "1 Apr 96", "1d");
```

**Values**

String values representing dates or times

**Default**

None

**TopLabels**

---

```
TopLabels = "Label1", "Label2", ...;
```

The `TopLabels` parameter specifies a list of custom tic mark labels that will be used instead of the numeric labels automatically generated by the axis. The `TopLabels` will be evenly placed along the axis, overriding any tic placement specified by the `StepValue` attribute.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
TopLabels = "Laurie", "Amy", "K.C.", "Arnie", "Mike", "Kathy", "Julie", "Steve",  
           "Paul";
```

**Attributes**

`Label`

---

## TopMargins

---

```
TopMargins = (LeftSideMargin, RightSideMargin);
```

The `TopMargins` parameter specifies the gap, in pixels, at the beginning and end of the top axis. Most often used to prevent clipping of data points at the extreme ends of the scale.

**Example:**

```
TopMargins = (20, 20);
```

---

## TopScale

---

```
TopScale = (MinValue, MaxValue, StepValue);
```

The `TopScale` parameter specifies the minimum and maximum data values which will be displayed along the top axis. If the `TopScale` parameter is not defined, or the `MinValue` and `MaxValue` parameters are the same or one of them is not defined, then the tic mark locations will be automatically determined based on the actual data values being displayed. That is, the axis will be "autoscaled" using the current data values to determine "reasonable" values for `MinValue`, `MaxValue` and `StepValue`. If values are supplied for any of `MinValue`, `MaxValue`, or `StepSize`, those values will be used as part of the autoscaling.

*Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Time, X-Y

**Example:**

```
TopScale = TopScale = ("1 Apr 96", "1 Jun 96", "14d");
```

**Attributes**

<code>MinValue</code>	<code>MaxValue</code>	<code>StepValue</code>
-----------------------	-----------------------	------------------------

---

### *MinValue*

---

`MinValue` sets the absolute lower visible limit for the top axis scale.

**Example:**

```
TopScale = ("1 Apr 96", "1 Jun 96", "14d");
```

**Values**

Any real number, date, or time less than `MaxValue`

**Default**

None

---

### *MaxValue*

---



`ScrollMax` sets the absolute upper visible limit for the top axis scale.

**Example:**

```
TopScale = ("1 Apr 96", "1 Jun 96", "14d");
```

**Values**

Any real number, date, or time greater than `MinValue`

**Default**

None

**StepValue**

---

`StepValue` is optional, and may be used to specify a given step between tic marks along the top axis, starting with the `MinValue`. If `StepValue` is not an even multiple of the difference between the `MinValue` and `MaxValue`, then no tic mark will be displayed at the `MaxValue`. If `StepValue` is not defined, tic marks will be placed at "reasonable" locations along the axis, depending on the range of values being displayed.

**Example:**

```
TopScale = ("1 Apr 96", "1 Jun 96", "14d");
```

**Values**

Any real number, date, or time between `MinValue` and `MaxValue`

**Default**

1

## TopScroll

---

```
TopScroll = (ScrollMin, ScrollMax);
```

The `TopScroll` parameter specifies a range of values through which an axis can be scrolled. When the `ScrollMin` and `ScrollMax` attributes are defined for the axis, the axis will be displayed as a slider bar, using the axis color defined, with a white background. The relative size of the slider represents the percentage of the entire range currently being displayed. That is, it graphically depicts the size of the current axis range (`MinValue` and `MaxValue` attributes) relative to the scrollable region (`ScrollMin` and `ScrollMax` attributes). See the `TopScale` parameter for `MinValue` and `MaxValue` definitions.

`TopScroll` should only be used in conjunction with the `TopScale` parameter.

**Example:**

```
TopScroll = (0, 98);
```

**Attributes**

`ScrollMin`                      `ScrollMax`

**ScrollMin**

`ScrollMin` sets the lower visible limit for a scrollbar defined with `TopScroll`

**Example:**

```
TopScroll = (0, 98);
```

**Values**

<MinValue

**Default**

None

**ScrollMax**

ScrollMax sets the upper visible limit for a scrollbar defined with TopScroll

**Example:**

```
TopScroll = (0, 98);
```

**Values**

>MaxValue

**Default**

None

**TopTics**

```
TopTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint);
```

The TopTics parameter specifies the label attributes for the tic marks displayed along the axis. The tic labels are generated automatically based on the other parameter settings, and are displayed using the given label attributes in the TopTics parameter. If any attribute is not defined, any previous value of that attribute will be used.

interiorAlignment	Specifies the alignment to use in text strings that contain multiple lines.
backgroundColor	Background color of the tic label area
rotationPoint	For rotated axis labels, anchor point for the rotation

The legal values for interiorAlignment are LEFT, RIGHT, or CENTER.

The legal values for rotationPoint are LEFT or RIGHT.

**Used in These Charts**

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

**Example:**

```
TopTics = ("ON", black, "Helvetica", 10);
```

**Attributes**

```
Mode      Color      FontName  FontSize  Angle  interiorAlignment
InteriorAlignment  BackgroundColor  RotationPoint
```

---

## Mode

---

Mode determines whether or not the tic labels are shown on that axis.

### Example:

```
TopTics = ("ON", black, "Helvetica", 10);
```

### Values

ON      Show tic labels for this axis  
OFF     Don't show tic labels on this axis

### Default

ON

---

## TopTicLayout

---

```
TopTicLayout = (Mode, SkipCount, StaggerLevels);
```

The TopTicLayout parameter is normally used in cases where the tic axis labels may overlap if not adjusted. This allows one to adjust the visual data densities for your chart.

### Used in These Charts

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### Example:

```
TopTicLayout = (AUTOSTAGGER, 5, 3);
```

### Attributes

Mode                      SkipCount                      StaggerLevels

---

## Mode

---

Mode controls the type of layout mode that should be used with the axis labels.

### Example:

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

**Values**

NORMAL No explicit layout processing should occur

AUTO Insures that labels never overlap. If labels are overlapping, it automatically staggers labels up to the number of levels defined in the `StaggerLevels` parameter (default = 2). If the labels still overlap, even after staggering, labels will be automatically skipped at a constant interval until none overlap.

AUTOSKIP The axis labels should be automatically skipped at a constant interval if (and only if) they overlap

AUTOSTAGGER The axis labels should be automatically staggered up to the number of levels defined in the `StaggerLevels` parameter, if (and only if) they overlap

SKIP a certain number of axis tics should be skipped, and uses the `SkipCount` parameter to determine that number. (Default = 1.)

STAGGER Axis labels should be staggered, using the number of levels defined in the `StaggerLevels` parameter. (Default = 2.)

SKIPSTAGGER Axis labels should be skipped and staggered, using the `SkipCount` and `StaggerLevels` parameters. (Defaults = 1 & 2, respectively.)

**Default**

NORMAL

**StaggerLevels**

`StaggerLevels` the number of visual "levels" to which axis labels can or should be staggered for good visibility.

**Example:**

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

**Values**

0 or 1 No stagger occurs

>=2 Integer Number of text lines staggered

**Default**

2

**SkipCount**

`SkipCount` controls the number of tics that should be skipped should the `Mode` be set to `SKIP`.

**Example:**

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

**Values**

0 No skipping occurs

>0 Integer Number of axis tics skipped

**Default**

1

---

## TopTicLength

---

```
TopTicLength = Number;
```

The `TopTicLength` parameter defines the size of axis tic marks which are displayed along the top axis of a chart. The parameter will reset the automatically generated tic length. The value defines the number of pixels to use for the length of the tic mark. By default, the number of pixels used is the width of the character zero (0) as found in the font applied to the label. Setting the `TopTicLength` to the value -1 will cause the default size to be used.

### *Attributes*

*Number*

### *Number*

---

Apparent length of a top axis tic mark in a chart, in pixels.

### *Used in These Charts*

Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

### *Example:*

```
TopTicLength = 10;
```

### *Values*

0	No effect (zero length tics are not drawn).
1 or greater	Whole number length in pixels

### *Default*

-1

---

## Update

---

```
Update;
```

`Update` causes one "slot" of data to be extracted from the input data queue for each data set and displayed. Update will also update the axes and configure the dwell labels for the slot (if defined).

### *Used in These Charts*

Strip

### *Example:*

```
Update;
```

### *Attributes*

None

---

## UniqueTaskColors

---

```
UniqueTaskColors = ON|OFF;
```

If set to ON, then a unique color will be chosen from the color table for each task in a taskbar. (See the ColorTable parameter.) The default is OFF which uses the previously specified taskbar color.

### *Used in These Charts*

*Time*

### *Example:*

```
UniqueTaskColors = ON;
```

### *Attributes*

mode

---

## WhiskerType

---

```
WhiskerType = Type;
```

WhiskerType controls the width of the whisker on a box. The whisker can be drawn as either a line or a box.

### *Used in These Charts*

Box Chart

### *Example:*

```
WhiskerType = BOX;  
WhiskerType = LINE;
```

### *Attributes*

Type

### *Type*

---

Type refers to the width of the whisker on a box.

### *Values*

BOX	BOX is 40% of the main box width
Line	Line whiskers are 1 pixel

### *Default*

BOX

## TwentyLineSetName

---

```
TwentyLineSetName = name;
```

Name assigned to the 20% line. Used in the legend if the Legend CDL parameter does not define a label for the 20% line.

### *Used in These Charts*

Pareto

### *Example:*

```
TwentyLineSetName = "20% Line"
```

## ViewPoint

---

```
ViewPoint = (CARTESIAN, X, Y, Z);  
or  
ViewPoint = (SPHERICAL, radius, phi, theta);
```

The view point from which a 3D BarChart is rendered. The viewpoint can only be in front, on top, and to the right of the center of the chart.

### *Used in These Charts*

3DBarchart

### *Example:*

```
ViewPoint = (CARTESIAN, 300, 0, 300);  
ViewPoint = (SPHERICAL, 0, 45, 300);
```

### *Attributes*

<i>CoordinateSystem</i>	<i>X</i>	<i>Y</i>	<i>Z</i>	<i>Radius</i>	<i>Phi</i>	<i>Theta</i>
-------------------------	----------	----------	----------	---------------	------------	--------------

## *CoordinateSystem*

---

The system in which the specified numeric values are defined. Legal values are CARTESIAN and SPHERICAL.

If the CoordinateSystem is CARTESIAN, then the numeric values that follow are the X, Y and Z coordinates of the viewpoint for the chart. These coordinates are specified relative to a center point of the chart.

If the CoordinateSystem is SPHERICAL, then the numeric values that follow are the radius, angle of elevation (phi) and the angle of rotation (theta).

## *X,Y,Z*

---

The location of a 3dbarchart viewpoint when the coordinate system is CARTESIAN. These numbers are relative to size of the chart. Only positive values are valid in Cartesian coordinates. Setting any of the coordinates to a negative number is the same as setting that coordinate to 0. To look at the front of the chart, set the view point to (0, 0, a). To look at the chart from a 45 degree angle, set the view point to (a, 0, a). The greater a is, the further the camera is from the chart.

## *Radius*

---

The distance the viewpoint is from the center of chart when the coordinate system is SPHERICAL. Radius should be specified as a positive integer. The vmagnitude of the radius is relative to the width and height of the chart.

## *Phi*

---

The angle of elevation of the viewpoint when coordinate system is SPHERICAL. Phi should be specified as an angle between 0 and 90. 0 is the angle looking directly at the front of the chart. 90 is the angle looking directly down at the top of the chart.

## *Theta*

---

The angle of rotation of the viewpoint when coordinate system is SPHERICAL. Theta should be specified as an angle between 0 and 90. 0 is the angle looking directly at the front of the chart. 90 is the angle looking directly at the end of the chart.

## **ZAxisLabels**

---

`ZAxisLabels = (ON/OFF, Color, FontName, FontSize, Angle, interiorAlignment)`

The drawing style for Z axis labels in a 3D Barchart.

### *Used in These Charts*

3DBarchart

### *Example:*

```
ZAxisLabels = ("ON",black,"Verdana Plain",12,0,null);
```

## **8. Common CDL Attributes**

Many of the CDL attributes are shared across CDL parameters. Rather than repeat descriptions of these attributes ad nauseam in the reference listings, we are listing their details here only once. Throughout the rest of the guide, the attribute list for each CDL element points to these common attributes when the attribute name is in *italic*. When you see an attribute listed in *italic*, it means that you should look to this section for specific details about the attribute. If you recognize a term from the list of attributes-in-



common, but it is not shown in italic, that means that there is some element-specific information about the attribute, so the description is provided with the element.

In the *Examples*, the attribute is shown in **bold**, to help you locate it. In actual code, attributes would not be bolded.

---

## *Angle*

---

`Angle` defines the angle of counterclockwise rotation, in degrees, for an entire label.

### *Example:*

```
LeftTitle = ("Rotated Label", black, "TimesRoman", 16, 90, LEFT);
```

### *Values*

0	degrees counterclockwise from horizontal
90	degrees counterclockwise from horizontal
180	degrees counterclockwise from horizontal
270	degrees counterclockwise from horizontal

### *Default*

0

---

## *BorderColor*

---

`BorderColor` defines the color used to draw a region border when the `BOX` value of `BorderType` is used.

### *Example:*

```
RightTitleBox = (white, BOX, 2, green);
```

### *Values*

Any valid `Color` value may be used. See below for `Color` definitions.

### *Default*

black

---

## *BorderType*

---

`BorderType` defines the edges of a visual region such as a background or label. When the `BOX` and `SHADOW` styles are specified, the border color defaults to black. When the `RAISED` and `RECESS` styles are specified, the border color will be chosen based on the region color.

### *Example:*

```
LeftTitleBox = (yellow, RAISED, 10);
```

**Values**

NONE	No border
BOX	Simple Box outline
SHADOW	Shadow border
RAISED	Raised border
RECESS	Recessed border

**Default**

NONE

**BorderWidth***optional*

`BorderWidth` defines the width of the edges of a visual region such as a background or label.

**Example:**

```
LeftTitleBox = (yellow, RAISED, 10);
```

**Values**

Integer number of pixels wide

**Color**

Colors can be specified in any NetCharts parameter using any of the following values:

- Predefined Color Name
- Hexadecimal RGB value
- NULL Keyword

These colors may be rendered differently on different platforms or within different browsers because of differences in system or browser defined color tables.

**Example:**

```
BarBorder = (SOLID, 2, red);
```

**Values**

Predefined color names are the most convenient method for specifying a color. Over 100 names are supported. Table 1, below is a list of the predefined color names. The default color palette in NetCharts Designer show these predefined colors. You may also use these:

xRRGGBB	RR, GG, BB are the hexadecimal values for red, green, and blue, in the range 00 to ff. The leading 'x' is mandatory and designates the beginning of the hexadecimal value.
NULL	May also be used depending on the specific chart parameter being defined. For example, the <code>DataSets</code> parameter in the <code>NFBarchartApp</code> accepts a NULL parameter, whereas the <code>Background</code> parameter <b>does not</b> accept NULL colors.

Translucency can be added to any color specification by appending a translucency qualifier between 0 and 250. For example `red_0` or `xFF0000_0` is completely translucent, while `red_250` and `xFF0000_250` both represent solid red.

### *Default*

Black for text, gray for backgrounds and chart elements

*Table 1: Predefined Color Names Used in Visual Mining Applets*

antiquewhite	dimgray	magenta	salmon
aquamarine	dodgerblue	maroon	slategray
azure	firebrick	mediumaquamarine	sandybrown
beige	floralwhite	mediumblue	saddlebrown
bisque	forestgreen	mediumpurple	seagreen
black	fuchsia	mediumseagreen	seashell
blue	gainsboro	mediumslateblue	sienna
blueviolet	ghostwhite	mediumvioletred	silver
brown	goldenrod	mediumspringgreen	skyblue
burlywood	gold	mediumturquoise	slateblue
cadetblue	gray	midnightblue	snow
chartreuse	green	mintcream	springgreen
chocolate	greenyellow	moccasin	steelblue
coral	honeydew	navy	tan
cornflowerblue	hotpink	oldlace	tomato
cornsilk	indigo	olive	teal
crimson	indianred	olivedrab	turquoise
cyan	ivory	orange	violet
darkblue	khaki	orangered	wheat
darkcyan	lavenderblush	orchid	whitesmoke
darkgoldenrod	lawngreen	palegoldenrod	white
darkgray	lemonchiffon	palevioletred	yellow
darkgreen	lightblue	palegreen	yellowgreen
darkkhaki	lightcoral	paleturquoise	
darkmagenta	lightgoldenrodyellow	papayawhip	
darkorchid	lightgrey	peachpuff	
darkorange	lightgreen	peru	
darkred	lightpink	pink	
darksalmon	lightsalmon	plum	
darkseagreen	lightseagreen	powderblue	
darkslateblue	lightskyblue	purple	
darkslategray	lightslategray	red	
darkturquoise	lightsteelblue	rosybrown	
darkviolet	lightyellow	royalblue	
darkolivegreen	lime		
deeppink	limegreen		
deepskyblue	linen		

### *FontName*

`FontName` describes the font style to be used for the label. The exact list of font styles is platform dependent, but the font styles below are guaranteed to be available for any Java implementation.

Font names can be augmented with additional font style information. Adding "Plain", "Bold" or "Italic" to the font name modifies the style as specified. For example "Courier Bold Italic" is a valid font name

specification. By default NetCharts chooses a BOLD style for the specified font so "TimesRoman" is equivalent to "TimesRoman Bold". To get a standard version of a font, add "Plain" to the name.

Font names can also be augmented with "underline", "overline", "linethrough", "ascent=N", "descent=N", "leading=N" and "maxLineAdvance=N".

Any combination of style modifiers is allowed. For example "TimesRoman Plain Italic underline overline linethrough ascent=10 descent=0 leading=-15" is a valid font specification which uses a non-bold italic type, draws a line above, below and through the text, and controls the amount of space allocated for character ascents, descents and the space between consecutive lines.

maxLineAdvance specifies the maximum length (in pixels) NetCharts will allow a string using this font specification to be before it wraps to a new line.

### **Example:**

```
RightTitle = ("Large\nLabel", blue, "Courier Plain underline leading=5", 12);
```

### **Values**

<b>Java font</b>	<b>X-11 font</b>	<b>Windows Font</b>
default	misc-fixed	Arial
Helvetica	adobe-helvetica	Arial
TimesRoman	adobe-times	Times New Roman
Courier	adobe-courier	Courier New
Dialog	b&h-lucida	MS Sans Serif

### **Default**

TimesRoman

### **FontSize**

---

FontSize defines the point size of the font.

### **Example:**

```
RightTitle = ("Large\nLabel", blue, "Courier", 12);
```

### **Values**

The available values depend on the font style defined and the platform.

### **Default**

12            12pt font size

### **FormatExpr**

---

The `FormatExpr` attribute specifies a numeric display format to be used for axis or pie slice labels and the active labels generated by default. The format allows a developer to specify numeric and date/time formats using expressions similar to those provided in the C/Unix `printf` function. The format expressions consist of an arbitrary string, plus one or more occurrences of format components, which are preceded by '%'.

**Example:**

```
LeftFormat   = (FLOAT, "$%,9.2f", , );
SliceFormat  = (FLOAT, "%5.2F");
```

<b>FORMAT</b>	<b>RESULT</b>
%g	1234.456
%f	1234.46
%,f	1,234.46
%d	1234
%,d	1,234
%8.1f	1234.5
%08.1f	001234.5
Cost=\$%,.1fM	Cost=\$1,234.5M

**Values**

%f	Display numeric value with fixed number of decimal digits, which can be zero.
%g	Display numeric value using platform specific default format. Generally, this results in a "reasonable" format, unless a large number of decimal digits exist or the number is very large or very small.
%d	Display numeric value as an integer, rounding if necessary.
	The following modifiers can be applied to the format component to yield different outputs:  <div style="text-align: center;">-,0ww.DD</div> where all parts are optional and are interpreted as follows:
-	Left justify the result, otherwise right justify it.
'	Separate the whole number into groups using the group symbol. By default, a comma is used as the group symbol and the group size is set to 3. This option is ignored if zero padding is also specified.
0	Pad with leading zeros, if necessary, to fill entire field.
ww	Total width of the formatted field. If the field is naturally wider than this value, it is ignored.
.	Separate the whole number from the decimal number using the decimal symbol. By default, a period is used as the decimal symbol.
DD	Number of decimal digits to display (for %f format only). If not specified, the default number is 2.
<b>Default</b> %.2f	

**ImageFormat**

ImageFormat specifies how the image will be displayed in the region.

**Example:**

```
HeaderBox = (lightblue, RAISED, 5, "$IMAGES/nc220x90.gif", CENTER);
```

**Values**

TILE	Tiles or clips the image if not the same size as the region.
SIZE	Scales the image to the size of the region.
CENTER	Image is centered in the region

**Default**

TILE

**ImageURL**

---

This attribute can be used to specify an image file to be used to fill the region. Any valid URL may be specified. If a relative URL is given, it will be interpreted based on the `DocumentBase` of the HTML document. See *Known Problems* for details concerning the limits on the use of images within specific browsers, in some situations.

Depending on your browser environment, files in the `$DOCBASE` directory hierarchy may not be accessible because of security restrictions. For that reason, data and image files used by all NetCharts examples are located within the `$CODEBASE` directory hierarchy.

**Example:**

```
RightTitleBox = (lightgreen, SHADOW, 4, "$SYMBOLS/paste.gif", CENTER);
```

**Values**

Relative addressing is supported, or you may use the following keywords for the sake of convenience:

<code>\$DOCBASE</code>	document base
<code>\$CODEBASE</code>	code base
<code>\$NETCHARTS</code>	<code>\$CODEBASE/netcharts</code>
<code>\$ICONS</code>	<code>\$NETCHARTS/icons</code>
<code>\$IMAGES</code>	<code>\$NETCHARTS/images</code>
<code>\$SYMBOLS</code>	<code>\$NETCHARTS/symbols</code>
<code>\$PATTERNS</code>	<code>\$NETCHARTS/patterns</code>

These variables are only interpreted if they are used as the first value in a GIF URL. For example, the following URL will cause the chart to look in the "images" sub-directory in the `NETCHARTS` classpath for the given file:

```
"$IMAGES/cut.gif"
```

**Default**

None

**Label**

---

Any text string can be specified for a `Label`, and may include newline (`\n`) characters for multi-line labels. If the `Label` attribute is not defined or is defined to be "" (the empty string), then a default label will be generated for that data value, depending on the specific chart type. Usually, the default label displays one or more of the numeric values associated with the data point.

In the case of dwell labels, if the first attribute is set to `OUTLINE` then instead of displaying a dwell label, an outline will be drawn around the associated item when the mouse hovers over that item.

**Example:**

```
FooterActiveLabel = ("Days", "cf1.html", "frame1");
FooterActiveLabel = ("OUTLINE", "cfoot.html", "body");
```

**Values**

Any text string	may include \n
""	default label, usually the numeral value of the data
ON	display a popup label over the target
OUTLINE	display an outline of the target
OFF	don't display anything

**Default**

None

**LineStyle**

`LineStyle` tells how to draw lines within a parameter belonging to a chart that features lines, such as a combo chart, line chart, stock chart, or strip chart.

**Example:**

```
LineStyle = (DASHED, 3, red), (SOLID, 2, blue);
```

**Values**

SOLID	solid lines (default), such as this: _____
DOTTED	dotted lines, such as this: .....
DASHED	dashed lines, such as this: -----
DOTDASH	combined dot and dash lines, something like this: _._._.

**Default**

SOLID

**LineWidth**

The `LineWidth` attribute allows the specification of the line thickness in pixels, with 1 as the default value. `LineWidth` is specified with a `GridLine` parameter, and refers to a `Grid` parameter.

**Example:**

```
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

**Values**

Integer number of pixels of line width

**Default**

1

---

## Name

---

The `Name` attribute is a string, in quotes, identifying the particular chart element or data set. `Name` may not actually be seen in a chart except when a legend is used, in which instance it is shown associated with a symbol.

### Example:

```
AddDataPoint = ("R1", 320, 199, 3.85);
<PARAM NAME=Charts VALUE='
    ("Piechart1",PIECHART),
    ("Piechart2",PIECHART),
    ("Piechart3",PIECHART);'>
NoteSets = ("note1"), ("note2");
NoteSets = ("Axes", BOTTOM), ("TextOnly", CENTER), ("Text2Only", TOP);
LineSets = ("Sprocket", black),
    ("Actuator", black),
    ("Do-Hicky", black),
    ("Thingy", black);
```

### Values

Any legal string value

### Default

None

---

## Target

---

`Target` is used in the same way that it is used in HTML. If the destination document is to be loaded into a window or frame other than the current window or frame, you can specify where the destination document should load by assigning a window or frame name to the `Target` attribute. Target frame names must be assigned to frames and windows as identifiers.

If `Target` is not defined, the current frame or window will be updated.

If the value, `LOADPARAMS`, is used as the target, then the `URL` will be assumed to point to a `NetCharts` parameter file, which contains *only* `NetCharts` parameter definitions. In that case, the `NetCharts` parameter file will be processed and all of parameters will be used to update the current chart. This allows users to modify the chart slightly or completely change all of the data, without creating a new chart.

### Example:

```
BarActiveLabels = ("Hardware Resources",
    http://www.visualmining.com/products/netcharts.html,
    _blank);
```

### Values

Standard HTTP target designation. Case-sensitive identifier when the frame or window name has been assigned via the target element's `NAME` attribute. Can use the four reserved HTML target names:

<code>_blank</code>	Browser creates a new window for the destination document.
<code>_parent</code>	Destination document replaces the current frame's framesetting document, if one exists; otherwise, defaults to <code>_self</code> .



---

<code>_self</code>	Destination document replaces the current document in its window or frame.
<code>_top</code>	Destination document is to occupy the entire browser window, replacing any and all framesets that may be loaded.
<code>LOADPARAMS</code>	Points to a NetCharts parameter file, which is specified by the <code>URL</code> attribute.

### ***Default***

The current frame or window will be updated.

## ***URL***

---

URLs are used to indicate the location of a file on the Web or your Intranet.

### ***Example:***

```
BarActiveLabels = ("Hardware Resources",  
                  http://www.visualmining.com/products/netcharts.html);  
  
ActiveLabels1   = ("", "Barchart9Mon.html", "InfoFrame"),  
                  ("", "Barchart9Tue.html"),  
                  ("", "Barchart9Wed.html"),  
                  ("OUTLINE", "Barchart9Thu.html"),  
                  ("OUTLINE", "Barchart9Fri.html");
```

### ***Values***

Standard HTTP URL designation, such as you would use to specify a web page file. If you use full addressing, the normal thing happens. If you use relative addressing, the address is relative to the document containing the applet, not the Codebase of the applet.

### ***Default***

None

## ***Width***

---

Specifies width of some chart element, in pixels.

### ***Example:***

```
SliceBorder = (SOLID, 2, red);
```

### ***Values***

Values are in whole numbers of pixels, unless otherwise specified.

### ***Default***

None

## ***XAxis***

---

Specifies which side of the chart becomes the X axis.

**Example:**

```
GridAxis = (BOTTOM, LEFT), (TOP, RIGHT);
```

**Values**

TOP	top of chart
BOTTOM	bottom of chart

**Default**

BOTTOM

**YAxis**

---

Specifies which side of the chart becomes the Y axis.

**Example:**

```
GridAxis = (BOTTOM, LEFT), (TOP, RIGHT);
```

**Values**

LEFT	left of chart
RIGHT	right of chart

**Default**

LEFT

## Appendix A: Date and Time Values

All NetCharts applications provide support for Date/Time values by allowing chart developers to input Date/Time values as data elements, and to configure axes tics and rollover labels with a variety of formatting options.

### *Date/Time Data Input*

#### Mapping Date/Time Information

Since Date/Time information is non-numeric, each Date/Time input datum must be transformed into a numeric value in order to properly process the information. The means by which this conversion occurs is based on the type of input date information (absolute date value vs. relative or numeric), whether or not the axis scale is defined or autoscaled, and whether or not a TimeBase has been defined. The following are some general guidelines on how the Date/Time processing occurs:

- Relative or numeric time values are converted into absolute time values by computing the offset from the TimeBase.
- The default TimeBase is the current date. The default TimeUnit is 1 day.
- Absolute time values are converted into numeric values by computing the difference from the TimeBase in terms of the TimeUnit. This applies to both axis scale values as well as data values. For example, if the value “15 Dec 2001” is entered as data, and the TimeBase is the 20<sup>th</sup> of December, 2001, and the TimeUnit is one day, the date item will map to  $-5$  (days) and be placed along the axis according to that value.

Parameters that can be used to input Date/Time data are:

```

TopScale      = (MinValue, MaxValue, StepValue);
BottomScale   = (MinValue, MaxValue, StepValue);
LeftScale     = (MinValue, MaxValue, StepValue);
RightScale    = (MinValue, MaxValue, StepValue);
TopScroll     = (ScrollMin, ScrollMax);
BottomScroll  = (ScrollMin, ScrollMax);
LeftScroll    = (ScrollMin, ScrollMax);
RightScroll   = (ScrollMin, ScrollMax);
DataSet[1-50] = a, b, c, ...;
LineSet[1-50] = y1, y2, y3, ...;
LineSet[1-50] = (x1, y1), (x2, y2), (x3, y3), ...;

```

All other DataSet / LineSet / StockSet / BubbleSet parameters

Date/Time data can be input as absolute dates, relative time units, or numerical time units.

#### Absolute Date Expressions

An absolute date expression is any quoted string representing a date or time in some standard convention.

Java automatically recognizes many date/time formats, including these:

**Date Formats**

- 96/04/10
- 4/10/96
- Apr 10, 1996
- 10 Apr 96

**Time Formats**

- 12:30:00
- 12:30:00 PM
- 12:40 GMT

Use absolute date expressions for specifying the minimum or maximum values in an axis range, or when specifying an exact time-based data value.

**Example:**

```
"1 Jan 1996 12:30"
"Jan 1, 1996"
"12:30"
"Wed, 10 Apr 96"
```

This sample of chart code shows how absolute dates can be used, in context.

**Example:**

```
BottomFormat = (DATE, "%M/%D");
BottomScale = ("10 Apr 96", "17 Apr 96");
DataSet1 = ("10 Apr 96", 27),
           ("12 Apr 96 05:30", 50),
           ("4/13/1996", 75),
           ("4/14/96 12:30", 37),
           ("April 15, 96", 87),
           ("Apr 16, 96 12:00", 64);
```

**Relative Time Units**

You may want to specify a date or time value that is relative to another such value. Relative time units are relative to the TimeBase (described below). If the TimeBase is not defined, relative time units are relative to the current Date/Time. The relative time unit components described below may be used to input data, to set the TimeUnit attribute of the axis Format parameter, or to set the StepValue of an axis Scale parameter

**Example:**

A measurement might be taken 20 minutes after the start of an experiment, or a task might end 5 days and 6 hours after its start. In such cases, one uses relative time units:

```
Relative Time Unit = "1Y 2M 3d 12h 30m";
```

Components are described thus:

Time Component	Description
Y	Years
M	Months
d	Days
h	Hours
m	Minutes
s	Seconds

**Example:**

"7d"                    A date one week after the base date  
 "3d 12h 30m"        A date 3 days, 12 hours, and 30 minutes after the base date and time.

The absolute date and time represented by a given relative time unit is calculated thus:

$$\text{Absolute DateTime} = \text{Axis TimeBase} + \text{Relative Time Unit}$$

Where *TimeBase* is defined in the `Format` parameter for the corresponding axis.

**Numeric Time Units**

Date and time values may also be specified as a numeric value, which is interpreted as follows:

$$\text{Absolute DateTime} = \text{Axis TimeBase} + N * \text{Axis TimeUnit}$$

Where *TimeBase* and *TimeUnit* are defined in the `Format` parameter for the corresponding axis. This allows data values to be specified as multiples of a given time unit.

The following parameters would generate the same chart as in the code example for absolute date and time, except that they use a combination of absolute dates, relative time units and numeric time units.

**Example:**

```
BottomFormat = (DATE, "%M/%D", "10 Apr 96", "1d");
BottomScale = (0, "Apr 17, 1996");
TopFormat    = (DATE, "%w", "10 Apr 96", "1d"); <!-- displays day of the
              week -->
TopScale     = (0, "Apr 17, 1996");
DataSet1    = ("10 Apr 96", 27),
              ("2d 5h 30m", 50),
              (3.0, 75),
              ("4d 12h 30m", 37),
              (5, 87),
              (6.5, 64);
```

**Date/Time Data Format and TimeBase**

Date/Time data labels are displayed using axis formatting parameters that convert numerical scale values into a date or time presentation. Additionally, the axis formatting parameters contain the attributes for setting the `TimeBase` and `TimeUnit` used in computing relative and numerical time units. The following parameters are used to configure Date Format and TimeBase information:

```
TopFormat     = (FormatType, "FormatExpression", "TimeBase", "TimeUnit");
BottomFormat  = (FormatType, "FormatExpression", "TimeBase", "TimeUnit");
LeftFormat    = (FormatType, "FormatExpression", "TimeBase", "TimeUnit");
RightFormat   = (FormatType, "FormatExpression", "TimeBase", "TimeUnit");
```

Regardless of how one enters the date or time as data, the display of the values, either as a tic mark labels or as active labels, will be rendered using the `FormatExpression` attribute specified in the `TopFormat`, `BottomFormat`, `LeftFormat` or `RightFormat` parameters.

For axes where data formatting is desired, the FormatType attribute is either DATE or SIMPLEDATE. SIMPLEDATE allows developers to use the same format expressions as described in the Java SimpleDateFormat class within the FormatExpression attribute.

### SIMPLEDATE Format Expression Attributes

The following attributes are supported for the SIMPLEDATE format type:

Field	Attribute	Sample
Year	YYYY	1999
Year	YY	99
Month	MMM	January
Month	MM	01
Month	M	1
Day of week	EEEE	Monday
Day of week	EE	Mon
Day of month	dd	01
Day of month	d	1
Hour (1-12)	hh	01
Hour (1-12)	h	1
Hour (0-23)	HH	00
Hour (0-23)	H	0
Hour (0-11)	KK	00
Hour (0-11)	K	0
Hour (1-24)	kk	01
Hour (1-24)	k	1
Minute	mm	00
Second	ss	00
Millisecond	SSS	001
AM/PM	a	AM
Time zone	zzzz	EST
Time zone	zz	ET
Day of week in month	F	3 <sup>rd</sup> Thursday
Day in year	DDD	001
Day in year	D	1
Week in year	ww	1
Era	G	AD

### DATE Format Expression Attributes

The following attributes are supported for the DATE format type:

Field	Attribute	Sample
Locale form	%L	Mon Jan 01 00:00:00 2000
GMT form	%G	1 Jan 2001 00:00:00 GMT
Year	%Y	1999
Year	%y	99

Month	%M	01
Month	%N	January
Month	%n	Jan
Weekday	%W	Monday
Weekday	%w	Mon
Day of month	%d	01
Hour	%h	00
Minute	%m	00
Second	%s	00

### TimeBase

The TimeBase attribute specifies the base date to be used when determining the actual date/time value when a time unit or numeric value is used. It effectively relates the 0 numeric axis value to the specified TimeBase date/time. By default, the TimeBase is set to the current Date/Time. Consider the following example. If the BottomScale is set to:

```
BottomScale = (-50, 50, 10);
```

And, if the BottomFormat is set to:

```
BottomFormat = (DATE, "%d/%n/%y", "1 Jan 2001", "1d");
```

Then the tic labels would appear as:

```
12/Nov/00  -50 days from 1 Jan 2001
22/Nov/00  -40 days from 1 Jan 2001
02/Dec/00  -30 days from 1 Jan 2001
12/Dec/00  -20 days from 1 Jan 2001
22/Dec/00  -10 days from 1 Jan 2001
01/Jan/01   0 days from 1 Jan 2001
11/Jan/01  +10 days from 1 Jan 2001
21/Jan/01  +20 days from 1 Jan 2001
31/Jan/01  +30 days from 1 Jan 2001
10/Feb/01  +40 days from 1 Jan 2001
20/Feb/01  +50 days from 1 Jan 2001
```

And similarly, if the BottomFormat is then changed to

```
BottomFormat = (DATE, "%d/%n/%y", "15 Jul 2001", "1d");
```

Then the tic labels would be changed to:

```
26/May/01  -50 days from 15 Jul 2001
05/Jun/01  -40 days from 15 Jul 2001
15/Jul/01  -30 days from 15 Jul 2001
25/Jul/01  -20 days from 15 Jul 2001
05/Aug/01  -10 days from 15 Jul 2001
15/Aug/01   0 days from 15 Jul 2001
25/Aug/01  +10 days from 15 Jul 2001
04/Sep/01  +20 days from 15 Jul 2001
14/Sep/01  +30 days from 15 Jul 2001
24/Sep/01  +40 days from 15 Jul 2001
```

03/Sep/01 +50 days from 15 Jul 2001

Please note, if you have input absolute time values as DataSet (or other Set) data, and you have explicitly set up the Scale parameter with absolute time values, then you should not use the TimeBase or TimeUnit axis Format attributes.

## TimeUnit

The TimeUnit axis Format attribute helps to compute numeric values into date or time values. If the value is “1d” (as shown in the above example) then each discrete value in the axis is equivalent to one day. The available TimeUnit symbols are the same as those described in the Relative Time Unit section, described above.

## Index

- \$**
- \$CODEBASE, 318
- \$DOCBASE, 318
- \$ICONS, 318
- \$IMAGES, 318
- \$NETCHARTS, 318
- \$PATTERNS, 318
- \$SYMBOLS, 318
  
- %**
- %f, 317
- %g, 317
  
- <**
- <PARAM> tag, 10
  - usage of, 9
  
- A**
- active label
  - ActiveClicks, 54
  - dwell box, 137
  - dwell label, 136, 137, 138, 190, 191
- active labels, 44
- ActiveClicks, 54
- ActiveLabels, 55
- ActiveLabels[n], 55
- AddDataPoint, 56
- Angle attribute, 313
- annotations
  - defined, 51
  - parameters specific to, 52
- AntiAlias, 58
- AppendDataSet[n], 57, 247
- applet, 9
- applets
  - defined, 9
- arrow style, 140
- arrows, 139, 170, 172, 173, 219
  
- ascent, 50
- attribute
  - defined, 5
- AutoscalePad, 58
- AxesGaps, 59
- AxesLayout, 59
- AxesLayoutDirection, 59
- AxesSizes, 60
- axis
  - color parameters, 46
  - labels, 46, 50
  - scale, 46
  - scrolling, 46
  - title box, 53
  - titles, 46
- Axis
  - ScaleSet, 262
  - Tic Layout, 297, 298
  - Tics, 296
- axis active labels, 44, 46
- axis labels
  - location in chart, 15
- axis modifications
  - parameters associated with, 45
- axis parameters, 45
- Axis Scale, 260
- axis tics
  - location in chart, 15
- AxisThickness, 61
- AxisTitleActiveLabel, 62, 90, 93
- AxisZoom, 62
  - BottomZoom, 62
  - LeftZoom, 62
  - RightZoom, 62
  - RubberbandBorderStyle, 259
  - RubberbandFill, 259
  - specifying the color of the rubberbanded box, 259
  - TopZoom, 62
  
- B**
- background, 53
  - images in, 63, 64, 71, 91, 96, 124, 200, 267, 274, 289



- parameter definition, 63, 64, 71, 91, 96, 124, 200, 267, 274, 289
  - Background, 63, 64, 71, 267, 274, 289
  - bar chart**, 19, 21, 22, 23, 25, 26, 28, 29, 30, 32, 33, 35, 36, 37, 39, 40, 42
    - active labels for stacked charts, 285
    - appearance of bars, 66, 78, 81, 108, 111, 112, 142, 161, 162, 209, 210, 271, 311
    - axes for data sets, 113
    - data values, 117, 118
    - depth of bars in, 65
    - orientation of bars in, 90, 145, 149, 310
    - parameters specific to a, 29
    - specifying a fill pattern for 3D bars, 73, 79
    - specifying a fill pattern in 3D grouped stacked bar, 74, 80
    - specifying active labels on bars, 65
    - specifying bar animation, 66
    - specifying bar corners, 68
    - specifying bar drop shadows, 70, 166
    - specifying bar highlights, 72
    - specifying spotlight overlays over bars, 75, 207
    - stacking in, 150, 240
  - Bar3DDepth, 65
  - BarActiveLabels, 65
  - BarAnimationStyle, 66
  - BarBorder, 66
  - BarCorners, 68
  - BarDropShadow, 70
  - BarHighlights, 72
  - BarRightFillPattern, 73
  - BarRightFillPattern[n]p[n], 74
  - BarSpotlights, 75
  - BarStyle, 78
  - BarTopFillPattern, 79
  - BarTopFillPattern[n]p[m], 80
  - BarValueLabel, 78, 81
  - BarValueLabelBox, 81
  - BarValueLabelStyle, 81
  - BarWidth, 82
  - BestFit, 82
  - BorderColor attribute, 313
  - borders, 313
  - BorderType attribute, 313
  - BorderWidth attribute, 314
  - bottom axis
    - active labels on, 82
    - color, 84, 108
    - custom tic mark labels, 84
    - formatting numbers on, 84
    - optional labels, 89
    - optional title background box, 89
    - scale, 85, 129, 261, 304
    - scroll bars, 87
    - tic length, 88
    - tic mark layout, 88
    - title active labels, 83
    - title box, 83
  - BottomActiveLabels, 82
  - BottomAxisTitleActiveLabel, 83
  - BottomAxisTitleBox, 83
  - BottomColor, 83, 107, 179
  - BottomDrawMinorTics, 84
  - BottomFormat, 84
  - BottomLabels, 84
  - BottomMargins, 85
  - BottomScale, 85
  - BottomScroll, 86
  - BottomTicLength, 87
  - BottomTics, 88
  - BottomTitle, 89
  - BottomTitleBox, 89
  - box chart
    - axes for data sets, 113
    - box height specification, 92, 94
    - data types, 119
    - data values, 117, 118
    - median, 213, 214
    - outliers, 227
    - parameters specific to a, 21
  - Boxchart
    - ShowDataPoints, 272
  - boxes, 52
  - BoxFence, 90
  - BoxFillPattern, 91
  - BoxHeight, 92
  - BoxLimitLines, 93
  - BoxLimitLineStyle, 93
  - BoxSymbolWidth, 94
  - BoxWidth, 94
  - bubble chart
    - adding data points dynamically, 56
    - axes for data sets specified, 95
    - bubble set scales, 97
    - bubble set vectors, 99
    - data sets definition, 99
    - line style, 109, 142, 204
    - parameters specific to a, 23
    - specifying bubble animation style, 95
    - symbols in, 100
  - BubbleAnimationStyle, 95
  - BubbleAxis, 95
  - BubbleFillPattern, 96
  - BubbleScale, 97
  - BubbleSet[n], 99
  - BubbleSets, 99
  - BubbleSymbol, 100
- ## C
- CellTextAutoColorThreshold, 102
  - CellTextAutoColorThresold, 102
  - CenterRadius, 102
  - chart styles, **19**
  - ChartElementSpacing, 103
  - ChartHeight, 105
  - ChartName, 105
  - Charts, 103
  - ChartScript[n], 106
  - ChartType, 106
  - ChartURL[n], 107
  - ChartWidth, 107
  - color, 151
    - hexidecimal format, 314

- making color palettes, 68, 96, 108, 199, 287, 294
- NULL color, 314
- pre-defined names, table of, 315
- Color attribute, 314
- ColorTable, 67, 68, 96, 108, 198
- combo chart
  - 3-D lines, 197
  - active labels for stacked charts, 285
  - active labels on lines, 201
  - axes for data sets, 113
  - data values, 117, 118
  - defining line sets, 203
  - left axis defined in, 177
  - line data sets, 202
  - line set axes, 198
  - line style, 109, 142, 204
  - line width, 210
  - orientation of bars in, 90, 145, 149, 310
  - parameters specific to a, 24, 33, 36, 38
  - right axis definitions, 250
  - specifying a fill pattern for 3D bars, 73, 79
  - specifying a fill pattern in 3D grouped stacked bar, 74, 80
  - specifying active labels on bars, 65
  - specifying bar animation, 66
  - specifying bar corners, 69
  - specifying bar drop shadows, 70, 166
  - specifying bar highlights, 73
  - specifying spotlight overlays over bars, 75, 207
  - stacking in, 150, 240
  - symbols, 110, 205
  - top axis appearance, 299
- Combo chart
  - specifying a line drop shadow, 199
- CumulativeLineStyleName, 108
- CumulativeLineStyle, 109
- CumulativeLineSymbol, 110
- CumulativeLineValueLabel, 111
- CumulativeLineValueLabelBox, 112
- CumulativeLineValueLabelStyle, 112

## D

- dashes in formatted expressions, 317
- data sets
  - identifying, 320
- DataAxis, 113
- DataPointActiveLabels, 115
- DataPointColor, 116
- DataPointJitter, 116
- DataPointSymbol, 116
- DataSet[n], 117, 118
- DataSets, 118
- DataType, 119
- dates, 148, 149, 180, 181, 253, 254, 302, 303, 323
  - absolute, 323
- DebugClear, 120
- DebugSe, 120
- descent, 50
- diagram chart
  - lines and arrows, 139
  - node box, 215

- node definition, 217
- node drag switch, 216, 221
- node label characteristics, 217
- parameters specific to a, 26
- dial chart
  - active labels for sectors, 263
  - assigning dials, 127
  - assigning hands to dials, 167
  - coloring dials, 125, 126
  - deleting sectors, 265
  - dial borders, 122, 123
  - dial scales, 129
  - dividing dials into sectors, 270
  - dragging hands, 165
  - dragging sectors, 266
  - hand active labels, 161
  - hand appearance, 170, 172, 173
  - hands, defining, 163
  - hands, deleting, 164
  - parameters specific to a, 26
  - removing dials, 123
  - sector appearance, 264
  - sector divisions, 264
  - sector labels, 268
  - specifying dial hand animation style, 127
  - specifying dial sector animation, 131
  - tic labels, 132, 133, 169
  - tic marks, 134
- DialActiveLabels, 122
- DialBorders, 122
- DialClip, 123
- DialClipPad, 123
- DialDelete, 123
- DialFillPattern, 124
- DialFills, 125
- DialFormats, 126
- DialHandAnimationStyle, 126
- Dials, 127
- DialScale, 129
- DialSectorAnimationStyle, 131
- DialSize, 131
- DialSquare, 132
- DialTicLabels, 132
- DialTicLabelStyles, 133
- DialTics, 134
- DrawFences, 57, 101, 135
- DrawOrder, 135
- drill-down
  - parameters associated with, 44
- dwel label
  - box, 53
  - box background, 137
- dwel labels, 44
- DwellAnimationHighlightBorderStyle, 135
- DwellAnimationHighlightFill, 136
- DwellAnimationStyle, 137
- DwellBox, 137
- DwellLabel, 55, 138
- DwellOffset, 138

**E**

Edges, 139  
EightyLineSetName, 142  
EightyTwentyLineStyle, 142  
EightyTwentyLineStyleSymbol, 143

**F**

FenceActiveLabels, 145  
FencePosition, 145  
font  
    sizes, 316  
FontEncoding, 146  
FontName attribute, 315  
FontSize attribute, 316  
footer, 146  
    box, 53  
    box background, 147  
Footer, 146  
FooterActiveLabel, 146  
FooterBox, 147  
FormatExpr attribute, 316  
frame, 54  
frames  
    targets in, 320

**G**

GraphLayout, 149  
GraphType, 150  
grid, 151  
    3-D depth, 152  
    axes for, 153  
    color, 151  
    line color, 114, 151  
    line style, 114, 160  
Grid, 114, 151  
Grid3DDepth, 152  
GridAnimationStyle, 152  
GridAxis, 153  
GridBlockActiveLabels, 153  
GridBlockActiveLabels, 153  
GridBlockBackgroundColor, 154  
GridBlockBackgroundColor, 154  
GridBlockCellColorType, 154  
GridBlockCellColorType, 154  
GridBlockColors, 154  
GridBlockColorSpectrum, 155  
GridBlockColorSpectrum, 155  
GridBlockExpressions, 155  
GridBlockLabel, 156  
GridBlockLabels, 156  
GridBlockLayout, 157  
GridBlockLine, 157  
GridBlockSort, 158  
GridBlockValueFormat, 159  
GridBlockValueFormat, 159  
GridBlockValues, 158  
GridBlockValueStyle, 159  
GridLine, 114, 160  
grids

3-D depth, 49  
axes, 49  
defined, 48  
lines in, 49  
parameters specific to, 49  
GroupStackLabels, 161  
GroupStackSegmentLabels, 161

**H**

HandActiveLabels, 161  
HandBorders, 161  
HandButtonBorder, 162  
HandButtonEdgeHighlights, 167  
HandData, 163  
HandDelete, 164  
HandDrag, 165  
HandDropShadow, 166  
HandLabels, 169  
Hands, 167  
HandStyles, 170  
header  
    box, 53  
    box background, 171  
    defining a, 170  
Header, 170  
HeaderActiveLabel, 63, 171  
HeaderBox, 171  
heat map  
    parameters specific to a, 28  
hexidecimal colors, 314  
HTML documents  
    replacing using active labels, 54  
HTML pages  
    targets in, 320

**I**

identifying chart elements, 320  
ImageFormat attribute, 317  
images, 63, 72, 75, 79, 151  
    formatting in a region, 317  
    URL, 318  
ImageURL attribute, 318

**J**

Java font names, 316

**L**

label  
    angles allowed, 313  
    defined as an attribute, 318  
    header, 170  
Label attribute, 318  
LabelAnimationStyle, 173  
LabelPos, 174  
labels  
    defined, 49  
    in legends, 50

- on axes, 50
  - parameters specific to, 50
  - titles as, 50
- Layout, 175
- leading, 50
- left axis
- active label for optional title, 184
  - box background for optional title, 184
  - box background for title, 178
  - color, 179
  - numeric format, 179
  - optional custom labels, 174, 181
  - optional title, 184
  - scrolling, 182
  - tic label style, 185
  - tic length, 183
  - title, 177
- LeftActiveLabels, 176
- LeftAxis, 177
- LeftAxisTitle, 177
- LeftAxisTitleActiveLabel, 178
- LeftAxisTitleBox, 178
- LeftColor, 179
- LeftFormat, 179
- LeftLabels, 174, 181
- LeftMargins, 182
- LeftScroll, 182
- LeftTicLength, 183
- LeftTics, 185
- LeftTitle, 184
- LeftTitleActiveLabel, 184
- LeftTitleBox, 184
- legend
- active labels in, 186
  - box background for, 188, 189
  - labels, 50
  - layout, 194
  - location of, 187
  - optional items, 191
- Legend, 186
- LegendActiveLabels, 186
- LegendAnimationStyle, 187
- LegendAxis, 187
- LegendBox, 188
- LegendBoxSize, 189
- LegendDwellAnimationHighlightBorderStyle, 190
- LegendDwellAnimationHighlightFill, 190
- LegendDwellAnimationStyle, 191
- LegendItems, 191
- LegendLayout, 194
- legends
- active labels in, 51
  - boxes around, 51
  - defined, 51
  - layout, 51
  - parameters specific to, 51
- line chart
- 3-D lines, 197
  - active labels for stacked charts, 285
  - defining line sets, 203
  - line data sets, 202
  - line set axes, 198
  - line style, 109, 142, 204
  - line width, 210
  - orientation of bar in, 90, 145, 149, 310
  - parameters specific to a, 31
  - specifying a line drop shadow, 199
  - specifying line animation style, 197
  - stacking in, 150, 240
  - symbols, 110, 205
- line styles, 319
- Line3DDepth, 197
- LineAnimationStyle, 197
- LineAxis, 198
- LineDropShadow, 199
- LineFillPattern, 200
- LineLabels[n], 201
- lines, 139
- LineSet[n], 202
- LineSets, 203
- LineStyle, 204
- LineSymbol, 205
- LineSymbolSpotlights, 207
- linethrough, 50
- LineType attribute, 319
- LineValueLabel, 209
- LineValueLabelBox, 209
- LineValueLabelStyle, 210
- LineWidth, 210
- LineWidth attribute, 319
- Locale, 211
- locating files, 321

## M

- maxLineAdvance*, 50
- MeanActiveLabels, 211
- MeanColor, 212
- MeanLine, 212
- MeanSymbol, 213
- MedianColor, 213
- MetaData, 214
- MinimumDataPoints, 214
- multi-chart
  - charts appearing in a, 103
  - defining charts with URLs, 107
  - layout of charts in a, 175
  - replacement for NFPParamScript, 106
  - sashes dividing charts, 260
- multiple
  - specifying pie edge highlights, 167, 231, 267
- multiple chart
  - specifying a pie drop shadow, 230
  - specifying pie animation, 284
  - specifying pie slice animation, 272
  - specifying spotlight overlays over pies, 232, 236

## N

- Name attribute, 320
- NaturalDisplayOrder, 215
- NFPParamURL, 11, 12
- NodeBox, 215
- NodeDrag, 216, 221

NodeLabel, 216  
Nodes, 217  
note  
  active labels, 218  
  active labels for, 52  
  arrows to and from, 219  
  axis location in chart, 220  
  box background, 102, 132, 221, 242, 243, 245, 246, 248, 249  
  boxes, 53  
  defined, 51  
  identifying note sets, 225, 226  
  labels, 52  
  locating in charts using axes, 52  
  parameters specific to, 52  
  sets, 52  
  sets of notes, 222  
  text appearing in, 222  
NoteActiveLabels[n], 218  
NoteArrow, 219  
NoteAxis, 220  
NoteBox, 221  
NoteLabel, 222  
NotesDrawnBeforeData, 226  
NoteSet[n], 222  
NoteSets, 225  
NumberFormat, 226  
numeric labels  
  formatting expressions for, 316

## O

OutlierActiveLabels, 227  
OutlierColor, 227  
OutlierSymbol, 228  
overline, 50

## P

parameter  
  in an include file, 11  
parameter file  
  loading parameters from a, 54  
parameter script, 10  
parameter server, 12  
  connection processing, 13  
parameters  
  dynamically updated from another applet, 13  
  pointing to with URL, 10  
  updated from JavaScript, 14  
Pareto  
  ShowEightyTwentyLines, 271  
  TwentyLineSetName, 311, 312  
pareto chart  
  specifying a fill pattern for 3D bars, 73, 79  
  specifying a fill pattern in 3D grouped stacked bar, 74, 80  
  specifying bar animation, 66  
  specifying bar corners, 69  
  specifying bar drop shadows, 70, 166  
  specifying bar highlights, 73  
  specifying spotlight overlays over bars, 75, 207

PercentileN, 228  
pie chart  
  3-D depth of, 238  
  data, 273, 280  
  keeping the pie round, 238  
  label placement, 277  
  label position, 174  
  numeric data format, 275  
  parameters specific to a, 32, 35  
  pre-exploding slices, 279  
  slice appearance, 273  
  slice label appearance, 276, 278, 283, 284  
  slice label background, 276  
  slice label identification, 278  
  slice labels, 277  
  specifying a pie drop shadow, 230  
  specifying pie animation, 284  
  specifying pie edge highlights, 167, 231, 267  
  specifying pie slice animation, 272  
  specifying spotlight overlays over pies, 232, 236  
  starting piece position, 228, 229, 233, 234, 235  
Pie3DDepth, 238  
PieAngle, 229  
PieAngles, 229  
PieBackgrounds, 229  
PieDropShadow, 230  
PieEdgeHighlights, 231  
PieHighlights, 232  
PieLabel, 233  
PieLabelBox, 233  
PieLabelLocation, 234  
PieLabels, 234  
PieLayout, 233  
PieMarin, 235  
PieSize, 235  
PieSpotlights, 235  
PieSquare, 238  
PolarLabel, 241  
PlotArea, 239  
PlotType, 239  
PolarLabelFormat, 241  
PolarLabelStep, 242  
PolarScale, 242  
PolarSize, 243  
PolarSquare, 243  
PolyActiveLabels, 243  
PolyColor, 244  
PolySet, 244

## R

RadarSize, 245  
RadarSquare, 245  
RadialAxes, 245  
RadialAxesAngles, 246  
RadialAxesColors, 246  
RadialAxesFormat, 246  
RadialAxesTitleActiveLabels, 249  
RadialGrids, 249  
region  
  background type, 313  
  border color, 313

- defined, 52
- images in a, 317
- parameters specific to, 53
- RelativeBoxSymbolWidth, 250
- right axis
  - active label for optional title, 258
  - active labels, 250
  - background for optional title, 258
  - color, 252
  - custom tic mark labels, 254
  - numeric label formatting, 253
  - optional title, 257
  - scrolling, 255
  - tic length, 256
  - tic mark labels, 256
  - title, 251
  - title active label, 251
  - title background, 252
- RightActiveLabels, 250
- RightAxis, 250
- RightAxisTitle, 251
- RightAxisTitleActiveLabel, 251
- RightAxisTitleBox, 252
- RightColor, 252
- RightDrawMinorTics, 252
- RightFormat, 253
- RightLabels, 254
- RightMargins, 255
- RightScroll, 255
- RightTicLength, 256
- RightTics, 256
- RightTitle, 257
- RightTitleActiveLabel, 258
- RightTitleBox, 258
- RubberbandBorderStyle, 259
- RubberbandFill, 259

## S

- Sash, 260
- ScaleFactor, 261
- ScaleMode, 261
- scroll bars
  - location in chart, 15
- scrolling, 46
- SectorActiveLabels[n], 263
- SectorBorders, 264
- SectorColors, 264
- SectorData, 264
- SectorDelete, 265
- SectorDrag, 266
- SectorEdgeHighlights, 267
- SectorLabels, 268
- Sectors, 270
- ShowGroupStackLabels, 271
- SliceAnimationStyle, 272
- SliceBorder, 273
- SliceColors, 273
- SliceData, 273
- SliceFormat, 275
- SliceLabel, 276
- SliceLabelBox, 276

- SliceLabelContent, 277
- SliceLabelContentDelimiter, 277
- SliceLabelLine, 278
- SliceLabels, 278
- SliceLabelStyle, 278
- SlicePos, 279
- Slices, 280
- SliceSet, 283
- SliceSets, 284
- SliceSlide, 284
- StackDisplayOrder, 284
- stacking, 150, 240
- StackLabel, 285
- stock
  - specifying bar drop shadows, 70, 166
- stock chart
  - 3-D lines, 197
  - active labels for stacked charts, 285
  - active labels on lines, 201
  - axes for data sets, 113
  - axis mapping, 286
  - bar and tic display, 292
  - data, 288
  - data set display attributes, 291
  - data values, 117, 118
  - defining line sets, 203
  - line data sets, 202
  - line set axes, 198
  - line style, 109, 142, 204
  - line width, 210
  - orientation of bars in, 90, 145, 149, 310
  - override default active labels, 290
  - parameters specific to a, 39
  - specifying a fill pattern for 3D bars, 73, 79
  - specifying a fill pattern in 3D grouped stacked bar, 74, 80
  - specifying active labels on bars, 65
  - specifying bar animation, 66
  - specifying bar corners, 69
  - specifying bar highlights, 73
  - specifying spotlight overlays over bars, 75, 207
  - stacking in, 150, 240
  - symbols, 110, 205
- Stock chart
  - specifying a line drop shadow, 199
- Stock Chart
  - specifying stock animation, 286
- StockAnimationStyle, 286
- StockAxis, 286
- StockColorTable, 287
- StockData[n], 288
- StockLabels[n], 290
- StockSets, 291
- StockWidth, 292
- strings, 318, 320
- strip chart
  - appending data sets, 57, 247, 295
  - axes for data sets, 113
  - data update, 309, 310
  - data values, 117, 118
  - layout, 293
  - parameters specific to a, 41

StripLayout, 293  
symbols  
  for bubbles, 100  
  for lines, 110, 205

## T

Target attribute, 320  
TaskColorTable, 294  
TaskHeight, 295  
text strings, 318, 320  
Tic Locations, 295  
tic marks, 46  
  layout, 46  
time chart  
  axes for data sets, 113  
  data values, 117, 118  
  parameters specific to a, 42  
time values, 148, 149, 180, 181, 253, 254, 302, 303, 323  
  calculating numeric units, 325  
  calculating relative numbers, 324  
title, 45  
  active labels for, 44, 46  
  boxes, 53  
  location in chart, 15  
ToggleDataVisibility, 299  
top axis  
  active label for title, 300  
  active labels, 299  
  color, 301  
  custom tic mark labels, 295, 303  
  numeric label format, 147, 302  
  scale, 304  
  scrolling, 262, 305  
  tic label layout, 297, 298, 307  
  tic length, 309  
  tic mark label appearance, 296, 306  
  title, 61, 300  
  title background, 62, 301  
TopActiveLabels, 299  
TopAxis, 299  
TopAxisTitle, 61, 300  
TopAxisTitleActiveLabel, 300  
TopAxisTitleBox, 62, 301  
TopColor, 301  
TopDrawMinorTics, 301  
TopFormat, 147, 302

TopLabels, 303  
TopMargins, 304  
TopScale, 304  
TopScroll, 262, 305  
TopTicLayout, 297, 298, 307  
TopTicLength, 309  
TopTics, 296, 306  
TwentyLineSetName, 311, 312

## U

underline, 50  
UniqueTaskColors, 310  
Update, 309  
URL attribute, 321

## W

WhiskerType, 310  
Width attribute, 321  
Windows font names, 316  
WW, 317

## X

XAxis attribute, 321  
X-Y chart  
  3-D lines, 197  
  defining line sets, 203  
  line data sets, 202  
  line set axes, 198  
  line style, 109, 142, 204  
  line width, 210  
  parameters specific to a, 43  
  specifying a line drop shadow, 199  
  symbols, 110, 205

## Y

YAxis attribute, 322

## Z

Zoom  
  AxisZoom. *See*

---

## General Information

NetCharts, NetCharts Pro, NetCharts Server, NetCharts Designer, NetCharts Performance Dashboards, Chart Definition Language and Visual Mining are trademarks of Visual Mining, a division of Tervela, Inc.

Other product names used in this document are trademarks of their respective owners.

© 1996-2015 Visual Mining, a division of Tervela, Inc. All rights reserved.

### **Visual Mining, a division of Tervela, Inc.**

2301 Research Blvd.  
Suite 201  
Rockville, MD 20850

#### **Inquiries**

<b>General Phone</b>	800.308.0731
<b>International</b>	+1.301.795.2200
<b>Fax</b>	301.947.8293
<b>General Inquiries</b>	<a href="mailto:info@visualmining.com">info@visualmining.com</a>
<b>Customer Support</b>	<a href="mailto:support@visualmining.com">support@visualmining.com</a>
<b>Sales</b>	<a href="mailto:sales@visualmining.com">sales@visualmining.com</a>
<b>Press and Media Inquiries</b>	<a href="mailto:marketing@visualmining.com">marketing@visualmining.com</a>

#### **Web**

<http://www.visualmining.com>