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

TABLE OF CONTENTS ............................................................................................................................. 2

1. AN OVERVIEW OF CHART DEFINITION LANGUAGE (CDL) ................................................................. 4
   1.1 CDL STATEMENTS .................................................................................................................................. 4
       Where Attributes Fit In .......................................................................................................................... 5
       CDL Statement Types ......................................................................................................................... 6
   1.2 VISUAL MINING’S SUITE OF CHART GENERATION SOLUTIONS .................................................. 7
       NetCharts Pro ....................................................................................................................................... 7
       NetCharts Designer .............................................................................................................................. 7
       NetCharts Server .................................................................................................................................. 8
       NetCharts Performance Dashboards ...................................................................................................... 8

2. NETCHARTS PRO APPLETS AND CDL ............................................................................................. 9
   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
       A Technical Note about Server Connection Processing ...................................................................... 13
   DYNAMIC UPDATES FROM ANOTHER APPLET .................................................................................. 13
       Loading Data Parameters ................................................................................................................... 14
   DYNAMIC UPDATES FROM JavaScript .................................................................................................. 14

3. ANATOMY OF A CHART ..................................................................................................................... 15

4. TYPES OF AXES ..................................................................................................................................... 17
   STANDARD AXES ................................................................................................................................... 17
   MULTI-SCALED SINGLE AXES ............................................................................................................... 17
   MULTIPLE AXES IN A SINGLE DIRECTION .......................................................................................... 17
   VARIABLE AXIS LABELING ............................................................................................................... 17
   LOGARITHMIC AXES ............................................................................................................................ 18

5. CDL PARAMETERS ARRANGED BY CHART TYPE ........................................................................ 19
   BAR CHART AND 3D BAR 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
   
   SIMPLDATE 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:

```plaintext
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:*

```plaintext
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:*

```plaintext
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.

<table>
<thead>
<tr>
<th>Bgcolor</th>
<th>LineColor</th>
<th>SymColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fgcolor</td>
<td>BorderColor</td>
<td>TipColor</td>
</tr>
<tr>
<td>Color</td>
<td>FillColor</td>
<td>SliceColor</td>
</tr>
</tbody>
</table>
1. An Overview of Chart Definition Language (CDL)

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:

```plaintext
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:

![Weekday Network Load](image)

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. An Overview of Chart Definition Language (CDL)

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](image)

**NetCharts Pro**

*NetCharts Pro* ([http://www.visualmining.com/nc-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/](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.

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.

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

```html
<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:

```html
<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:**

```html
<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 a `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:**
<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);
2. NetCharts Pro Applets and CDL

File: MyChart.html

```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:

```html
<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:

```html
<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:**

```java
AppletContext ac = getAppletContext();
NFPiechartApp pie = (NPiechartApp) 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:*

```java
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:*

```javascript
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.

![Chart anatomy, using a stacked bar chart as an example.](image)

Note that the axis labels and axis tics, 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 tic 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 tics – 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:**

```cdl
BottomTics = ("ON",black,"SansSerif",10,80,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:**

```cdl
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.
<table>
<thead>
<tr>
<th>TopTicLocations</th>
<th>Top Axis Tic Mark Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomTicLocations</td>
<td>Bottom Axis Tic Mark Locations</td>
</tr>
<tr>
<td>LeftTicLocations</td>
<td>Left Axis Tic Mark Locations</td>
</tr>
<tr>
<td>RightTicLocations</td>
<td>Right Axis Tic Mark Locations</td>
</tr>
<tr>
<td>TopTicLength</td>
<td>Top Axis Tic Mark Length</td>
</tr>
<tr>
<td>BottomTicLength</td>
<td>Bottom Axis Tic Mark Length</td>
</tr>
<tr>
<td>LeftTicLength</td>
<td>Left Axis Tic Mark Length</td>
</tr>
<tr>
<td>RightTicLength</td>
<td>Right Axis Tic Mark Length</td>
</tr>
</tbody>
</table>

**Logarithmic Axes**

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

```plaintext
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 3DBar Chart**

**Generally Supported Parameter Types**

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

**Specifically Supported Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BarAnimationStyle</td>
<td>BarAnimationStyle = GROW</td>
</tr>
<tr>
<td>Bar3DDepth</td>
<td>Bar3DDepth = Number;</td>
</tr>
<tr>
<td>BarActiveLabels</td>
<td>BarActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</td>
</tr>
<tr>
<td>BarBorder</td>
<td>BarBorder = (LineType, LineWidth, Color);</td>
</tr>
<tr>
<td>BarColorTable[n]</td>
<td>BarColorTable[1-50] = color1,color2...;</td>
</tr>
<tr>
<td>BarColorTable[n]P[m]</td>
<td>BarColorTable[1-50]P[1-50] = color1,color2...;</td>
</tr>
<tr>
<td>BarCorners</td>
<td>BarCorners = (topleft, topright, bottomright, bottomleft);</td>
</tr>
<tr>
<td>BarDropShadow</td>
<td>BarDropShadow = (color, offsetx, offset, size);</td>
</tr>
<tr>
<td>BarFillPattern</td>
<td>BarFillPattern = (type, Color1, Color2, imageURL), ...;</td>
</tr>
<tr>
<td>BarFillPattern[n]P[m]</td>
<td>BarFillPattern [1-50]P[1-50] = (type, Color1, Color2, imageURL), ...;</td>
</tr>
<tr>
<td></td>
<td>(for STACKEDGROUPED BarChart only)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>BarHighlights</strong></td>
<td>Barhighlights = (type,start,stop,top,right, bottom,left,width,height,topLeft,topRight,bottomRight,bottomLeft), ...;</td>
</tr>
<tr>
<td><strong>BarRightFillPattern</strong></td>
<td>BarRightFillPattern = (type, color1, color2, imageUrl), ...;</td>
</tr>
<tr>
<td><strong>BarRightFillPattern[n][p][m]</strong></td>
<td>BarRightFillPattern[1-50][p][1-50] = (type, color1, color2, imageUrl), ...; (for STACKEDGROUPED BarChart only)</td>
</tr>
<tr>
<td><strong>BarSpotlights</strong></td>
<td>BarSpotlights = (start,stop, center, centeroffsetx, centeroffsety, focusoffsetx, focusoffsety, radius), ...;</td>
</tr>
<tr>
<td><strong>BarSymbol</strong></td>
<td>BarSymbol = (BarSymbolType, BarColor);</td>
</tr>
<tr>
<td><strong>BarTopFillPattern</strong></td>
<td>BarTopFillPattern = (type, Color1, Color2, imageUrl), ...;</td>
</tr>
<tr>
<td><strong>BarTopFillPattern[n][p][m]</strong></td>
<td>BarTopFillPattern[1-50][p][1-50] = (type, color1, color2, imageUrl), ...; (for STACKEDGROUPED BarChart only)</td>
</tr>
<tr>
<td><strong>BarValueLabel</strong></td>
<td>BarValueLabel = (mode, color, font name, width);</td>
</tr>
<tr>
<td><strong>BarValueLabelBox</strong></td>
<td>BarValueLabelBox = (color, mode, depth);</td>
</tr>
<tr>
<td><strong>BarValueLabelStyle</strong></td>
<td>BarValueLabelStyle = labelposition1, labelposition2, ...;</td>
</tr>
<tr>
<td><strong>BarWidth</strong></td>
<td>BarWidth = Percent1,Percent2,...;</td>
</tr>
<tr>
<td><strong>DataAxis</strong></td>
<td>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</td>
</tr>
<tr>
<td><strong>DataSets</strong></td>
<td>DataSets = (Label1, Color1, Type1), (Label2, Color2, Type2), ...;</td>
</tr>
<tr>
<td><strong>DataSet[n]</strong></td>
<td>DataSet[1-50] = a, b, c, ...;</td>
</tr>
<tr>
<td><strong>DataSet[n][p][m]</strong></td>
<td>DataSet[1-50][p][1-50] = a, b, c, ...;</td>
</tr>
<tr>
<td><strong>DataLegend</strong></td>
<td>DataLegend = ON</td>
</tr>
<tr>
<td><strong>DataLegendGrid</strong></td>
<td>DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);</td>
</tr>
<tr>
<td><strong>DataLegendGridLine</strong></td>
<td>DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);</td>
</tr>
<tr>
<td><strong>GraphType</strong></td>
<td>GraphType = Type;</td>
</tr>
<tr>
<td><strong>GraphLayout</strong></td>
<td>GraphLayout = Type;</td>
</tr>
<tr>
<td><strong>GroupStackLabels</strong></td>
<td>GroupStackLabels = &quot;label1&quot;,&quot;label2&quot;,...;</td>
</tr>
<tr>
<td><strong>GroupStackSegment Labels</strong></td>
<td>GroupStackSegmentLabels = &quot;label1&quot;,&quot;label2&quot;,...;</td>
</tr>
<tr>
<td><strong>PlotArea</strong></td>
<td>PlotArea = (xlocation, ylocation, width, height);</td>
</tr>
<tr>
<td><strong>ShowGroupStackLabels</strong></td>
<td>ShowGroupStackLabels = ON</td>
</tr>
<tr>
<td><strong>StackDisplayOrder</strong></td>
<td>StackDisplayOrder = mode;</td>
</tr>
<tr>
<td><strong>StackedBar Connectors</strong></td>
<td>StackedBarConnectors = OFF</td>
</tr>
<tr>
<td><strong>StackLabel</strong></td>
<td>StackLabel = Type;</td>
</tr>
<tr>
<td><strong>ViewPoint</strong></td>
<td>ViewPoint = (CARTENSIAN</td>
</tr>
<tr>
<td><strong>ZAxisLabels</strong></td>
<td>ZAxisLabels = ((ON</td>
</tr>
</tbody>
</table>
**Box Chart**

### Generally Supported Parameter Types

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

### Specifically Supported Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoxActiveLabels</td>
<td>BoxActiveLabels = (&quot;Label1&quot;,&quot;URL1&quot;,&quot;Target1&quot;), ...;</td>
</tr>
<tr>
<td>BoxFence</td>
<td>BoxFence = ON</td>
</tr>
<tr>
<td>BoxFillPattern</td>
<td>BoxFillPattern = (type, color1, color2, imageURL), ...;</td>
</tr>
<tr>
<td>BoxHeight</td>
<td>BoxHeight = Height;</td>
</tr>
<tr>
<td>BoxLabels</td>
<td>BoxLabels = &quot;Label1&quot;,&quot;Label2&quot;,...;</td>
</tr>
<tr>
<td>BoxLimitLines</td>
<td>BoxLimitLines = (limit1-l,limit1-2,...limit1-N),...(limitM-1, limitM-2,... limitM-N);</td>
</tr>
<tr>
<td>BoxLimitLineStyle</td>
<td>BoxLimitLineStyle = (type1,width1,color1),... (typeN,widthN,colorN);</td>
</tr>
<tr>
<td>BoxSymbolWidth</td>
<td>BoxSymbolWidth = Percent;</td>
</tr>
<tr>
<td>BoxWidth</td>
<td>BoxWidth = Percent1,Percent2,...;</td>
</tr>
<tr>
<td>DataAxis</td>
<td>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</td>
</tr>
</tbody>
</table>
Bubble Chart

Generally Supported Parameter Types

ActiveLabels[1-50] = ("Labell", "URL1", "Target1"), ...;
See also Chapter 6, Active Labels and Drilldown

Axis
See Chapter 6,
Axis Modifications 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, Color 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, Notes, or Annotations

Title
Title
See Chapter 6, Labels for various parameters available.

TitleBox
TitleBox
See Chapter 6, Regions, or Boxes for various parameters available.

**Specifically Supported Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDataPoint</td>
<td>AddDataPoint = (&quot;Name&quot;, X, Y, Z, &quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</td>
</tr>
<tr>
<td>BubbleAxis</td>
<td>BubbleAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</td>
</tr>
<tr>
<td>BubbleColorTable[n]</td>
<td>BubbleColorTable[1-50] = color1,color2...;</td>
</tr>
<tr>
<td>BubbleFillPattern</td>
<td>BubbleFillPattern = (type, color1, color2, imageURL), ...;</td>
</tr>
<tr>
<td>BubbleScale</td>
<td>BubbleScale = (MinValue, MaxValue, AREA</td>
</tr>
<tr>
<td>BubbleSets</td>
<td>BubbleSets = (&quot;Name1&quot;, Color1), (&quot;Name2&quot;, Color2), ...;</td>
</tr>
<tr>
<td>BubbleSets[n]</td>
<td>BubbleSets[n] = (x,y,z), (x,y,z), ...;</td>
</tr>
<tr>
<td>BubbleSymbol</td>
<td>BubbleSymbol = (SymType, MaxSize, Style, BorderColor, BorderWidth, SymbolColor, ShadowWidth), ...;</td>
</tr>
<tr>
<td>BubbleSymbolAnimationStyle</td>
<td>BubbleSymbolAnimationStyle = SCALE</td>
</tr>
<tr>
<td>LineStyle</td>
<td>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</td>
</tr>
<tr>
<td>LineDropShadow</td>
<td>LineDropShadow = (color, offsetx, offsety, size);</td>
</tr>
<tr>
<td>PlotArea</td>
<td>PlotArea = (xlocation, ylocation, width, height);</td>
</tr>
</tbody>
</table>

**Combo Chart**

**Generally Supported Parameter Types**

ActiveLabels[1-50]
ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...;
See also Chapter 6, Active Labels and Drilldown
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>See Chapter 6. Axis Modifications for various parameters available</td>
</tr>
<tr>
<td>Axis Thickness</td>
<td>AxisThickness = 15;</td>
</tr>
<tr>
<td>Background</td>
<td>Background = (Color, BorderType, BorderWidth, &quot;ImageURL&quot;, ImageFormat, BorderColor);</td>
</tr>
<tr>
<td>BackgroundFillPattern</td>
<td>BackgroundFillPattern = (type, color1, color2, imageURL), ...;</td>
</tr>
<tr>
<td>BuildAnimationEnabled</td>
<td>BuildAnimationEnabled = ON</td>
</tr>
<tr>
<td>ChartElementSpacing</td>
<td>ChartElementSpacing = spacing;</td>
</tr>
<tr>
<td>ColorTable</td>
<td>See Chapter 8. Color for application of color attributes</td>
</tr>
<tr>
<td>DataLegend</td>
<td>DataLegend = ON</td>
</tr>
<tr>
<td>DataLegendGrid</td>
<td>DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);</td>
</tr>
<tr>
<td>DataLegendGridLine</td>
<td>DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);</td>
</tr>
<tr>
<td>Grid</td>
<td>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</td>
</tr>
<tr>
<td>Legend</td>
<td>Legend = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment);</td>
</tr>
<tr>
<td>NoteSets</td>
<td>NoteSets = (&quot;Name1&quot;, Justify1), (&quot;Name2&quot;, Justify2), ...; See also Chapter 6, Notes, or Annotations</td>
</tr>
<tr>
<td>Title</td>
<td>See Chapter 6. Labels for various parameters available.</td>
</tr>
<tr>
<td>TitleBox</td>
<td>See Chapter 6. Regions, or Boxes for various parameters available.</td>
</tr>
</tbody>
</table>

Specifically Supported Parameters

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

StackLabel

Line3DDepth

LineAxis

LineColorTable[n]

LineDropShadow

LineFillPattern

LineLabels[n]

LineSets

LineSet[n]

LineStyle

LineSymbol

LineSymbolSpotlights

LineValueLabel

LineValueLabelBox

LineValueLabelStyle

LineWidth

StackDisplayOrder

StackedBarConnectors

StackLabel

---

**Diagram/Map Chart**

Generally Supported Parameter Types

---

ActiveLabels[1-50] = ("Label1", "URL1", "Target1"), ...

See also Chapter 6, *Active Labels and Drilldown*

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

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

See Chapter 8, *Color* for application of color attributes

Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);

NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...

See also Chapter 6, *Notes, or Annotations*
Title

See Chapter 6, *Labels* for various parameters available.

TitleBox

See Chapter 6, *Regions, or Boxes* 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, *Active Labels and Drilldown*

**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, *Color* for application of color attributes

**Legend**

Legend = ("Label", Color, "FontName", FontSize, Angle, 
interiorAlignment);

**NoteSets**

NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...; 
See also Chapter 6, *Notes, or Annotations*

**Title**

See Chapter 6, *Labels* for various parameters available.

**TitleBox**

See Chapter 6, *Regions, or Boxes* for various parameters available.

### Specifically Supported Parameters

**Dials**

Dials = (Name, StartAngle, StopAngle,RadiusPercentage, 
NONE|INSIDE|OUTSIDE), ...;
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialActiveLabels</td>
<td>DialActiveLabels = (Name, Label, URL, Target), ...;</td>
</tr>
<tr>
<td>DialBorders</td>
<td>DialBorders = (Name, Type, Thickness, Color, NONE</td>
</tr>
<tr>
<td>DialClip</td>
<td>DialClip = clipType;</td>
</tr>
<tr>
<td>DialClipPad</td>
<td>DialClipPad = N;</td>
</tr>
<tr>
<td>DialDelete</td>
<td>DialDelete = (Name</td>
</tr>
<tr>
<td>DialFills</td>
<td>DialFills = (Name, Color, NONE</td>
</tr>
<tr>
<td>DialFillPattern</td>
<td>DialFillPattern = (type, color1, color2, imageURL), ...;</td>
</tr>
<tr>
<td>DialFormats</td>
<td>DialFormats = (Name, FLOAT</td>
</tr>
<tr>
<td>DialHandAnimationStyle</td>
<td>DialHandAnimationStyle = GROW</td>
</tr>
<tr>
<td>DialOuterBorder</td>
<td>DialOuterBorder = (color1, color2, width);</td>
</tr>
<tr>
<td>DialOuterFillPattern</td>
<td>DialOuterFillPattern = (type, color1, color2, imageURL);</td>
</tr>
<tr>
<td>DialScale</td>
<td>DialScale = (Name, MinValue, MaxValue, StepValue), ...;</td>
</tr>
<tr>
<td>DialSize</td>
<td>DialSize = (minWidth, minHeight, maxWidth, maxHeight);</td>
</tr>
<tr>
<td>DialSectorAnimationStyle</td>
<td>DialSectorAnimationStyle = GROW</td>
</tr>
<tr>
<td>DialSquare</td>
<td>DialSquare = Switch;</td>
</tr>
<tr>
<td>DialTics</td>
<td>DialTics = (Name, Color, LineWidth, PercentofRadius), ...;</td>
</tr>
<tr>
<td>DialTicLabels</td>
<td>DialTicLabels = (Name, Label1, Label2, ..., LabelN), ...;</td>
</tr>
<tr>
<td>DialTicLabelStyles</td>
<td>DialTicLabelStyles = (Name, ON</td>
</tr>
<tr>
<td>Hands</td>
<td>Hands = (Name, TipColor, ShaftColor, DialName, HandLabel), ...;</td>
</tr>
<tr>
<td>HandActiveLabels</td>
<td>HandActiveLabels = (Name, Label, URL, Target), ...;</td>
</tr>
<tr>
<td>HandBorders</td>
<td>HandBorders = (Name, lineHeight, LineWidth, Color);</td>
</tr>
<tr>
<td>HandButtonBorder</td>
<td>HandButtonBorder = (lineType, LineWidth, Color);</td>
</tr>
<tr>
<td>HandButtonEdgeHighlights</td>
<td>HandButtonEdgeHighlights = (start, stop, gap, size), ...;</td>
</tr>
<tr>
<td>HandData</td>
<td>HandData = (Name, Value, Length), ...;</td>
</tr>
<tr>
<td>HandDelete</td>
<td>HandDelete = (Name</td>
</tr>
<tr>
<td>HandDrag</td>
<td>HandDrag = &quot;ON&quot;</td>
</tr>
<tr>
<td>HandDropShadow</td>
<td>HandDropShadow = (color, offsetX, offsetY, size);</td>
</tr>
<tr>
<td>HandLabels</td>
<td>HandLabels = (&quot;Name&quot;, &quot;ON</td>
</tr>
<tr>
<td>HandStyles</td>
<td>HandStyles = (Name, NEEDLELINE</td>
</tr>
<tr>
<td>Sectors</td>
<td>Sectors = (Name, Color, DialName, OuterRadius, InnerRadius, SectorLabel), ...;</td>
</tr>
<tr>
<td>SectorActiveLabels</td>
<td>SectorActiveLabels = (Name, Label, URL, Target), ...;</td>
</tr>
<tr>
<td>SectorBorders</td>
<td>SectorBorders = (Name, Type, LineWidth, Color), ...;</td>
</tr>
<tr>
<td>SectorColors</td>
<td>SectorColors = (Name</td>
</tr>
</tbody>
</table>
### SectorData

SectorData = (Name, StartValue, StopValue), ...

### SectorDelete

SectorDelete = (Name|ALL), ...

### SectorDrag

SectorDrag = "ON"|"OFF";

### SectorEdgeHighlights

SectorEdgeHighlights = (start,stop,gap,size), ...

### SectorFillPattern

SectorFillPattern = (type, color1, color2, imageURL), ...

### SectorLabels

SectorLabels = (Name, ON|OFF, LabelPos, Color, FontName, FontSize, Angle, interiorAlignment), ...

---

### Heat Map Chart

**Generally Supported Parameter Types**

**ChartElementSpacing**

ChartElementSpacing = spacing;

**Legend**

Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);

**NoteSets**

NoteSets = ("Name1", Justify1), ("Name2", Justify2), ...

See also Chapter 6, *Notes, or Annotations*

**Title**

See Chapter 6, *Labels* for various parameters available.

**TitleBox**

See Chapter 6, *Regions, or Boxes* for various parameters available.

---

### Specifically Supported Parameters

**CellTextAutoColorThreshold**

CellTextAutoColorThreshold = 0-100 %;

**GridBlockActiveLabels**

GridBlockActiveLabels = (Name, Label, URL, Target), ...

**GridBlockBackgroundColor**

GridBlockBackgroundColor = color;

**GridBlockCellColorType**

GridBlockCellColorType = type;

**GridBlockColors**

GridBlockColors = (color1, color2,...,colorN);

**GridBlockColorSpectrum**

GridBlockColorSpectrum = (color1,color2,min,max,gradientstep);

**GridBlockExpressions**

GridBlockExpressions = ("operator",value1,value2,color),…;

**GridBlockBottomLabel**

GridBlockBottomLabel = ("mode", color, "font name", font size, angle, interiorAlignment);

**GridBlockLeftLabel**

GridBlockLeftLabel = ("mode", color, "font name", font size, angle, interiorAlignment);

**GridBlockRightLabel**

GridBlockRightLabel = ("mode", color, "font name", font size, angle, interiorAlignment);

**GridBlockTopLabel**

GridBlockTopLabel = ("mode", color, "font name", font size, angle, interiorAlignment);
### Generally Supported Parameter Types

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

### Specifically Supported Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar3DDepth</td>
<td><code>Bar3DDepth = Number;</code></td>
</tr>
<tr>
<td>BarActiveLabels</td>
<td><code>BarActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</code></td>
</tr>
<tr>
<td>BarBorder</td>
<td><code>BarBorder = (LineType, LineWidth, Color);</code></td>
</tr>
<tr>
<td>BarColorTable[n]</td>
<td><code>BarColorTable[1-50] = color1,color2...;</code></td>
</tr>
<tr>
<td>BarColorTable[n][p]</td>
<td><code>BarColorTable[1-50][p-50] = color1,color2...;</code></td>
</tr>
<tr>
<td>BarFillPattern</td>
<td><code>BarFillPattern = (type, Color1, Color2, imageURL), ...;</code></td>
</tr>
<tr>
<td>BarFillPattern [1-50][p][1-50]</td>
<td><code>BarFillPattern [1-50][p-50] = (type, Color1, Color2, imageURL), ...;</code> (for STACKEDGROUPED BAR)</td>
</tr>
<tr>
<td>BarValueLabel</td>
<td><code>BarValueLabel = (mode, color, font name, width);</code></td>
</tr>
<tr>
<td>BarValueLabelBox</td>
<td><code>BarValueLabelBox = (color, mode, depth);</code></td>
</tr>
<tr>
<td>BarValueLabelStyle</td>
<td><code>BarValueLabelStyle = labelposition1, labelposition2, ...;</code></td>
</tr>
<tr>
<td>BarWidth</td>
<td><code>BarWidth = Percent1,Percent2,...;</code></td>
</tr>
<tr>
<td>DataAxis</td>
<td><code>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</code></td>
</tr>
<tr>
<td>DataSets</td>
<td><code>DataSets = (Label1, Color1, Type1), (Label2, Color2, Type2), ...;</code></td>
</tr>
<tr>
<td>DataSet[n]</td>
<td><code>DataSet[1-50] = a, b, c, ...;</code></td>
</tr>
<tr>
<td>DataSet[n][p]</td>
<td><code>DataSet[1-50][p-50] = a, b, c, ...;</code></td>
</tr>
<tr>
<td>GraphType</td>
<td><code>GraphType = Type;</code></td>
</tr>
<tr>
<td>GraphLayout</td>
<td><code>GraphLayout = Type;</code></td>
</tr>
<tr>
<td>GroupStackLabels</td>
<td><code>GroupStackLabels = &quot;label1&quot;,&quot;label2&quot;,...;</code></td>
</tr>
<tr>
<td>GroupStackSegmentLabels</td>
<td><code>GroupStackSegmentLabels = &quot;label1&quot;,&quot;label2&quot;,...;</code></td>
</tr>
<tr>
<td>HistogramBin</td>
<td><code>HistogramBin = (HistogramBinType, HistogramBinSize);</code></td>
</tr>
<tr>
<td>HistogramScale</td>
<td><code>HistogramScale = (histogramMinValue, histogramMaxValue);</code></td>
</tr>
<tr>
<td>HistogramType</td>
<td>`HistogramType = BYNUMER</td>
</tr>
<tr>
<td>PlotArea</td>
<td><code>PlotArea = (xlocation, ylocation, width, height);</code></td>
</tr>
<tr>
<td>ShowGroupStackLabels</td>
<td>`ShowGroupStackLabels = ON</td>
</tr>
<tr>
<td>StackDisplayOrder</td>
<td><code>StackDisplayOrder = mode;</code></td>
</tr>
<tr>
<td>StackLabel</td>
<td><code>StackLabel = Type;</code></td>
</tr>
</tbody>
</table>

**Line Chart**

**Generally Supported Parameter Types**

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

- **Axis**
  
  See Chapter 6, *Axis Modifications for various parameters available*
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Axis Thickness</strong></td>
<td>( \text{AxisThickness} = 15; )</td>
</tr>
<tr>
<td><strong>Background</strong></td>
<td>( \text{Background} = (\text{Color}, \text{BorderType}, \text{BorderWidth}, \text{&quot;ImageUrl&quot;}, \text{ImageFormat}, \text{BorderColor}); )</td>
</tr>
<tr>
<td><strong>BackgroundFillPattern</strong></td>
<td>( \text{BackgroundFillPattern} = (\text{type}, \text{color1}, \text{color2}, \text{imageURL}), \ldots; )</td>
</tr>
<tr>
<td><strong>BuildAnimationEnabled</strong></td>
<td>( \text{BuildAnimationEnabled} = \text{ON</td>
</tr>
<tr>
<td><strong>ChartElementSpacing</strong></td>
<td>( \text{ChartElementSpacing} = \text{spacing}; )</td>
</tr>
<tr>
<td><strong>ColorTable</strong></td>
<td>See Chapter 8, <em>Color</em> for application of color attributes</td>
</tr>
<tr>
<td><strong>Grid</strong></td>
<td>( \text{Grid} = (\text{LineColor1}, \text{bgColor1}, \text{borderColor1}, \text{bgImage1}, \text{ImageFormat1}), \ldots; )</td>
</tr>
<tr>
<td><strong>Legend</strong></td>
<td>( \text{Legend} = (&quot;\text{Label}&quot;, \text{Color}, &quot;\text{FontName}&quot;, \text{FontSize}, \text{Angle}, \text{interiorAlignment}); )</td>
</tr>
<tr>
<td><strong>NoteSets</strong></td>
<td>( \text{NoteSets} = (&quot;\text{Name1}&quot;, \text{Justify1}), (&quot;\text{Name2}&quot;, \text{Justify2}), \ldots; )</td>
</tr>
<tr>
<td><strong>Title</strong></td>
<td>See also Chapter 6, <em>Notes, or Annotations</em></td>
</tr>
<tr>
<td><strong>TitleBox</strong></td>
<td>See Chapter 6, <em>Labels</em> for various parameters available.</td>
</tr>
<tr>
<td><strong>Specifically Supported Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ChartElementSpacing</strong></td>
<td>( \text{ChartElementSpacing} = \text{spacing}; )</td>
</tr>
<tr>
<td><strong>DataLegend</strong></td>
<td>( \text{DataLegend} = \text{ON</td>
</tr>
<tr>
<td><strong>DataLegendGrid</strong></td>
<td>( \text{DataLegendGrid} = (\text{lineColor}, \text{bgColor}, \text{borderColor}, \text{bgImage}, \text{bgImageType}); )</td>
</tr>
<tr>
<td><strong>DataLegendGridLine</strong></td>
<td>( \text{DataLegendGridLine} = (\text{lineType1}, \text{lineStyle1}, \text{lineWidth}); )</td>
</tr>
<tr>
<td><strong>DwellOffset</strong></td>
<td>( \text{DwellOffset} = \text{size}; )</td>
</tr>
<tr>
<td><strong>GraphType</strong></td>
<td>( \text{GraphType} = \text{Type}; )</td>
</tr>
<tr>
<td><strong>Line3DDepth</strong></td>
<td>( \text{Line3DDepth} = \text{depth}; )</td>
</tr>
<tr>
<td><strong>LineAxis</strong></td>
<td>( \text{LineAxis} = (\text{XAxis1}, \text{YAxis1}), (\text{XAxis2}, \text{YAxis2}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineFillPattern</strong></td>
<td>( \text{LineFillPattern} = (\text{type}, \text{color1}, \text{color2}, \text{imageURL}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineSets</strong></td>
<td>( \text{LineSets} = (\text{Name1}, \text{SymColor1}), (\text{Name2}, \text{SymColor2}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineSet[1-50]</strong></td>
<td>( \text{LineSet}[1-50] = y1, y2, y3, \ldots; )</td>
</tr>
<tr>
<td><strong>LineStyle</strong></td>
<td>( \text{LineStyle} = (\text{LineType}, \text{LineWidth}, \text{Color}, \text{FillColor}, \text{LineType}, \text{FillType}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineSymbol</strong></td>
<td>( \text{LineSymbol} = (\text{Type}, \text{Size}, \text{Style}, \text{BorderColor}, \text{BorderWidth}, \text{ImageUrl}, \text{SymbolColor}, \text{ShadowWidth}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineSymbolSpotlights</strong></td>
<td>( \text{LineSymbolSpotlights} = (\text{start}, \text{stop}, \text{center}, \text{centeroffsetx}, \text{centeroffsety}, \text{focusoffsetx}, \text{focusoffsety}, \text{radius}), \ldots; )</td>
</tr>
<tr>
<td><strong>LineValueLabelBox</strong></td>
<td>( \text{LineValueLabelBox} = (\text{color}, \text{mode}, \text{depth}); )</td>
</tr>
<tr>
<td><strong>LineValueLabelStyle</strong></td>
<td>( \text{LineValueLabelStyle} = \text{labelposition1}, \text{labelposition2}, \ldots; )</td>
</tr>
<tr>
<td><strong>LineWidth</strong></td>
<td>( \text{LineWidth} = \text{PercentDepth}; )</td>
</tr>
<tr>
<td><strong>PlotArea</strong></td>
<td>( \text{PlotArea} = (\text{xlocation}, \text{ylocation}, \text{width}, \text{height}); )</td>
</tr>
</tbody>
</table>
StackLabel

MultiPie Chart

Generally Supported Parameter Types

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

Specifically Supported Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestFit</td>
<td>BestFit = ON</td>
</tr>
<tr>
<td>LabelPos</td>
<td>LabelPos = Float;</td>
</tr>
<tr>
<td>Pie3DDepth</td>
<td>Pie3DDepth = Pixels;</td>
</tr>
<tr>
<td>PieAngles</td>
<td>PieAngles = Value1, Value2, Value3, Value4, ..., ValueN;</td>
</tr>
<tr>
<td>PieDropShadow</td>
<td>PieDropShadow = (color, offsetx, offsety, size);</td>
</tr>
<tr>
<td>PieEdgeHighlights</td>
<td>PieEdgeHighlights = (start,stop,gap,size), ...;</td>
</tr>
<tr>
<td>PieSquare</td>
<td>PieSquare = ON</td>
</tr>
<tr>
<td>PieBackgrounds</td>
<td>PieBackgrounds = (Region Tuple), ...;</td>
</tr>
<tr>
<td>PieLabel</td>
<td>PieLabel = (State,Color,FontName,FontSize,Angle,InteriorAlignment), ...;</td>
</tr>
<tr>
<td>PieLabelBox</td>
<td>PieLabelBox = = (Region Tuple), ...;</td>
</tr>
<tr>
<td>PieLayout</td>
<td>PieLayout = (Orientation,Row,Columns);</td>
</tr>
<tr>
<td>PieLabelLocation</td>
<td>PieLabelLocation = Location;</td>
</tr>
<tr>
<td>PieMargin</td>
<td>PieMargin = Integer;</td>
</tr>
<tr>
<td>PieLabels</td>
<td>PieLabels = Label1, Label2, Label3, ... LabelN;</td>
</tr>
<tr>
<td>SliceBorder</td>
<td>SliceBorder = (LineType, Width, Color);</td>
</tr>
<tr>
<td>SliceSets</td>
<td>SliceSets = (&quot;Name&quot;, Color, &quot;State&quot;);</td>
</tr>
</tbody>
</table>
SliceSet[N] = Value1, Value2, Value3, Value4, ..., ValueN;
SliceFillPattern = (type, color1, color2, imageURL), ...
SliceFormat = (FormatType, "FormatExpr");
SliceLabel = (State, Color, FontName, FontSize, Angle, interiorAlignment);
SliceLabelBox = (Color, BorderType, BorderWidth);
SliceLabelContent = Content;
SliceLabelContentDelimiter = "delimiter";
SliceLabelLine = (LineStyle, LineWidth, Color);
SliceLabels = Label1, Label2, Label3, Label4, ..., LabelN;
SliceLabelStyle = (Style);
SlicePos[N] = PiePosition1, PiePosition2, ...

**Pareto Chart**

**Generally Supported Parameter Types**

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

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

### Generally Supported Parameter Types

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

### Specifically Supported Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabelPos</td>
<td>LabelPos = Float;</td>
</tr>
<tr>
<td>Pie3DDepth</td>
<td>Pie3DDepth = Pixels;</td>
</tr>
<tr>
<td>PieAngle</td>
<td>PieAngle = Integer;</td>
</tr>
<tr>
<td>PieDropShadow</td>
<td>PieDropShadow = (color, offsetx, offsety, size);</td>
</tr>
<tr>
<td>PieSize</td>
<td>PieSize = (minWidth, minHeight, maxWidth, maxHeight);</td>
</tr>
<tr>
<td>PieSquare</td>
<td>PieSquare = Switch;</td>
</tr>
<tr>
<td>SliceBorder</td>
<td>SliceBorder = (LineType, Width, Color);</td>
</tr>
<tr>
<td>SliceColor</td>
<td>SliceColor = Color1, Color2,Color3 ..., Colorn;</td>
</tr>
<tr>
<td>SliceData</td>
<td>SliceData = Value1, Value2, Value3, Value4, ..., Valuen;</td>
</tr>
<tr>
<td>SliceFillPattern</td>
<td>SliceFillPattern = (type, color1, color2, imageURL), ...;</td>
</tr>
<tr>
<td>SliceFormat</td>
<td>SliceFormat = (FormatType, &quot;FormatExpr&quot;);</td>
</tr>
<tr>
<td>SliceLabel</td>
<td>SliceLabel = (State, Color, FontName, FontSize, Angle, interiorAlignment);</td>
</tr>
<tr>
<td>SliceLabelBox</td>
<td>SliceLabelBox = (Color, BorderType, BorderWidth);</td>
</tr>
<tr>
<td>SliceLabelContent</td>
<td>SliceLabelContent = Content;</td>
</tr>
</tbody>
</table>
SliceLabelContentDelimiter
SliceLabelLine
SliceLabels
SliceLabelStyle
SlicePos
Slices

**Polar Chart**

**Generally Supported Parameter Types**

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

**Specifically Supported Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineSets</td>
<td>LineSets = (Name1, SymColor1), (Name2, SymColor2), ...;</td>
</tr>
<tr>
<td>LineSet[n]</td>
<td>LineSet[1-50] = (x1,y1), (x2,y2), (x3,y3), ...;</td>
</tr>
<tr>
<td>LineStyle</td>
<td>LineStyle = (LineType, LineWidth, Color, FillColor, LineType, FillType), ...;</td>
</tr>
<tr>
<td>LineSymbol</td>
<td>LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth), ...;</td>
</tr>
</tbody>
</table>
CDL Parameter Arranged by Chart Style

Radar Chart

## Generally Supported Parameter Types

### ActiveLabels[1-50]

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

### Background

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

### BackgroundFillPattern

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

### ChartElementSpacing

ChartElementSpacing = spacing;

### ColorTable

See Chapter 8, Color 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, Notes, or Annotations
Title
See Chapter 6, *Labels* for various parameters available.

TitleBox
See Chapter 6, *Regions, or Boxes* for various parameters available.

**Specifically Supported Parameters**

CenterRadius
CenterRadius = radius;

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), ...

LineSymbol
LineSymbol = (Type, Size, Style, BorderColor, BorderWidth, ImageURL, SymbolColor, ShadowWidth), ...

LineValueLabel
LineValueLabel = (mode, color, font name, width);

LineValueLabelBox
LineValueLabelBox = (color, mode, depth);

LineValueLabelStyle
LineValueLabelStyle = labelposition1, labelposition2, ...

PlotArea
PlotArea = (xlocation, ylocation, width, height);

RadarSize
RadarSize = (minWidth, minHeight, maxWidth, maxHeight);

RadarSquare
RadarSquare = mode;

RadialAxes
RadialAxes = ("axisTitle", minValue, maxValue, stepSize), ...

RadialAxesColors
RadialAxesColors = color1, color2, ...

RadialAxesFormat
RadialAxesFormat = (dataType, formatString), ...

RadialAxesLabel
RadialAxesLabel = ("axisLabelMode", axisLabelColor, "axisLabelFont", axisLabelFontSize, axisLabelFontAngle, axisLabelInteriorAlignment), ...

RadialAxesScales
RadialAxesScales = (min1, max1, step1), (min2, max2, step2), ...

RadialAxesTics
RadialAxesTics = ("axisTicLabelMode", axisTicLabelColor, "axisTicLabelFont", axisTicLabelFontSize, axisTicLabelFontAngle, interiorAlignment), ...

RadialAxesTitles
RadialAxesTitles = "title1", "title2", ...

RadialGrids
RadialGrids = (gridRadius, gridLineType, gridLineWidth, gridLineColor, gridAreaColor), ...
Stock Chart

Generally Supported Parameter Types

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

Specifically Supported Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar3Ddepth</td>
<td>Bar3DDepth = Number;</td>
</tr>
<tr>
<td>BarActiveLabels</td>
<td>BarActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</td>
</tr>
<tr>
<td>BarBorder</td>
<td>BarBorder = (LineType, LineWidth, Color);</td>
</tr>
<tr>
<td>BarFillPattern</td>
<td>BarFillPattern = (type, color1, color2, &quot;ImageURL&quot;), ...;</td>
</tr>
<tr>
<td>BarColorTable[n]</td>
<td>BarColorTable[1-50] = color1,color2...;</td>
</tr>
<tr>
<td>BarWidth</td>
<td>BarWidth = Percent1,Percent2,...;</td>
</tr>
<tr>
<td>DataAxis</td>
<td>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</td>
</tr>
<tr>
<td>DataSets</td>
<td>DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, &quot;ImageURL&quot;, ImageFormat1, BorderColor1), ...;</td>
</tr>
<tr>
<td>DataSet[n]</td>
<td>DataSet[1-50] = a, b, c,...;</td>
</tr>
<tr>
<td>DrawOrder</td>
<td>DrawOrder = Symbol, ...;</td>
</tr>
<tr>
<td>GraphType</td>
<td>GraphType = Type;</td>
</tr>
<tr>
<td>GraphLayout</td>
<td>GraphLayout = Type;</td>
</tr>
</tbody>
</table>
CDL Parameter Arranged by Chart Style

*Strip Chart*

**Generally Supported Parameter Types**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActiveLabels[1-50]</strong></td>
<td>ActiveLabels[1-50] = (&quot;Label1&quot;, &quot;URL1&quot;, &quot;Target1&quot;), ...; See also Chapter 6, Active Labels and Drilldown</td>
</tr>
<tr>
<td><strong>Axis</strong></td>
<td>See Chapter 6, Axis Modifications for various parameters available</td>
</tr>
</tbody>
</table>

*GraphType*  
*Line3Ddepth*  
*LineAxis*  
*LineColorTable[n]*  
*LineFillPattern*  
*LineLabels[1-50]*  
*LineSets*  
*LineSet[n]*  
*LineStyle*  
*LineSymbol*  
*LineValueLabel*  
*LineValueLabelBox*  
*LineValueLabelStyle*  
*LineWidth*  
*PlotArea*  
*StackDisplayOrder*  
*StackLabel*  
*StockAxis*  
*StockData[1-50]*  
*StockColorTable[n]*  
*StockFillPattern*  
*StockLabels[1-50]*  
*StockSets*  
*StockWidth*
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis Thickness</td>
<td>AxisThickness = 15;</td>
</tr>
<tr>
<td>Background</td>
<td>Background = (Color, BorderType, BorderWidth, &quot;ImageURL&quot;, ImageFormat, BorderColor);</td>
</tr>
<tr>
<td>BackgroundFillPattern</td>
<td>BackgroundFillPattern = (type, color1, color2, &quot;ImageURL&quot;), ...;</td>
</tr>
<tr>
<td>ChartElementSpacing</td>
<td>ChartElementSpacing = spacing;</td>
</tr>
<tr>
<td>ColorTable</td>
<td>See Chapter 8, Color for application of color attributes</td>
</tr>
<tr>
<td>Grid</td>
<td>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</td>
</tr>
<tr>
<td>Legend</td>
<td>Legend = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment);</td>
</tr>
<tr>
<td>NoteSets</td>
<td>NoteSets = (&quot;Name1&quot;, Justify1), (&quot;Name2&quot;, Justify2), ...;</td>
</tr>
<tr>
<td>Title</td>
<td>See also Chapter 6, Notes, or Annotations</td>
</tr>
<tr>
<td>TitleBox</td>
<td>See Chapter 6, Regions, or Boxes for various parameters available.</td>
</tr>
</tbody>
</table>

### Specifically Supported Parameters

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

**Generally Supported Parameter Types**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveLabels[1-50]</td>
<td>ActiveLabels[1-50] = (&quot;Label1&quot;, &quot;URL1&quot;, &quot;Target1&quot;), ...; See also Chapter 6, Active Labels and Drilldown</td>
</tr>
<tr>
<td>Axis</td>
<td>See Chapter 6.</td>
</tr>
<tr>
<td>Background</td>
<td>Axis Modifications for various parameters available</td>
</tr>
<tr>
<td>BackgroundFillPattern</td>
<td>BackgroundFillPattern = (type, color1, color2, imageUrl), ...;</td>
</tr>
<tr>
<td>BuildAnimationEnabled</td>
<td>BuildAnimationEnabled = ON</td>
</tr>
<tr>
<td>ChartElementSpacing</td>
<td>ChartElementSpacing = spacing;</td>
</tr>
<tr>
<td>ColorTable</td>
<td>See Chapter 8, Color for application of color attributes</td>
</tr>
<tr>
<td>Grid</td>
<td>Grid = (LineColor1, bgColor1, borderColor1, bgImage1, ImageFormat1), ...;</td>
</tr>
<tr>
<td>Legend</td>
<td>Legend = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment);</td>
</tr>
<tr>
<td>NoteSets</td>
<td>NoteSets = (&quot;Name1&quot;, Justify1), (&quot;Name2&quot;, Justify2), ...;</td>
</tr>
<tr>
<td>Title</td>
<td>See also Chapter 6, Notes, or Annotations</td>
</tr>
<tr>
<td>TitleBox</td>
<td>See Chapter 6, Labels for various parameters available.</td>
</tr>
</tbody>
</table>

**Specifically Supported Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataAxis</td>
<td>DataAxis = (XAxis1, YAxis1), (XAxis2, YAxis2), ...;</td>
</tr>
<tr>
<td>DataSets</td>
<td>DataSets = (DataSetName1, Color1, BorderType1, BorderWidth1, &quot;ImageURL&quot;, ImageFormat1, BorderColor1), ...;</td>
</tr>
<tr>
<td>DataSet[n]</td>
<td>DataSet[1-50] = a, b, c, ...;</td>
</tr>
<tr>
<td>PlotArea</td>
<td>PlotArea = (xlocation, ylocation, width, height);</td>
</tr>
<tr>
<td>TaskColorTable[n]</td>
<td>TaskColorTable[1-50] = color1,color2...;</td>
</tr>
<tr>
<td>TaskHeight</td>
<td>TaskHeight = value;</td>
</tr>
<tr>
<td>UniqueTaskColors</td>
<td>UniqueTaskColors = ON</td>
</tr>
</tbody>
</table>

**X-Y Chart**

**Generally Supported Parameter Types**
### CDL Parameter Arranged by Chart Style

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

**Axis**  
See Chapter 6,  
Axis Modifications 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, *Color* 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, *Notes, or Annotations*

**Title**  
See Chapter 6, *Labels* for various parameters available.

**TitleBox**  
See Chapter 6, *Regions, or Boxes* for various parameters available.

### Specifically Supported Parameters

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

<table>
<thead>
<tr>
<th>ActiveClicks</th>
<th>ActiveClicks = Number;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveLabels</td>
<td>ActiveLabels = (&quot;Label1&quot;, &quot;URL1&quot;, &quot;Target1&quot;), ...;</td>
</tr>
<tr>
<td>BackgroundActiveLabel</td>
<td>BackgroundActiveLabel = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>BarActiveLabels</td>
<td>BarActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</td>
</tr>
<tr>
<td>BottomActiveLabels</td>
<td>BottomActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>BottomAxisTitleActiveLabel</td>
<td>BottomAxisTitleActiveLabel = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>BoxActiveLabels</td>
<td>BoxActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;), ...;</td>
</tr>
<tr>
<td>DataPointActiveLabels[1-50]</td>
<td>DataPointActiveLabels[1-50] = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>DwellAnimationHighlightBorderStyle</td>
<td>DwellAnimationHighlightBorderStyle = (lineType, LineWidth, lineColor);</td>
</tr>
<tr>
<td>DwellAnimationHighlightFill</td>
<td>DwellAnimationHighlightFill = (color);</td>
</tr>
<tr>
<td>DwellAnimationStyle</td>
<td>DwellAnimationStyle = HIGHLIGHT</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Parameters Involved, Alphabetically</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoscalePad</td>
<td>AutoscalePad = pad;</td>
</tr>
<tr>
<td>AxisThickness</td>
<td>AxisThickness = thickness;</td>
</tr>
<tr>
<td>BottomActiveLabels</td>
<td>BottomActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>BottomAxesGaps</td>
<td>BottomAxesGaps = value, value, ...;</td>
</tr>
<tr>
<td>BottomAxesLayout</td>
<td>BottomAxesLayout = value, value, ...;</td>
</tr>
<tr>
<td>BottomAxisTitle</td>
<td>BottomAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</td>
</tr>
<tr>
<td>BottomAxisTitleActiveLabel</td>
<td>BottomAxisTitleActiveLabel = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>BottomAxisTitleBox</td>
<td>BottomAxisTitleBox = (Color, BorderType, BorderWidth, &quot;ImageURL&quot;, ImageFormat, BorderColor);</td>
</tr>
<tr>
<td>BottomColor</td>
<td>BottomColor = Color;</td>
</tr>
<tr>
<td>BottomDrawMinorTics</td>
<td>BottomDrawMinorTics = ON</td>
</tr>
<tr>
<td>BottomFormat</td>
<td>BottomFormat = (FormatType, &quot;FormatExpr&quot;, &quot;TimeBase&quot;, &quot;TimeUnit&quot;);</td>
</tr>
<tr>
<td>BottomLabels</td>
<td>BottomLabels = &quot;Label1&quot;, &quot;Label2&quot;, ...;</td>
</tr>
<tr>
<td>BottomMajorTicColor</td>
<td>BottomMajorTicColor = Color;</td>
</tr>
<tr>
<td>BottomMinorTicColor</td>
<td>BottomMinorTicColor = Color;</td>
</tr>
<tr>
<td>BottomMargins</td>
<td>BottomMargins = (LeftSideMargin, RightSideMargin);</td>
</tr>
<tr>
<td>BottomScale</td>
<td>BottomScale = (MinValue, MaxValue, StepValue);</td>
</tr>
<tr>
<td>BottomScaleMode</td>
<td>BottomScaleMode = (mode,logBase), (mode,logBase),...;</td>
</tr>
<tr>
<td>BottomScaleSet</td>
<td>BottomScaleSet = (MinValue, MaxValue, StepValue, Percentage);</td>
</tr>
<tr>
<td>BottomScroll</td>
<td>BottomScroll = (ScrollMin, ScrollMax);</td>
</tr>
<tr>
<td>BottomTics</td>
<td>BottomTics = (&quot;Mode&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment);</td>
</tr>
<tr>
<td>BottomTicLayout</td>
<td>BottomTicLayout = (Mode, SkipCount, StaggerLevels);</td>
</tr>
<tr>
<td>BottomTicLocations</td>
<td>BottomTicLocations = value, value, value, ...;</td>
</tr>
<tr>
<td>BottomTitle</td>
<td>BottomTitle = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, exteriorAlignment);</td>
</tr>
<tr>
<td>BottomTitleActiveLabel</td>
<td>BottomTitleActiveLabel = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>LeftActiveLabels</td>
<td>LeftActiveLabels = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>LeftAxesGaps</td>
<td>LeftAxesGaps = value, value, ...;</td>
</tr>
<tr>
<td>LeftAxesLayout</td>
<td>LeftAxesLayout = value, value, ...;</td>
</tr>
<tr>
<td>LeftAxisTitle</td>
<td>LeftAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);</td>
</tr>
<tr>
<td>LeftAxisTitleActiveLabel</td>
<td>LeftAxisTitleActiveLabel = (&quot;Label&quot;, &quot;URL&quot;, &quot;Target&quot;);</td>
</tr>
<tr>
<td>LeftAxisTitleBox</td>
<td>LeftAxisTitleBox = (Color, BorderType, BorderWidth, &quot;ImageURL&quot;, ImageFormat, BorderColor);</td>
</tr>
</tbody>
</table>
LeftColor
LeftDrawMinorTics
LeftFormat
LeftLabels
LeftMajorTicColor
LeftMinorTicColor
LeftMargins
LeftScale
LeftScaleMode
LeftScaleSet
LeftScroll
LeftTics
LeftTicLayout
LeftTicLocations
LeftTitle
LeftTitleActiveLabel
RightActiveLabels
RightAxesGaps
RightAxesLayout
RightAxisTitle
RightAxisTitleActiveLabel
RightAxisTitleBox
RightColor
RightDrawMinorTics
RightFormat
RightLabels
RightMajorTicColor
RightMinorTicColor
RightMargins
RightScale
RightScaleMode
RightScaleSet
RightScroll

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

RightTics
RightTics = ("Mode", Color, "FontName", FontSize, Angle, interiorAlignment);

RightTicLayout
RightTicLayout = (Mode, SkipCount, StaggerLevels);

RightTicLocations
RightTicLocations = value, value, value, ...;

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

RightTitleActiveLabel
RightTitleActiveLabel = ("Label", "URL", "Target");

TopActiveLabels
TopActiveLabels = ("Label", "URL", "Target");

TopAxesGaps
TopAxesGaps = value, value, ...;

TopAxesLayout
TopAxesLayout = value, value, ...;

TopAxisTitle
TopAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);

TopAxisTitleActiveLabel
TopAxisTitleActiveLabel = ("Label", "URL", "Target");

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

TopColor
TopColor = Color;

TopDrawMinorTics
TopDrawMinorTics = ON | OFF;

TopFormat
TopFormat = (FormatType, "FormatExpr", "TimeBase", "TimeUnit");

TopLabels
TopLabels = "Label1", "Label2", ...;

TopMajorTicColor
TopMajorTicColor = Color;

TopMinorTicColor
TopMinorTicColor = Color;

TopMargins
TopMargins = (LeftSideMargin, RightSideMargin);

TopScale
TopScale = (MinValue, MaxValue, StepValue);

TopScaleMode
TopScaleMode = (mode, logBase), (mode, logBase), ...;

TopScaleSet
TopScaleSet = (MinValue, MaxValue, StepValue, Percentage);

TopScroll
TopScroll = (ScrollMin, ScrollMax);

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

TopTicLayout
TopTicLayout = (Mode, SkipCount, StaggerLevels);

TopTicLocations
TopTicLocations = value, value, value, ...;

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

TopTitleActiveLabel
TopTitleActiveLabel = ("Label", "URL", "Target");

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.

**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**
  
  \[
  \text{Grid} = (\text{LineColor1}, \text{bgColor1}, \text{borderColor1}, \text{bgImage1}, \text{ImageFormat1}), \ldots;
  \]

- **Grid3DDepth**
  
  \[
  \text{Grid3DDepth} = \text{depth};
  \]

- **GridAxis**
  
  \[
  \text{GridAxis} = (\text{XAxis1}, \text{YAxis1}), \ldots;
  \]

- **GridLine**
  
  \[
  \text{GridLine} = (\text{LineType}, \text{LineStyle}, \text{LineWidth}), \ldots;
  \]

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomAxisTitle</td>
<td>BottomAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment)</td>
</tr>
<tr>
<td>BottomTics</td>
<td>BottomTics = (&quot;Mode&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint)</td>
</tr>
<tr>
<td>BottomTitle</td>
<td>BottomTitle = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, exteriorAlignment)</td>
</tr>
<tr>
<td>DwellLabel</td>
<td>DwellLabel = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment)</td>
</tr>
<tr>
<td>Footer</td>
<td>Footer = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, exteriorAlignment)</td>
</tr>
<tr>
<td>Header</td>
<td>Header = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, exteriorAlignment, extend)</td>
</tr>
<tr>
<td>GridBlockLabel</td>
<td>GridBlockLabel = (&quot;mode&quot;, color, &quot;Font name&quot;, Font size, Angle, interiorAlignment)</td>
</tr>
<tr>
<td>LeftAxisTitle</td>
<td>LeftAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment)</td>
</tr>
<tr>
<td>LeftTics</td>
<td>LeftTics = (&quot;Mode&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint)</td>
</tr>
<tr>
<td>LeftTitle</td>
<td>LeftTitle = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, exteriorAlignment, extend)</td>
</tr>
<tr>
<td>Legend</td>
<td>Legend = (&quot;Label&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment)</td>
</tr>
<tr>
<td>RightAxisTitle</td>
<td>RightAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment)</td>
</tr>
<tr>
<td>RightTics</td>
<td>RightTics = (&quot;Mode&quot;, Color, &quot;FontName&quot;, FontSize, Angle, interiorAlignment, backgroundColor, rotationPoint)</td>
</tr>
</tbody>
</table>
RightTitle

RightTitle = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment, exteriorAlignment, extend);

TopAxisTitle

TopAxisTitle = (Label, Color, FontName, FontSize, Angle, interiorAlignment, exteriorAlignment);

TopTics

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

TopTitle

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

---

**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**
  
  DataLegend = ON|OFF;

- **DataLegendGrid**
  
  DataLegendGrid = (lineColor, bgColor, borderColor, bgImage, bgImageType);

- **DataLegendGridLine**
  
  DataLegendGridLine = (lineType1, lineStyle1, lineWidth1);

- **Legend**
  
  Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);

- **LegendActiveLabels**
  
  LegendActiveLabels = ("Label", "URL", "Target"), ...

- **LegendAxis**
  
  LegendAxis = (XAxis, Yaxis);

- **LegendBox**
  
  LegendBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor);

- **LegendBoxSize**
  
  LegendBoxSize = (MaxWidth, MaxHeight);

- **LegendItems**
  
  LegendItems = ("Label1", Color1, SymType1, SymSize1, SymStyle1, LineType1, LineWidth1, LineColor1, Patternfill, Patternforeground, Patternbackground, Image, Shadowwidth);

- **LegendItemBox**
  
  LegendItemBox = (lineType, lineWidth, lineColor);

- **LegendLayout**
  
  LegendLayout = (Type, Location, X, Y, Justify, Columns, ItemOrder);

- **LegendSymbolPostion**
  
  LegendSymbolPostion = LEFT | RIGHT;

---

**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**

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

**BottomAxisTitleBox**

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

**DwellBox**

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

**FooterBox**

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

**HeaderBox**

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

**LeftAxisTitleBox**

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

**LeftTitleBox**

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

**LegendBox**

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

**NoteBox**

```
NoteBox = (Color1, BorderType1, BorderWidth1, "ImageURL1", ImageFormat1, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor1), ...;
```

**PieBackgrounds**

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

**RightAxisTitleBox**

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

**RightTitleBox**

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

**TopAxisTitleBox**

```
TopAxisTitleBox = (Color, BorderType, BorderWidth, "ImageURL", ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor);
```
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 NFParamScript 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] = ("Label1", "URL1", "Target1"), ...;

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
Label    URL    Target

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 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 sets are ignored. If too few are defined, the remaining data values will use the default dwell label generated.
### Common CDL Attributes

#### X, Y, and Z

#### Used in These Charts

All

**Example:**

```python
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**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

### 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:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Target</th>
<th>URL</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
</table>

### 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:**

```python
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
Common CDL Attributes

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|ON|DRAW|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;
Common CDL Attributes

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:**

```plaintext
AxesLayoutDirection = NORMAL;
```

**Values**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL</td>
<td>Draws a 2D bar</td>
</tr>
<tr>
<td>OUTWARD</td>
<td></td>
</tr>
</tbody>
</table>

**AxesSizes**

```plaintext
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 >= 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 >= 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 >= 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 >= 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:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottomsize</td>
<td></td>
</tr>
<tr>
<td>topsize</td>
<td></td>
</tr>
<tr>
<td>leftsize</td>
<td></td>
</tr>
<tr>
<td>rightsize</td>
<td></td>
</tr>
</tbody>
</table>
**AxisThickness**

\[ \text{AxisThickness} = \text{thickness}; \]

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:**

\[ \text{AxisThickness} = 15; \]

**Attributes**

- **Thickness**

**AxisTitle**

\[
\begin{align*}
\text{TopAxisTitle}[N] &= (\text{Label}, \text{Color}, \text{FontName}, \text{FontSize}, \text{Angle}, \text{interiorAlignment}); \\
\text{BottomAxisTitle}[N] &= (\text{Label}, \text{Color}, \text{FontName}, \text{FontSize}, \text{Angle}, \text{interiorAlignment}); \\
\text{LeftAxisTitle}[N] &= (\text{Label}, \text{Color}, \text{FontName}, \text{FontSize}, \text{Angle}, \text{interiorAlignment}); \\
\text{RightAxisTitle}[N] &= (\text{Label}, \text{Color}, \text{FontName}, \text{FontSize}, \text{Angle}, \text{interiorAlignment});
\end{align*}
\]

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.

- **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**

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

**Example:**

\[ \text{TopAxisTitle} = ("\text{Ceres Prototype Project Schedule}\n", \text{black}, "\text{Helvetica}", 12, \text{LEFT}); \]

**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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Target</th>
<th>URL</th>
</tr>
</thead>
</table>

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**

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
</tr>
</thead>
</table>

AxisZoom

```
BottomZoom[N] = ON|OFF;
LeftZoom[N] = ON|OFF;
RightZoom[N] = ON|OFF;
TopZoom[N] = ON|OFF;
```
**Common CDL Attributes**

**XXCornerStyle**

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**

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.

**BackgroundActiveLabel**

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

BackgroundActiveLabel defines a single active label content and destination for the background of a chart.
Common CDL Attributes

**Used in These Charts**
All

**Example:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**BackgroundFillPattern**

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

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

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

<table>
<thead>
<tr>
<th>Gradients</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Images</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td>use an image specified in the optional imageURL element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color1</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Foreground color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Starting color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color 2</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Background color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Stopping color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

| imageURL                  | The URL to an image to use as the fill                 |
Common CDL Attributes

Used in These Charts
All

Example:
BackgroundFillPattern = (GRADIENTVERTICAL, blue, white);

Attributes
Type           Color1           Color2
ImageFormat    ImageURL

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**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROW</td>
<td>The bars grow from the origin of the axis to their actual values.</td>
</tr>
<tr>
<td>FADE</td>
<td>The bars fade in.</td>
</tr>
<tr>
<td>NONE</td>
<td>The bars are immediately visible.</td>
</tr>
</tbody>
</table>

**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.
**Common CDL Attributes**

**LineType**

*Used in These Charts*
Bar, Combo, Pareto, Stock

*Example:*

\[
\text{BarBorder} = (\text{DASHED}, 2, \text{DarkGray});
\]

**Attributes**

<table>
<thead>
<tr>
<th>LineType</th>
<th>LineWidth</th>
<th>Color</th>
</tr>
</thead>
</table>

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

*Example:*

\[
\text{BarBorder} = (\text{DASHED}, 2, \text{DarkGray});
\]

**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:*

\[
\text{BarBorder} = (\text{DASHED}, 2, \text{DarkGray});
\]

**Values**

- 1 or greater Whole number width in pixels

**Default**

1

**BarColorTable[n]**

\[
\text{BarColorTable}[1-50] = \text{Color1, Color2, Color3, Color4, Color5, ...};
\]

**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)
Common CDL Attributes

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

**Attributes**

<table>
<thead>
<tr>
<th>TopLeft</th>
<th>TopRight</th>
<th>BottomRight</th>
<th>BottomLeft</th>
</tr>
</thead>
</table>

**TopLeft**

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

**Example:**
```plaintext
BarCorners = (SNIP,SQUARE,SQUARE,SQUARE);
```

**Values**

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

**Default**
SQUARE

**TopRight**

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

**Example:**
```plaintext
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:**
```plaintext
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", "-.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.

<table>
<thead>
<tr>
<th>Type</th>
<th>NONE</th>
<th>no pattern, do default fill, if any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In Patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
<td></td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
<td></td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
<td></td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
<td></td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
<td></td>
</tr>
<tr>
<td>Gradients</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTBTDIAG</td>
<td>top left to bottom right style gradient</td>
<td></td>
</tr>
</tbody>
</table>
Common CDL Attributes

<table>
<thead>
<tr>
<th>Size</th>
<th>GradientRadial</th>
<th>radial style gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
<td></td>
</tr>
</tbody>
</table>

Images

<table>
<thead>
<tr>
<th>Images</th>
<th>IMAGE</th>
<th>use an image specified in the optional imageURL element</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>color1</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Foreground color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Starting color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color2</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Background color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Stopping color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

| imageUrl     | The URL to an image to use as the fill   |

Used in These Charts
Bar, Combo, Pareto

Example:

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

Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td></td>
<td>left to right style gradient</td>
</tr>
<tr>
<td></td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td></td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td></td>
<td>radial style gradient</td>
</tr>
<tr>
<td></td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td></td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>

| start    | This color is used in the following ways:     |
|          | - Starting color for gradients                |

| stop     | This color is used in the following ways:     |
|          | - Stopping color for gradients                |

**Used in These Charts**
Bar, Combo, Pareto, Stock

**Example:**
BarHighlights = (GRADIENTRADIAL,yellow,white,15,15,15,15,-1,-1);

**Attributes**

<table>
<thead>
<tr>
<th>type</th>
<th>start</th>
<th>stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>right</td>
<td>left</td>
</tr>
<tr>
<td>width</td>
<td>height</td>
<td></td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Built-In Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
</tbody>
</table>
### Gradients

<table>
<thead>
<tr>
<th>Gradient Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>GRADIENTF DIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>

### Images

<table>
<thead>
<tr>
<th>Image Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td>use an image specified in the optional imageURL element</td>
</tr>
</tbody>
</table>

**color1**

This color is used in the following ways:
- **Foreground color for patterns**
- **Starting color for gradients**
- **Ignored in images**

**color 2**

This color is used in the following ways:
- **Background color for patterns**
- **Stopping color for gradients**
- **Ignored in images**

<table>
<thead>
<tr>
<th>imageUrl</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The URL to an image to use as the fill</td>
<td></td>
</tr>
</tbody>
</table>

### Used in These Charts

Bar, Combo, Pareto, Stock

### Example:

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

### Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>imageUrl</td>
<td></td>
</tr>
</tbody>
</table>

### BarRightFillPattern[n]P[m]

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]`.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
</tr>
</tbody>
</table>

**Built-In Patterns**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
</tr>
</tbody>
</table>
Common CDL Attributes

<table>
<thead>
<tr>
<th>Size</th>
<th>DGRID</th>
<th>diagonal grid lines, (front and back slash lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td></td>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td></td>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
<tr>
<td>Gradients</td>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
<tr>
<td>Images</td>
<td>IMAGE</td>
<td>use an image specified in the optional imageURL element</td>
</tr>
</tbody>
</table>

**color1**

This color is used in the following ways:
- Foreground color for patterns
- Starting color for gradients
- Ignored in images

**color 2**

This color is used in the following ways:
- Background color for patterns
- Stopping color for gradients
- Ignored in images

**imageURL**

The URL to an image to use as the fill

---

**Used in These Charts**

StackedBar

**Example:**

\[
\text{BarTopRightFillPattern1P2} = (\text{GRADIENTVERTICAL,blue,white}), \ldots;
\]

**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

**BarSpotlights**

\[
\text{BarSpotlights} = (\text{start,stop,center,centeroffsetx,centeroffsety, focusoffsetx, focusoffsety, radius}), \ldots;
\]

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 \text{centeroffsetx} and \text{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, \text{centeroffsetx} and \text{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:**
```plaintext
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:**
```plaintext
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:**
```plaintext
BarBorder = (DASHED, 2, DarkGray);
```

**Center**

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

**Example:**
```plaintext
BarSpotlights = (purple_40, blue_155, LEFT, 20, 10, 120, -120, 250);
```
**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>Offsets the center point to the right of the center point of the chart.</td>
</tr>
<tr>
<td>LEFT</td>
<td>Offsets the center point to the left.</td>
</tr>
<tr>
<td>TOP</td>
<td>Offsets the center point to the top.</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Offsets the center point to the bottom.</td>
</tr>
<tr>
<td>CENTER</td>
<td>Uses the chart center point for the spotlight center point.</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Offsets the center point of the spotlight to the top right.</td>
</tr>
<tr>
<td>TOPLEFT</td>
<td>Offsets the center point of the spotlight to the top left.</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Offsets the center point of the spotlight to the bottom right.</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Offsets the center point of the spotlight to the bottom left.</td>
</tr>
</tbody>
</table>

**Default**

CENTER

**Centeroffsetx**

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

**Example:**

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

**Centeroffsety**

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

**Example:**

```plaintext
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:**

```plaintext
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:**

```plaintext
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:*

```plaintext
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:*

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

**Attributes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>Font Name</th>
<th>Width</th>
</tr>
</thead>
</table>

**BarSymbol**

BarSymbol = (BarSymbolType, BarSymbolColor)…;

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

*Used in These Charts*

Bar, Combo

*Example:*

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

**Attributes**

<table>
<thead>
<tr>
<th>BarSymbolType</th>
<th>BarSymbolColor</th>
</tr>
</thead>
</table>

**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]\[N\] = \((type, \text{color}1, \text{color}2, \text{imageURL}), ...,\);

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Built-In Patterns</th>
<th>Gradients</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
<td>GRADIENTVERTICAL</td>
<td>IMAGE use an image specified in the optional imageURL element</td>
</tr>
<tr>
<td>BSLASH</td>
<td>front slash type</td>
<td>GRADIENTHORIZONTAL</td>
<td></td>
</tr>
<tr>
<td>DGRID</td>
<td>back slash type</td>
<td>GRADIENTFDIAG</td>
<td></td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal grid lines. (front and back slash lines)</td>
<td>GRADIENTBDDIAG</td>
<td></td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
<td>GRADIENTRADIAL</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines. (horizontal and vertical lines)</td>
<td>GRADIENTCENTERHORIZONTAL</td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
<td>GRADIENTCENTERVERTICAL</td>
<td></td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTBDDIAG</td>
<td>top left to bottom right style gradient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**color1** This color is used in the following ways:
- Foreground color for patterns
- Starting color for gradients
- Ignored in images

**color 2** This color is used in the following ways:
- Background color for patterns
- Stopping color for gradients
- Ignored in images

**imageURL** 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]`.

<table>
<thead>
<tr>
<th>Type</th>
<th>Built-In Patterns</th>
<th>Gradients</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
<td>none pattern, do default fill, if any</td>
<td>none pattern, do default fill, if any</td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
<td>GRADIENTVERTICAL (bottom to top style gradient)</td>
<td>IMAGE (use an image specified in the optional imageURL element)</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
<td>GRADIENTHORIZONTAL (left to right style gradient)</td>
<td>GRADIENTCENTERHORIZONTAL (center out horizontal style gradient)</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
<td>GRADIENTFDIAG (top right to bottom left style gradient)</td>
<td>GRADIENTCENTERVERTICAL (center out vertical style gradient)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
<td>GRADIENTBDIAG (top left to bottom right style gradient)</td>
<td>GRADIENTRADIAL (radial style gradient)</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
<td></td>
<td>GRADIENTCENTERHORIZONTAL (center out horizontal style gradient)</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
<td>GRADIENTCENTERVERTICAL (center out vertical style gradient)</td>
<td>GRADIENTRADIAL (radial style gradient)</td>
</tr>
</tbody>
</table>

**color1**
This color is used in the following ways:
- Foreground color for patterns
- Starting color for gradients
- Ignored in images

**color2**
This color is used in the following ways:
- Background color for patterns
- Stopping color for gradients
- Ignored in images

**imageURL**
The URL to an image to use as the fill

**Used in These Charts**
StackedBar

**Example:**
```
BarTopFillPattern1P2 = (GRADIENTVERTICAL, blue, white), ...;
```
**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>Font Name</th>
<th>Width</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Color</th>
<th>Mode</th>
<th>Depth</th>
</tr>
</thead>
</table>

**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 = $\text{Percent1,Percent2, \ldots, PercentN}$;

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 = $\text{ON|OFF}$

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

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>BorderColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:**
```plaintext
BottomColor = xB5D5F0;
```

**Attributes**
- Color

**BottomDrawMinorTics**

**BottomDrawMinorTics** = **ON** | **OFF**;

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

**Example:**
```plaintext
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:**
```plaintext
BottomFormat = (INTEGER);
BottomFormat = (FLOAT, "$%9.2f", "\n");
BottomFormat = (DATE, "%M/%y", "1/1/00 12:00", "1M");
BottomFormat = (INTEGER, "$%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:**
```java
```

**Attributes**

- **Label**

---

**BottomMargins**

```java
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:**
```java
BottomMargins = (20, 20);
```

---

**BottomScale**

```java
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:**
```java
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:**

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

**Attributes**

- `ScrollMin`
- `ScrollMax`

---

### ScrollMin

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

**Example:**

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

**Values**

- `<MinValue`
- `Default`
  - None

### ScrollMax

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

**Example:**

```plaintext
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**

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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No effect (zero length tics are not drawn).</td>
</tr>
<tr>
<td>&gt;0</td>
<td>Whole number length in pixels</td>
</tr>
</tbody>
</table>

**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.

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

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**

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

**Example:**

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

**Values**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Show tic labels for this axis</td>
</tr>
<tr>
<td>OFF</td>
<td>Don’t show tic labels on this axis</td>
</tr>
</tbody>
</table>

**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.
Common CDL Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Used in These Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Used in These Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderRadius</td>
<td></td>
</tr>
<tr>
<td>BorderType</td>
<td></td>
</tr>
<tr>
<td>BorderWidth</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>ImageFormat</td>
<td></td>
</tr>
<tr>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

**BoxActiveLabels**

```plaintext
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:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Used in These Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
</tr>
</tbody>
</table>

**BoxFence**

```plaintext
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:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Used in These Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
</tr>
</tbody>
</table>

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

**Example:**

```plaintext
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.

<table>
<thead>
<tr>
<th>Type</th>
<th>NONE</th>
<th>no pattern, do default fill, if any</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In Patterns</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
<td></td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
<td></td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
<td></td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
<td></td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
<td></td>
</tr>
<tr>
<td><strong>Gradients</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
<td></td>
</tr>
<tr>
<td><strong>Images</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td>use an image specified in the optional imageURL element</td>
<td></td>
</tr>
</tbody>
</table>

- **color1**: This color is used in the following ways:
  - Foreground color for patterns
  - Starting color for gradients
  - Ignored in images

- **color 2**: This color is used in the following ways:
  - Background color for patterns
  - Stopping color for gradients
  - Ignored in images

- **imageURL**: The URL to an image to use as the fill
Common CDL Attributes

**Height**

*Used in These Charts*

Box

*Example:*

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

**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

**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
Common CDL Attributes

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);

BoxLimitLineStyle 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** 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
</tr>
<tr>
<td><strong>Built-In Patterns</strong></td>
<td></td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
<tr>
<td><strong>Gradients</strong></td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>
Common CDL Attributes

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

**Used in These Charts**

Bubble

**Example:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
</tr>
</tbody>
</table>

**BubbleScale**

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

**Attributes**

<table>
<thead>
<tr>
<th>MinValue</th>
<th>MaxValue</th>
<th>AREA or DIAMETER</th>
<th>PointColor</th>
</tr>
</thead>
</table>

**MinValue**

**Example:**

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

**Entity**

**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);
```

**Entity**

**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);
```
**Common CDL Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual attribute</td>
<td>See the common Color attribute in Chapter 4 for details.</td>
</tr>
</tbody>
</table>

**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.
Common CDL Attributes

SymbolColor

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
SymType         MaxSize         Style     BorderColor     BorderWidth
ShadowThickness SymbolColor

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);
### Common CDL Attributes

**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 NFParamScript 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:

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

Type

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

Example:

```xml
<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:**

```xml
<PARAM NAME=Layout VALUE="(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
**Common CDL Attributes**

**ChartName**

Example:

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

**Attributes**

**Name**

**ChartScript[n]**

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

The `ChartScript` parameter takes the place of the standard `NFParamScript` 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;
Common CDL Attributes

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, xBFCA0, 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.
**Common CDL Attributes**

**Size**

**Values**
- NORMAL
- FIT
- BOTH

**Default**
- NORMAL

**CumulativeLineSymbol**

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

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

**Used in These Charts**

Pareto

**Example:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Size</th>
<th>BorderColor</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>SymbolColor</th>
<th>ShadowWidth</th>
</tr>
</thead>
</table>

**Size**

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

**Example:**

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

**Values**

Any integer value in pixels

**Default**

None

**SymbolColor**

SymbolColor specifies forground color of the CumulativeLineSymbol.

**ShadowThickness**

ShadowThickness specifies size of the shadow behind a CumulativeLineSymbol. 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.
**Common CDL Attributes**

**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:**

```
CumulativeLineSymbol = (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 cumulative line color and the outline is drawn using theBorderColor

**Default**

None

**Type**

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

**Example:**

```
CumulativeLineSymbol = (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 ImageURL 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**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>Font Name</th>
<th>Width</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Color</th>
<th>Mode</th>
<th>Depth</th>
</tr>
</thead>
</table>

**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;
```
Common CDL Attributes

<table>
<thead>
<tr>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>Displayed at the top left point</td>
</tr>
<tr>
<td>TOP</td>
<td>Displayed at the top of the point</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Displayed at the top right point</td>
</tr>
<tr>
<td>LEFT</td>
<td>Displayed at the left of the point</td>
</tr>
<tr>
<td>CENTER</td>
<td>Displayed at the center point</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Displayed at the right of the point</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Displayed at the bottom left point</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Displayed at the bottom of the point</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Displayed at the bottom right point</td>
</tr>
</tbody>
</table>

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:*
  DataLegendFridGrid = (green, white, black, null, CENTER);

*Attributes*
LineColor  
BackgroundColor  
BorderColor  
BackgroundColor

**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
**Common CDL Attributes**

---

**LineType**

Tells where to draw the lines specified with a `DataLegendGrid` 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 `DataLegendGrid` 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 = \textit{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.

\textit{Used in These Charts}

- Box Chart

\textit{Example:}

\texttt{DataPointColor = red;}

\textit{Attributes}

Color

**DataPointJitter**

DataPointJitter = \textit{ON} | \textit{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.

\textit{Used in These Charts}

- Box Chart

\textit{Example:}

\texttt{DataPointJitter = ON;}
\texttt{DataPointJitter = OFF;}

\textit{Default}

\texttt{ON}

\textit{Attributes}

Mode

**DataPointSymbol**

DataPointSymbol = \texttt{(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 |
**Common CDL Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sizeN</td>
<td>the size in pixels of the symbols for points in data series N</td>
</tr>
<tr>
<td>styleN</td>
<td>the drawing style for points in data series N. Legal values are Filled, Outlined or BOTH</td>
</tr>
<tr>
<td>borderColorN</td>
<td>the color of the border for points in data series N</td>
</tr>
<tr>
<td>borderWidthN</td>
<td>the width in pixels of the border for points in data series N</td>
</tr>
<tr>
<td>imageN</td>
<td>the image to use for displaying for points in data series N</td>
</tr>
<tr>
<td>colorN</td>
<td>the color for points in data series N</td>
</tr>
</tbody>
</table>

**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, ...;
```

**DataSet[n]** 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", x3ffcc), ("Donuts", x999ff), ("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]**

\[ \text{DataSet[1-50]P[1-50]} = a, b, c, \ldots; \]

\text{DataSet} 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:

\begin{verbatim}
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;
\end{verbatim}

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

\text{DataSet1P1} defines two values - the first value in the first set at each tic mark. \text{DataSet2P3} defines two values - the third value of the second set at each tic.

**DataSets**

\[ \text{DataSets[N]} = (\text{DataSetName1}, \text{BorderType1}, \text{BorderWidth1}, \"\text{ImageURL}\", \text{ImageFormat1}, \text{BorderColor1}), \ldots; \]

\text{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 \text{Name} attributes will be used as items in the legend; and the \text{Color} attribute will be used for each bar in the data set. The optional \text{Type} attribute indicates the shape of bar to use. \text{DataSets} must be paired with the corresponding \text{DataSet} parameter(s); you may specify from 1 up to 50 data sets.

**Used in These Charts**

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

**Example:**

\begin{verbatim}
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;
\end{verbatim}

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>Type</th>
</tr>
</thead>
</table>

**Type**

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

Example:

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

Values

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR</td>
<td>A standard rectangular 3-D bar is shown</td>
</tr>
<tr>
<td>CYLINDER</td>
<td>The bars are 3-D, and the cross-section of the bars is circulars</td>
</tr>
<tr>
<td>DIAMONDBAR</td>
<td>The bars are 3-D, and the cross-section of the bars is a diamond shape</td>
</tr>
<tr>
<td>PIEHORIZONTAL</td>
<td>The bars are a horizontal pie shape</td>
</tr>
<tr>
<td>PIEVERTICAL</td>
<td>The bars are a vertical pie shape</td>
</tr>
<tr>
<td>TRIANGLEBAR</td>
<td>The bars are 3-D, and the cross-section of the bars is triangular.</td>
</tr>
</tbody>
</table>

Default

BAR

DataType

```
DataType[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:

```
DataType = RAW;
DataType = SUMMARY;
```

Attributes

Type

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

Example:

```
DataType = RAW;
DataType = SUMMARY;
```

DataSets = ("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: 25th percentile
- Value 2: 50th percentile
- Value 3: 75th 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT</td>
<td>Agent status and errors.</td>
</tr>
<tr>
<td>ALL</td>
<td>No filtering, ALL debug information.</td>
</tr>
<tr>
<td>AXIS</td>
<td>Axis generation and rendering information.</td>
</tr>
<tr>
<td>BEANS</td>
<td>Loadable data object information.</td>
</tr>
<tr>
<td>CACHE</td>
<td>Internal memory cache status and error messages.</td>
</tr>
<tr>
<td>DWELL</td>
<td>Any issues relating to dwell (popup/active) labels.</td>
</tr>
</tbody>
</table>
### Common CDL Attributes

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

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)
```

**Used in These Charts**

All
Example:
```
DebugSet = FILE;
```

Attributes
dbgFilter

**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);
```
**Common CDL Attributes**

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>The border doesn’t connect its ends</td>
</tr>
<tr>
<td>CENTER</td>
<td>Includes the (circular) center of the dial in the borders</td>
</tr>
<tr>
<td>ENDTOEND</td>
<td>Only includes the points in the border.</td>
</tr>
</tbody>
</table>

**Default**

No defaults

**DialClip**

\[ \text{DialClip} = \text{clipType}; \]

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:**

\[ \text{DialClip} = \text{TOP}; \]

DialClip values can be:

<table>
<thead>
<tr>
<th>TOPLEFT</th>
<th>TOP</th>
<th>TOPRIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT</td>
<td>NONE</td>
<td>RIGHT</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>BOTTOM</td>
<td>BOTTOMRIGHT</td>
</tr>
</tbody>
</table>

**DialClipPad**

\[ \text{DialClipPad} = \text{N}; \]

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

**Used in These Charts**

Dial

**Example:**

\[ \text{DialClipPad} = 10; \]

**DialDelete**

\[ \text{DialDelete} = (\text{Name|ALL}), \ldots; \]

DialDelete is used to delete a specific dial, or all dials, in a dial chart.
**Used in These Charts**

Dial

**Example:**

```plaintext
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:**

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

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A string that names a dial</td>
</tr>
<tr>
<td>ALL</td>
<td>All the dials</td>
</tr>
</tbody>
</table>

**Default**

No defaults

**DialFillPattern**

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

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Built-In Patterns</th>
<th>Gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>
### Images

<table>
<thead>
<tr>
<th>IMAGES</th>
<th>use an image specified in the optional imageURL element</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>color1</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Foreground color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Starting color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color 2</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Background color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Stopping color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

| imageURL | The URL to an image to use as the fill |

### Used in These Charts

Dial

**Example:**

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

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### DialFills

```plaintext
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:**

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

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Color</th>
<th>(Switch)</th>
</tr>
</thead>
</table>

### Switch

This switch controls the dial border’s polygon behavior.

**Example:**

```plaintext
DialFills = ("Hour Dial", xebf0f3, CENTER);
```
### Common CDL Attributes

#### Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>The border doesn’t connect its ends</td>
</tr>
<tr>
<td>CENTER</td>
<td>Includes the (circular) center of the dial in the borders</td>
</tr>
<tr>
<td>ENDTOEND</td>
<td>Only includes the points in the border.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Name</th>
<th>(Switch)</th>
<th>formatExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>INTEGER</td>
<td>%d</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>INTEGER</td>
<td>$,d</td>
<td>$1,000</td>
</tr>
<tr>
<td>1000</td>
<td>INTEGER</td>
<td>%d%</td>
<td>1000%</td>
</tr>
<tr>
<td>1000</td>
<td>FLOAT</td>
<td>%f</td>
<td>1000.0</td>
</tr>
<tr>
<td>1000</td>
<td>FLOAT</td>
<td>%.2f</td>
<td>1000.00</td>
</tr>
<tr>
<td>1000</td>
<td>FLOAT</td>
<td>$,.2f</td>
<td>$1,000.00</td>
</tr>
</tbody>
</table>

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* 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>StartAngle</th>
<th>StopAngle</th>
<th>RadiusPercentage</th>
<th>(Switch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StartAngle</td>
<td>StopAngle</td>
<td>RadiusPercentage</td>
<td>(Switch)</td>
</tr>
</tbody>
</table>
**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.
**Common CDL Attributes**

**Switch**

This switch controls where the tics fall on the dial's face.

**Example:**

```plaintext
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:**

```plaintext
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**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No tics are shown</td>
</tr>
<tr>
<td>INSIDE</td>
<td>Tics appear within the dial’s face</td>
</tr>
<tr>
<td>OUTSIDE</td>
<td>Tics appear outside (but adjacent to) the dial’s face</td>
</tr>
</tbody>
</table>

**Default**

No defaults

**DialScale**

```plaintext
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.
**Common CDL Attributes**

**MinValue**

MinValue sets the absolute lower visible limit for the dial scale.

**Example:**

```plaintext
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:**

```plaintext
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:**

```plaintext
DialScale = ("Internal Pressure", 0, 10000, 1000), ("Atmospheric Pressure", 10, 100, 1), ("Pressure Change", 0, 1000, 100);
```
Common CDL Attributes

<table>
<thead>
<tr>
<th>Style</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Any real number between MinValue and MaxValue</td>
</tr>
</tbody>
</table>

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 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.
**Common CDL Attributes**

**Switch**

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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
</table>

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**

Name (Switch) LabelPos Color FontName
FontSize Angle interiorAlignment

**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
**DrawFences**

\[ \text{DrawFences} = \text{ON} \mid \text{OFF} ; \]

The `DrawFences` parameter specifies whether or not to draw fences on the chart. The default is ON.

**Used in These Charts**

Box

**Example:**

\[ \text{DrawFences} = \text{OFF} ; \]

**Attributes**

Mode

**DrawOrder**

\[ \text{DrawOrder}[N] = (\text{SYMBOL}, \ldots) ; \]

The `DrawOrder` parameter defines the order in which Bar, Line, LineFill and Stock options will be drawn.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR</td>
<td>refers to ALL bar sets</td>
</tr>
<tr>
<td>LINE</td>
<td>refers to line sets which do NOT have a fill</td>
</tr>
<tr>
<td>LINEFILL</td>
<td>refers to line sets which DO have a fill</td>
</tr>
<tr>
<td>STOCK</td>
<td>refers to ALL stock sets</td>
</tr>
</tbody>
</table>

**Used in These Charts**

Combo, Pareto, Stock

**Example:**

\[ \text{DrawOrder} = (\text{BAR}) ; \]

**Attributes**

Symbol

**DwellAnimationHighlightBorderStyle**

\[ \text{DwellAnimationHighlightBorderStyle} = (\text{lineType, LineWidth, lineColor}) ; \]
Defines the border style to be applied to a datapoint when `DwellAnimationStyle = HIGHLIGHT`.

**Used in These Charts**
All

**Example:**
```csharp
DwellAnimationHighlightBorderStyle = (DASHED, 1, BLACK);
```

**Attributes**

<table>
<thead>
<tr>
<th>LineType</th>
<th>LineWidth</th>
<th>LineColor</th>
</tr>
</thead>
</table>

**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:**
```csharp
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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>BorderType</td>
</tr>
<tr>
<td>ImageFormat</td>
<td>BorderWidth</td>
</tr>
<tr>
<td>BLCornerStyle</td>
<td>ImageURL</td>
</tr>
</tbody>
</table>

**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.
Common CDL Attributes

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.

| 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

All

Example:

DwellLabel = ("", black, "Courier", 12, LEFT);

Attributes

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
</tr>
</thead>
</table>

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 |
Common CDL Attributes

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:*

```plaintext
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*

<table>
<thead>
<tr>
<th>NodeStart</th>
<th>NodeEnd</th>
<th>Color</th>
<th>Direction</th>
<th>LineStyle</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineWidth</td>
<td>ArrowStyle</td>
<td>ArrowLength</td>
<td>ArrowWidth</td>
<td></td>
</tr>
</tbody>
</table>

*NodeStart*

NodeStart names the node where the edge begins.

*Example:*

```plaintext
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:**

```plaintext
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**

The direction of the arrow head(s), if any, for the edge.

**Example:**

```plaintext
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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No Arrows are shown</td>
</tr>
<tr>
<td>FROMTO</td>
<td>Arrow from NodeStart node to NodeEnd node</td>
</tr>
<tr>
<td>TOFROM</td>
<td>Arrow from NodeEnd to NodeStart node</td>
</tr>
<tr>
<td>BOTH</td>
<td>Arrow in both directions</td>
</tr>
</tbody>
</table>

**Default**
FROMTO

**ArrowStyle**

ArrowStyle, if used, determines the style of the arrowhead(s) on the edge. This attribute is optional.
Common CDL Attributes

**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**

- **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

---

**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**

Color    FillColor    LineType    LineWidth    Type

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>Also, value left unspecified: No color fills the area under the line.</td>
</tr>
<tr>
<td>Any legal color</td>
<td>Area under the line is filled. See Chapter 4 for the Color attribute.</td>
</tr>
</tbody>
</table>

**Default**

None

**Type**

The style of line to draw
Common CDL Attributes

**LineType**

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

**Values**

NONE  
SOLID  
DOTTED  
DASHED  
DOTDASH

**Default**

SOLID

**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**

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:**

EightyTwentyLineSymbol = (CIRCLE, 6, BOTH, white, 1);
**Common CDL Attributes**

**SymbolColor**

| Values                  | SymbolColor specifies for foreground color of the EightyTwentyLineSymbol. |

**ShadowThickness**

| Values                  | 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**

| Values                  | 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**

| FILLED                  | None |
| OUTLINED                | None |
| BOTH                    | None |

**Type**

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

**Example:**

- EightyTwentyLineSymbol = (SQUARE, 6, BOTH, cyan, 1);
### Common CDL Attributes

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

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDER</td>
<td>Draw fences under the box</td>
</tr>
<tr>
<td>OVER</td>
<td>Draw fences over the box</td>
</tr>
</tbody>
</table>

**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);

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
</tbody>
</table>

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:*

```plaintext
FooterActiveLabel = ("Where To Go", "demo.html", "frame1"),
```

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**FooterBox**

```plaintext
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**

<table>
<thead>
<tr>
<th>.setBorderColor</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td>TRCornerStyle</td>
<td>BRCornerStyle</td>
</tr>
<tr>
<td>BLCornerStyle</td>
<td>CornerColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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**

```plaintext
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
Common CDL Attributes

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

<table>
<thead>
<tr>
<th>FormatType</th>
<th>FormatExpr</th>
<th>TimeBase</th>
<th>TimeUnit</th>
</tr>
</thead>
</table>

FormatType specifies the type of number being displayed on the given axis.

Example:

LeftFormat = (INTEGER,"$%f",,);

Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>Axis value are shown as date and/or time values. See Appendix A: Date and Time Values for further detail.</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Axis values are shown with decimal parts.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Axis values are shown as integers, and are rounded if necessary.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>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.</td>
</tr>
</tbody>
</table>

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. Need not be quoted in prefix or suffix.
; | 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.
Common CDL Attributes

<table>
<thead>
<tr>
<th>Prefix or suffix</th>
<th>N</th>
<th>Used to quote special characters in a prefix or suffix, for example, &quot;'''&quot; formats 123 to &quot;#123&quot;. To create a single quote itself, use two in a row: &quot;# o'oclock&quot;.</th>
</tr>
</thead>
</table>

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
%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.
**Common CDL Attributes**

**Used in These Charts**
Bar, Box, Combo, Line, Stock,

**Example:**
```
GraphLayout = HORIZONTAL;
GraphLayout = VERTICAL;
```

**Attributes**

**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 3DDepth 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 NULL 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 NULL 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

GridColumn

LineColor specifies the color of the grid lines. See the common Color attribute in Chapter 4 for details.
Common CDL Attributes

**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 = (XAxis1, YAxis1),...;
```

`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
### GridBlockBackgroundColor

**GridBlockBackgroundColor**

```plaintext
GridBlockBackgroundColor = color...;
```

*GridBlockBackgroundColor* specifies the default grid block background color.

**Used in These Charts**

Heat Map

**Attributes**

**Color**

**Default**

White

### GridBlockCellColorType

**GridBlockCellColorType**

```plaintext
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**

```plaintext
GridBlockColors = (color1, color2,...,colorN);
```

If too few colors are specified for the grid blocks the color pattern repeats.
**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.

**Example**

GridBlockColorSpectrum = (black, grey, 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.

**Attributes**

Operator | value1 | value2 | color
---|---|---|---
Operator | value1 | value2 | color

**Operator**

Specifies the operator used

Values

">", "<", "\GE", "\LE", "\EQ", "\NE", \BETWEEN
**Common CDL Attributes**

- **Operator**
- **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";
```
**Common CDL Attributes**

---

**LineStyle**

- **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*

<table>
<thead>
<tr>
<th>SortType</th>
<th>SortOrder</th>
</tr>
</thead>
</table>

*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
**GridColumnValueFormat**

GridColumnValueFormat = (FormatType, "FormatExpr")

The `GridColumnValueFormat` parameter defines the format for the grid block values in the heat map.

**Used in These Charts**
Heat Map

**Example:**

GridBlockValueFormat = (DECIMAL, "%.2f");

**Attributes**

<table>
<thead>
<tr>
<th>FormatType</th>
<th>FormatString</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>Integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Decimal</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

**GridColumnValueStyle**

GridColumnValueStyle = ("mode", color, "font name", font size);

Defines styles for grid values.

**Used in These Charts**
Heat Map

**Attributes**

<table>
<thead>
<tr>
<th>mode</th>
<th>font name</th>
<th>font size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>purple</td>
<td>Arial, 12</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOTH</td>
<td>draw both horizontal and vertical lines (default)</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>draw vertical lines only</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>draw horizontal lines only</td>
</tr>
<tr>
<td>NONE</td>
<td>draw no grid lines</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLID</td>
<td>solid lines (default)</td>
</tr>
<tr>
<td>DOTTED</td>
<td>dotted lines</td>
</tr>
<tr>
<td>DASHED</td>
<td>dashed lines</td>
</tr>
<tr>
<td>DOTDASH</td>
<td>dot-dash lines</td>
</tr>
<tr>
<td>BAR</td>
<td>alternating bars instead of lines</td>
</tr>
</tbody>
</table>

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:**

```plaintext
HandActiveLabels = ("Minute", "Minute", "DialChartMin.html", "InfoFrame"),
("Hour", "Hour", "DialChartHr.html", InfoFrame),
("Second", "Second", "DialChartSec.html", InfoFrame);
```

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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:**

```plaintext
HandBorders = ("Hand1", DASHED, 2, DarkGray), ("Hand2", DASHED, 2, DarkGray);
```
**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);
### Common CDL Attributes

<table>
<thead>
<tr>
<th>LineType</th>
<th>LineWidth</th>
<th>Color</th>
</tr>
</thead>
</table>

#### LineType

LineType specifies the style of the border to be drawn on a chart’s hands.

#### Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLID</td>
<td>Draws a solid line of LineWidth thickness.</td>
</tr>
<tr>
<td>DOTTED</td>
<td>Draws a dotted line of LineWidth thickness.</td>
</tr>
<tr>
<td>DASHED</td>
<td>Draws a dashed line of LineWidth thickness.</td>
</tr>
<tr>
<td>DOTDASH</td>
<td>Draws a dot-dashed line of LineWidth thickness.</td>
</tr>
</tbody>
</table>

**Default**

SOLID

#### LineWidth

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

#### Values

<table>
<thead>
<tr>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 or greater</td>
<td>Whole number width in pixels</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Length</th>
</tr>
</thead>
</table>

**Value**

Value gives the value of the hand relative to the dial’s range.

**Example:**

```plaintext
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:**

```plaintext
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:**

```plaintext
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:**

```plaintext
HandDelete = ("Pressure"), ("PressureChange");
HandDelete = ALL;
```

**Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>All the hands</td>
</tr>
<tr>
<td>Name</td>
<td>A string that names a hand</td>
</tr>
</tbody>
</table>

**Default**

No defaults

---

**HandDrag**

```plaintext
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:**

```plaintext
HandDrag = "ON";
HandDrag = "OFF";
```

**Attributes**

(Switch)

---

**Switch**

This switch sets the on/off state.

**Example:**

```plaintext
HandDrag = "ON";
HandDrag = "OFF";
```

**Values**

<table>
<thead>
<tr>
<th>ON</th>
<th>Allows the user to drag the hands on the applet dial</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Stops the user from dragging the hands on the applet dial</td>
</tr>
</tbody>
</table>

**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 centerpoint; 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

<table>
<thead>
<tr>
<th>Color</th>
<th>Offsetx</th>
<th>offsety</th>
<th>Size</th>
</tr>
</thead>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>TipColor</th>
<th>ShaftColor</th>
<th>DialName</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>TipColor</th>
<th>ShaftColor</th>
<th>DialName</th>
</tr>
</thead>
</table>
**TipColor**

TipColor controls the color of the hand’s tip.

*Example:*

```plaintext
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:*

```plaintext
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:*

```plaintext
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.
Common CDL Attributes

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

<table>
<thead>
<tr>
<th>Name</th>
<th>(Switch)</th>
<th>LabelPos</th>
<th>Color</th>
<th>FontName</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontSize</td>
<td>Angle</td>
<td>interiorAlignment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Name (Switch) TipWidth ShaftWidth

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.
Common CDL Attributes

Used in These Charts
All

Example:

```
Header = ("Jets Per Minute", crimson, Helvetica, 12, 0, LEFT, CENTER, OFF);
```

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
<tr>
<td>extend</td>
<td>Specified background should extend entire length of chart.</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>HistogramBinType</th>
<th>HistogramBinSize</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>HistogramMinValue</th>
<th>HistogramMaxValue</th>
</tr>
</thead>
</table>
**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;
Common CDL Attributes

HistogramType

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:
```
            "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:**

```plaintext
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:**

```plaintext
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;
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);
```

---

**Common CDL Attributes**

Float
Attributes

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 = ("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;
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;
'}
```

Values

<table>
<thead>
<tr>
<th>LayoutType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWS</td>
<td>Charts will be arranged in rows</td>
</tr>
<tr>
<td>COLS</td>
<td>Charts will be arranged in columns</td>
</tr>
</tbody>
</table>

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.
**Common CDL Attributes**

**LayoutType**

---

### Used in These Charts

All

### Attributes

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

### 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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
</table>

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

### Used in These Charts

Combo

### Example:

```
LeftAxis = ("", black, "TimesRoman", 16, 0);
```

### Attributes

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

### 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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
</tbody>
</table>

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
**Common CDL Attributes**

**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.

**Example:**

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

**LeftAxisTitleActiveLabel**

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

LeftAxisTitleActiveLabel defines a single active label destination for the LeftAxisTitle parameter.

**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.

**Attributes**

- **Color**
- **BorderType**
- **BorderWidth**
- **ImageURL**
- **ImageFormat**
- **BorderColor**
- **TRCornerStyle**
- **BRCornerStyle**
- **BLCornerStyle**
- **CornerColor**

**Example:**

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

**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.
Common CDL Attributes

**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 [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](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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Location</th>
<th>Localized?</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>0</code></td>
<td>Number</td>
<td>Y</td>
<td>Digit</td>
</tr>
<tr>
<td><code>#</code></td>
<td>Number</td>
<td>Y</td>
<td>Digit, zero shows as absent</td>
</tr>
<tr>
<td><code>.</code></td>
<td>Number</td>
<td>Y</td>
<td>Decimal separator or monetary decimal separator</td>
</tr>
<tr>
<td><code>-</code></td>
<td>Number</td>
<td>Y</td>
<td>Minus sign</td>
</tr>
<tr>
<td><code>,</code></td>
<td>Number</td>
<td>Y</td>
<td>Grouping separator</td>
</tr>
<tr>
<td><code>E</code></td>
<td>Number</td>
<td>Y</td>
<td>Separates mantissa and exponent in scientific notation. <em>Need not be quoted in prefix or suffix.</em></td>
</tr>
<tr>
<td><code>;</code></td>
<td>Subpattern boundary</td>
<td>Y</td>
<td>Separates positive and negative subpatterns</td>
</tr>
<tr>
<td><code>%</code></td>
<td>Prefix or suffix</td>
<td>Y</td>
<td>Multiply by 100 and show as percentage</td>
</tr>
<tr>
<td><code>‰</code></td>
<td>Prefix or suffix</td>
<td>Y</td>
<td>Multiply by 1000 and show as per mille</td>
</tr>
<tr>
<td><code>¤</code></td>
<td>Prefix or suffix</td>
<td>N</td>
<td>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.</td>
</tr>
<tr>
<td><code>\u00A4</code></td>
<td>Prefix or suffix</td>
<td>N</td>
<td>Used to quote special characters in a prefix or suffix, for example, &quot;'##&quot; formats 123 to &quot;#123&quot;. To create a single quote itself, use two in a row: &quot;# o'clock&quot;.</td>
</tr>
</tbody>
</table>

**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:**

```plaintext
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:**

```plaintext
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**

```plaintext
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
Attributes

Label

LeftMargins

\begin{align*}
\text{LeftMargins} &= (\text{BottomSideMargin}, \text{TopSideMargin});
\end{align*}

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:

\begin{align*}
\text{LeftMargins} &= (20, 20);
\end{align*}

LeftScroll

\begin{align*}
\text{LeftScroll} &= (\text{ScrollMin}, \text{ScrollMax});
\end{align*}

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.

Example:

\begin{align*}
\text{LeftScroll} &= (0, 98);
\end{align*}

Attributes

\begin{align*}
\text{ScrollMin} & \quad \text{ScrollMax}
\end{align*}

ScrollMin

ScrollMin sets the lower visible limit for a scrollbar defined with LeftScroll.

Example:

\begin{align*}
\text{TopScroll} &= (0, 98);
\end{align*}

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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No effect (zero length tics are not drawn).</td>
</tr>
<tr>
<td>1 or greater</td>
<td>Whole number length in pixels</td>
</tr>
</tbody>
</table>

**Default**

-1
Common CDL Attributes

**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);

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
<tr>
<td>extend</td>
<td>Specified background should extend entire length of chart.</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderColor</td>
<td>Border color</td>
</tr>
<tr>
<td>BorderType</td>
<td>Border type</td>
</tr>
<tr>
<td>BorderWidth</td>
<td>Border width</td>
</tr>
<tr>
<td>Color</td>
<td>Color of the element</td>
</tr>
<tr>
<td>ImageFormat</td>
<td>Image format</td>
</tr>
<tr>
<td>ImageURL</td>
<td>Image URL</td>
</tr>
<tr>
<td>TRCornerStyle</td>
<td>Top Right corner style</td>
</tr>
<tr>
<td>BRCornerStyle</td>
<td>Bottom Right corner style</td>
</tr>
<tr>
<td>BLCornerStyle</td>
<td>Bottom Left corner style</td>
</tr>
<tr>
<td>CornerColor</td>
<td>Corner color</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interiorAlignment</td>
<td>Specifies the alignment to use in text strings that contain multiple lines.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Background color of the tic label area</td>
</tr>
<tr>
<td>rotationPoint</td>
<td>For rotated axis labels, anchor point for the rotation</td>
</tr>
</tbody>
</table>

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:**

```plaintext
LeftTics = ("OFF", black, "Helvetica", 10,LEFT,,RIGHT);
```

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Determines whether or not the tic labels are shown on that axis.</td>
</tr>
<tr>
<td>Color</td>
<td>Color of the element</td>
</tr>
<tr>
<td>FontName</td>
<td>Font of the element</td>
</tr>
<tr>
<td>FontSize</td>
<td>Size of the font</td>
</tr>
<tr>
<td>Angle</td>
<td>Angle of the text</td>
</tr>
<tr>
<td>interiorAlignment</td>
<td>Interior alignment</td>
</tr>
<tr>
<td>BackgroundColor</td>
<td>Background color of the tic label area</td>
</tr>
<tr>
<td>RotationPoint</td>
<td>Rotation point for the rotation</td>
</tr>
</tbody>
</table>

**Mode**

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

**Example:**

\[
\text{LeftTics} = ("\text{OFF}", \text{black}, "\text{Helvetica}", 10, \text{LEFT}, \text{RIGHT});
\]

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Show tic labels for this axis</td>
</tr>
<tr>
<td>OFF</td>
<td>Don’t show tic labels on this axis</td>
</tr>
</tbody>
</table>

**Default**

ON

**Legend**

Legend = ("Label", Color, "FontName", FontSize, Angle, interiorAlignment);

All Visual Mining charts, except for the Diagram chart and Multi-Chart, support the display of a legend anywhere on the chart. Legends offer a key to the data sets. The entries for the Legend parameter can be fully specified, or can be loaded automatically using the names and colors assigned to all data sets being displayed.

| 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**

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

**Example:**

Legend = ("Last Quarter", blue, Helvetica, 12, 270);

**Attributes**

Label    Color    FontName    FontSize    Angle    interiorAlignment

**LegendActiveLabels**

LegendActiveLabels = ("Label", "URL", "Target"), ...;

LegendActiveLabels define sets of active label destinations for the Legend parameter.

**Used in These Charts**

All

**Example:**

LegendActiveLabels = ("Set 1", "demo1.html", "frame1"),
("Set 2", "demo2.html", "frame1"),
("Set 3", "demo3.html", "frame1");
**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:*

```plaintext
LegendAnimationStyle = FADE;
```

*Values*

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADE</td>
<td>The legend fades in.</td>
</tr>
<tr>
<td>NONE</td>
<td>The legend is immediately visible.</td>
</tr>
</tbody>
</table>

*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:*

```plaintext
LegendAxis = (Bottom, Right);
```

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAxis</td>
<td>Specifies which side of a Legend becomes the X axis.</td>
</tr>
<tr>
<td>Yaxis</td>
<td></td>
</tr>
</tbody>
</table>

*Example:*

```plaintext
LegendAxis = (Bottom, Right);
```
Common CDL Attributes

**Values**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.</td>
</tr>
<tr>
<td>Pixel</td>
<td>Raw pixel coordinates</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Map values with the bottom axis of the chart</td>
</tr>
<tr>
<td>TOP</td>
<td>Map values with the top axis of the chart</td>
</tr>
</tbody>
</table>

**Default**

PERCENT

**YAxis**

Specifies which side of a Legend becomes the Y axis.

**Example:**

```plaintext
LegendAxis = (Bottom, Right);
```

**Values**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.</td>
</tr>
<tr>
<td>Pixel</td>
<td>Raw pixel coordinates</td>
</tr>
<tr>
<td>LEFT</td>
<td>Map values with the left side of the chart</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Map values with the right side of the chart</td>
</tr>
</tbody>
</table>

**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:**

```plaintext
LegendBox = (yellow, SHADOW, 5);
```

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderColor</td>
<td></td>
</tr>
<tr>
<td>BorderType</td>
<td></td>
</tr>
<tr>
<td>BorderWidth</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>ImageFormat</td>
<td></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>LineType</th>
<th>LineWidth</th>
<th>LineColor</th>
</tr>
</thead>
</table>

**LineType**

**Values**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLID</td>
<td>A solid line is displayed.</td>
</tr>
<tr>
<td>DOTTED</td>
<td>A dotted line is displayed.</td>
</tr>
<tr>
<td>DASHED</td>
<td>A dashed line is displayed.</td>
</tr>
<tr>
<td>DOTDASH</td>
<td>Alternating dots and dashes are displayed.</td>
</tr>
</tbody>
</table>

**Default**

SOLID

**LineWidth**

**Values**

Width of the line in pixels

**Default**

1

**LineColor**

**Default**

Black

**LegendDwellAnimationHighlightFill**

LegendDwellAnimationHighlightFill = Color;
**Common CDL Attributes**

**LineColor**

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 = ("Label1", 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
Common CDL Attributes

**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),
("2nd Data Set", mintcream, DIAMOND, 5, FILLED, SOLID, 2, mintcream),
("3rd Data Set", plum, DIAMOND, 5, FILLED, SOLID, 2, plum);
```

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>SymType</th>
<th>SymSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>SymStyle</td>
<td>LineType</td>
<td>LineWidth</td>
<td>LineColor</td>
</tr>
</tbody>
</table>

**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),
```
### 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>NONE</td>
<td>No symbol is displayed.</td>
</tr>
<tr>
<td></td>
<td>CIRCLE</td>
<td>Displays circles</td>
</tr>
<tr>
<td></td>
<td>SQUARE</td>
<td>Displays squares</td>
</tr>
<tr>
<td></td>
<td>DIAMOND</td>
<td>Displays diamonds</td>
</tr>
<tr>
<td></td>
<td>CROSS</td>
<td>Displays crosses</td>
</tr>
<tr>
<td></td>
<td>TARGET</td>
<td>Displays targets (bulls-eye)</td>
</tr>
<tr>
<td></td>
<td>TRIANGLE_DOWN</td>
<td>Displays downward pointing triangles</td>
</tr>
<tr>
<td></td>
<td>TRIANGLE_UP</td>
<td>Displays upward pointing triangles</td>
</tr>
<tr>
<td>Default</td>
<td>SQUARE</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Built-In Patterns</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FSLASH</td>
<td>Front slash type</td>
</tr>
<tr>
<td></td>
<td>BSLASH</td>
<td>Back slash type</td>
</tr>
<tr>
<td></td>
<td>DGRID</td>
<td>Diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td></td>
<td>HORIZONTAL</td>
<td>Horizontal lines</td>
</tr>
<tr>
<td></td>
<td>VERTICAL</td>
<td>Vertical lines</td>
</tr>
<tr>
<td></td>
<td>GRID</td>
<td>Grid lines, (horizontal and vertical lines)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gradients</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GRADIENT_VERTICAL</td>
<td>Bottom to top style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENT_HORIZONTAL</td>
<td>Left to right style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENT_FDIAG</td>
<td>Top right to bottom left style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENT_BDIAG</td>
<td>Top left to bottom right style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENT_RADIAL</td>
<td>Radial style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTCENTER_HORIZONTAL</td>
<td>Center out horizontal style gradient</td>
</tr>
<tr>
<td></td>
<td>GRADIENTCENTER_VERTICAL</td>
<td>Center out vertical style gradient</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Images</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IMAGE</td>
<td>Use an image specified in the optional imageURL element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color1</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
</table>
|                   | This color is used in the following ways: | - Foreground color for patterns  
|                   | - Starting color for gradients | - Ignored in images |

<table>
<thead>
<tr>
<th>color2</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
</table>
|                   | This color is used in the following ways: | - Background color for patterns  
|                   | - Stopping color for gradients | - Ignored in images |

<table>
<thead>
<tr>
<th>imageURL</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The URL to an image to use as the fill</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Columns</th>
<th>Justify</th>
<th>Location</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Justify</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>TOP</td>
<td>TOPRIGHT</td>
</tr>
<tr>
<td>LEFT</td>
<td>CENTER</td>
<td>RIGHT</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>BOTTOM</td>
<td>BOTTOMRIGHT</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>Displayed at the top of the panel and is left justified</td>
</tr>
<tr>
<td>TOP</td>
<td>Displayed at the top of the panel and is center justified</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Displayed at the top of the panel and is right justified</td>
</tr>
<tr>
<td>LEFT</td>
<td>Displayed at the left of the panel and is left justified</td>
</tr>
<tr>
<td>CENTER</td>
<td>X and Y attributes used to specify legend control point.</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Displayed at the right of the panel and is right justified</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Displayed at the bottom of the panel and is left justified</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Displayed at the bottom of the panel and is center justified</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Displayed at the bottom of the panel and is right justified</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>Displayed at the top of the panel and is left justified</td>
</tr>
<tr>
<td>TOP</td>
<td>Displayed at the top of the panel and is center justified</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Displayed at the top of the panel and is right justified</td>
</tr>
<tr>
<td>LEFT</td>
<td>Displayed at the left of the panel and is left justified</td>
</tr>
<tr>
<td>CENTER</td>
<td>X and Y attributes used to specify legend control point.</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Displayed at the right of the panel and is right justified</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Displayed at the bottom of the panel and is left justified</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Displayed at the bottom of the panel and is center justified</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Displayed at the bottom of the panel and is right justified</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>HORIZONTAL</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If <code>Columns</code> is 0 or 1, a single column will be displayed.</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>If <code>Columns</code> is 0 or 1, a single row will be displayed.</td>
</tr>
</tbody>
</table>

**Default**

VERTICAL

**X**

Specifies the X position of the control point of the Legend.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```

**Values**

<table>
<thead>
<tr>
<th>Percent</th>
<th>Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel</td>
<td>Raw pixel coordinates</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Map values with the bottom axis of the chart</td>
</tr>
<tr>
<td>TOP</td>
<td>Map values with the top axis of the chart</td>
</tr>
</tbody>
</table>

**Default**

50

**Y**

Specifies the Y position of the control point of the Legend.

**Example:**

```
LegendLayout = (HORIZONTAL, BOTTOM, 100, 100, BOTTOMRIGHT, 2);
```
Common CDL Attributes

<table>
<thead>
<tr>
<th>Values</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>Values will be between 0 and 1.0 or 0 and 100, being the total width or height of the graph.</td>
</tr>
<tr>
<td>Pixel</td>
<td>Raw pixel coordinates</td>
</tr>
<tr>
<td>LEFT</td>
<td>Map values with the left side of the chart</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Map values with the right side of the chart</td>
</tr>
</tbody>
</table>

**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:**

```plaintext
LineAnimationStyle = 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:**

```plaintext
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, LineSymbol, 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
**Common CDL Attributes**

**Color**

Color specifies the base color of the shadow to be drawn behind the chart.

*Example:*

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

**Offsetx**

Offsetx specifies the x-coordinate offset from center.

*Example:*

```plaintext
LineDropShadow = (black, 25 -10, 25);
```

**Offsety**

Offsety specifies the y-coordinate offset from center.

*Example:*

```plaintext
LineDropShadow = (black, 25 -10, 25);
```

**Size**

Size specifies the width of the blurred area.

*Example:*

```plaintext
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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
</tr>
<tr>
<td>Built-In Patterns</td>
<td></td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
</tbody>
</table>
### Common CDL Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS\textbackslash{}</td>
<td>back slash type</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
</tbody>
</table>

### Gradients

<table>
<thead>
<tr>
<th>Gradient</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>

### Images

<table>
<thead>
<tr>
<th>Image</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td>use an image specified in the optional imageUrl element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color1</td>
<td>This color is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>Foreground color for patterns</td>
</tr>
<tr>
<td></td>
<td>Starting color for gradients</td>
</tr>
<tr>
<td></td>
<td>Ignored in images</td>
</tr>
<tr>
<td>color2</td>
<td>This color is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>Background color for patterns</td>
</tr>
<tr>
<td></td>
<td>Stopping color for gradients</td>
</tr>
<tr>
<td></td>
<td>Ignored in images</td>
</tr>
</tbody>
</table>

| imageUrl | The URL to an image to use as the fill                                      |

### Used in These Charts

**Line**

**Example:**

```
LineFillPattern = (VERTICAL, blue, white);
```

**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td></td>
<td>ImageURL</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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**

```plaintext
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:**
```plaintext
LineStyle =(SOLID, 3, blue, blue, NORMAL, OFF),
(SOLID, 2, red, red, FIT, COLOR);
```

**Attributes**

<table>
<thead>
<tr>
<th>Color</th>
<th>FillColor</th>
<th>LineType</th>
<th>LineWidth</th>
<th>Type</th>
</tr>
</thead>
</table>

**FillColor**

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

**Example:**
```plaintext
LineStyle =(SOLID, 3, skyblue, lightskyblue),
(SOLID, 3, red, pink),
(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 **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 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**

\[
\text{LineSymbol}[N] = (\text{Type}, \text{Size}, \text{Style}, \text{BorderColor}, \text{BorderWidth}, \text{ImageURL}, \text{SymbolColor}, \text{ShadowWidth}), \ldots;
\]

`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:**

```plaintext
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**

<table>
<thead>
<tr>
<th>.BorderColor</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ShadowWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Style</td>
<td>SymbolColor</td>
<td>Type</td>
</tr>
</tbody>
</table>
Common CDL Attributes

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 the foreground color of the LineSymbol.

ShadowThickness

ShadowThickness specifies the 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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILLED</td>
<td>Symbol is filled with the LineSet color.</td>
</tr>
<tr>
<td>OUTLINED</td>
<td>Only the outline is drawn, using the LineSet color</td>
</tr>
<tr>
<td>BOTH</td>
<td>Symbol is filled with the LineSet and the outline is drawn using the BorderColor</td>
</tr>
</tbody>
</table>

Default

None

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

**Example:**

```plaintext
LineSymbol = (CIRCLE, 6, BOTH, white, 1),
(SQUARE, 6, BOTH, cyan, 1),
(DIAMOND, 6, BOTH, firebrick, 1),
(CROSS, 6, BOTH, green, 1);
```

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

**Default**

None

**LineSymbolSpotlights**

```plaintext
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:**

```plaintext
LineSymbolSpotlights = (purple_40,blue_155,RIGHT,50,-50,100,150,250);
```
Attributes

<table>
<thead>
<tr>
<th>start</th>
<th>stop</th>
<th>center</th>
</tr>
</thead>
<tbody>
<tr>
<td>centeroffsetx</td>
<td>centeroffsety</td>
<td>focusoffsetx</td>
</tr>
</tbody>
</table>

**Start**

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

**Example:**

```plaintext
LineSymbolSpotlights = (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**

Specifies the x-coordinate offset for the center of the spotlight’s focus.

**Focusoffsety**

Specifies the y-coordinate offset for the center of the spotlight’s focus.

**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
Common CDL Attributes

**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*

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>Displayed at the top left point in a lineset</td>
</tr>
<tr>
<td>TOP</td>
<td>Displayed at the top of the point in a lineset</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Displayed at the top right point in a lineset</td>
</tr>
<tr>
<td>LEFT</td>
<td>Displayed at the left of the point in a lineset</td>
</tr>
<tr>
<td>CENTER</td>
<td>Displayed at the center point in a lineset</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Displayed at the right of the point in a lineset</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Displayed at the bottom left point in a lineset</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Displayed at the bottom of the point in a lineset</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Displayed at the bottom right point in a lineset</td>
</tr>
</tbody>
</table>

*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 = ("Labell", "URLl", "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:**

```plaintext
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:**

```plaintext
MeanLine = (DASHED, 3, green);
```

<table>
<thead>
<tr>
<th>type</th>
<th>type of line to draw. Legal values are SOLID, DASHED, DOTTED and DOTDASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>width in pixels of the line, the default is 1.</td>
</tr>
<tr>
<td>color</td>
<td>color of the line, the default is the MeanColor</td>
</tr>
</tbody>
</table>

**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);

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typeN</td>
<td>the type of symbol to use for the mean in data series N. Legal values are</td>
</tr>
<tr>
<td></td>
<td>NONE, CIRCLE, SQUARE, DIAMOND, CROSS, TARGET, TRIANGLEDOWN, TRIANGLEUP, IMAGE</td>
</tr>
<tr>
<td>sizeN</td>
<td>the size in pixels of the symbols for the mean in data series N</td>
</tr>
<tr>
<td>styleN</td>
<td>the drawing style for the mean in data series N. Legal values are</td>
</tr>
<tr>
<td></td>
<td>FILLED, OUTLINED or BOTH</td>
</tr>
<tr>
<td>borderColorN</td>
<td>the color of the border for the mean in data series N</td>
</tr>
<tr>
<td>borderWidthN</td>
<td>the width in pixels of the border for the mean in data series N</td>
</tr>
<tr>
<td>imageN</td>
<td>the image to use for displaying for the mean in data series N</td>
</tr>
<tr>
<td>colorN</td>
<td>the color for the mean in data series N</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
</table>
NaturalDisplayOrder

NaturalDisplayOrder = \texttt{ON} \mid \texttt{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.

\textbf{Used in These Charts}

Box Chart

\textbf{Example:}

\begin{verbatim}
NaturalDisplayOrder = ON;
NaturalDisplayOrder = OFF;
\end{verbatim}

\textbf{Default}

\texttt{ON}

\textbf{Attributes}

\texttt{Mode}

NodeBox

\begin{verbatim}
NodeBox[N] = (Color, BorderType, BorderWidth, ImageURL, ImageFormat, BorderColor, TRCornerStyle, BRCornerStyle, BLCornerStyle, CornerColor), ...
\end{verbatim}

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.

\textbf{Used in These Charts}

Diagram

\textbf{Example:}

\begin{verbatim}
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"),
\end{verbatim}
**Attributes**

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ImageFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderColor</td>
<td>TRCornerStyle</td>
<td>BRCornerStyle</td>
<td>BLCornerStyle</td>
<td>CornerColor</td>
</tr>
</tbody>
</table>

**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:**

```plaintext
NodeDrag = "ON";
NodeDrag = "OFF";
```

**Attributes**

(Switch)

**Switch**

This switch sets the on/off state.

**Example:**

```plaintext
NodeDrag = "ON";
NodeDrag = "OFF";
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Allows the user to drag the hands on the applet dial</td>
</tr>
<tr>
<td>OFF</td>
<td>Stops the user from dragging the hands on the applet dial</td>
</tr>
</tbody>
</table>

**Default**

ON

**NodeLabel**

NodeLabel[N] = (Color, FontName, FontSize, Angle), ...;
**Common CDL Attributes**

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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
</table>

**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),
Common CDL Attributes

```
("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]

\[\text{NoteActiveLabels}[1-20] = ("Label", "URL", "Target"), \ldots;\]

\text{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 "dwell" 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");
```
**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**

<table>
<thead>
<tr>
<th>LineType</th>
<th>LineWidth</th>
<th>LineColor</th>
<th>ArrowType</th>
<th>ArrowStyle</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No arrowhead is shown</td>
</tr>
<tr>
<td>FROMTO</td>
<td>Arrowhead is oriented and placed with last point on the line</td>
</tr>
<tr>
<td>TOFROM</td>
<td>Arrowhead is oriented and placed with first point on the line</td>
</tr>
<tr>
<td>BOTH</td>
<td>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</td>
</tr>
</tbody>
</table>

**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**

* XAxis
* YAxis

---

**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**

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ImageFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRCornerStyle</td>
<td>BRCornerStyle</td>
<td>BLCornerStyle</td>
<td>CornerColor</td>
<td></td>
</tr>
</tbody>
</table>

**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.
**Common CDL Attributes**

**XXCornerStyle**

**Used in These Charts**
Diagram

**Example:**

```plaintext
NoteDrag = "ON";
```

**Attributes**

**Mode**

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Allows the user to drag NoteSets</td>
</tr>
<tr>
<td>OFF</td>
<td>Stops the user from dragging NoteSets</td>
</tr>
</tbody>
</table>

**Default**

ON

**NoteLabel**

```plaintext
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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The legal values for <code>interiorAlignment</code> are LEFT, RIGHT, or CENTER.</td>
</tr>
</tbody>
</table>

**Used in These Charts**

All

**Example:**

```plaintext
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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

**NoteSet[n]**

```plaintext
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:**

```plaintext
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_{n \to a}", 38, 25);
```

**Attributes**

<table>
<thead>
<tr>
<th>Text</th>
<th>X</th>
<th>Y</th>
<th>X1</th>
<th>Y1</th>
<th>X2</th>
<th>Y2</th>
<th>X3</th>
<th>Y3</th>
</tr>
</thead>
</table>

**Text**

Text displays the text of the NoteSet. The text may contain multiple lines, delimited by the \n (new line) symbol.

**Example:**

```plaintext
NoteSet1 = ("The Amplifier clearly led all components in overall sales. Bob McIness and staff 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:**

```plaintext
NoteSet1 = ("The Amplifier clearly led all components in overall sales. Bob McIness and staff 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.
Common CDL Attributes

**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*

<table>
<thead>
<tr>
<th>Name</th>
<th>Justify</th>
</tr>
</thead>
</table>

*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);
```
**Common CDL Attributes**

**Values**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPLEFT</td>
<td>The note’s top left corner becomes its origin</td>
</tr>
<tr>
<td>TOP</td>
<td>A point horizontally centered on the note’s top edge becomes its origin</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>The note’s top right corner becomes its origin</td>
</tr>
<tr>
<td>LEFT</td>
<td>A point vertically centered on the left side becomes its origin</td>
</tr>
<tr>
<td>CENTER</td>
<td>The center of the note becomes its origin</td>
</tr>
<tr>
<td>RIGHT</td>
<td>A point vertically centered on the right side becomes its origin</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>The note’s bottom left corner becomes its origin</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>A point horizontally centered on the note’s bottom edge becomes its origin</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>The note’s bottom right corner becomes its origin</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Format</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%f</td>
<td>1234.46</td>
</tr>
<tr>
<td>%.f</td>
<td>1.234,46</td>
</tr>
<tr>
<td>%d</td>
<td>1234</td>
</tr>
</tbody>
</table>
Common CDL Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>.d</td>
<td>1.234</td>
</tr>
<tr>
<td>.8,1f</td>
<td>1.234,5</td>
</tr>
<tr>
<td>%08,1f</td>
<td>001234,5</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>typeN</th>
<th>the type of symbol to use for outliers in data series N. Legal values are NONE, CIRCLE, SQUARE, DIAMOND, CROSS, TARGET, TRIANGLEDOWN, TRIANGLEUP, IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>sizeN</td>
<td>the size in pixels of the symbols for outliers in data series N</td>
</tr>
<tr>
<td>styleN</td>
<td>the drawing style for outliers in data series N. Legal values are FILLED, OUTLINED or BOTH</td>
</tr>
<tr>
<td>borderColorN</td>
<td>the color of the border for outliers in data series N</td>
</tr>
<tr>
<td>borderWidthN</td>
<td>the width in pixels of the border for outliers in data series N</td>
</tr>
<tr>
<td>imageN</td>
<td>the image to use for displaying for outliers in data series N</td>
</tr>
<tr>
<td>colorN</td>
<td>the color for outliers in data series N</td>
</tr>
</tbody>
</table>

**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

\[ \text{PieAngle}[N] = \text{Integer}; \]

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:**

\[ \text{PieAngle} = 180; \]

**Attributes**

None

### PieAngles

\[ \text{PieAngles} = \text{Value1}, \text{Value2}...,; \]

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:**

\[ \text{PieAngles} 15,0,10,10...,; \]

**Attributes**

None

### PieBackgrounds

\[ \text{PieBackgrounds} = (\text{Color}, \text{BorderType}, \text{BorderWidth}, "\text{ImageURL}" , \text{ImageFormat}, \text{BorderColor}),...; \]

PieBackground specifies the background for each pie.

**Used in These Charts**

MultiPie

**Example:**

\[ \text{PieBackgrounds} = (\text{pink,RAISED,10,,}), (\text{lightgray,BOX,1,,}); \]

**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 centerpoint; 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

<table>
<thead>
<tr>
<th>Color</th>
<th>Offsetx</th>
<th>offsety</th>
<th>Size</th>
</tr>
</thead>
</table>

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, "-0.05","-0.05", 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**

- `start`
- `stop`
- `gap`
- `size`
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.

<table>
<thead>
<tr>
<th>Type</th>
<th>NONE</th>
<th>no pattern, do default fill, if any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradients</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTHORIZONTAL</td>
<td>left to right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTTANGLED</td>
<td>top left to bottom right style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
<td></td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Starting color for gradients</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stop</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Stopping color for gradients</td>
</tr>
</tbody>
</table>

**Used in These Charts**

Pie, Multipie

**Example:**

```
PieHighlights = (GRADIENTRADIAL,yellow,white,270,15,-1);
```
### Attributes

<table>
<thead>
<tr>
<th>type</th>
<th>start</th>
<th>stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>gap</td>
<td>extent</td>
</tr>
</tbody>
</table>

### 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**

<table>
<thead>
<tr>
<th>State</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>InteriorAlignment</th>
</tr>
</thead>
</table>

### 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**

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ImageFormat</th>
<th>BorderColor</th>
</tr>
</thead>
</table>

### 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**

<table>
<thead>
<tr>
<th>Orientation</th>
<th>Number of Rows</th>
<th>Number Of Columns</th>
</tr>
</thead>
</table>

**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 centeroffsets and centeroffset 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, centeroffsets and centeroffset set 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 focusoffset is the offset from the pie chart
center which defines the x-coordinate of the focus point of the spotlight. The element \textit{focusoffset} 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, \textit{focusoffsetx} and \textit{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 \textit{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 \textit{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.

\textit{Used in These Charts}
Pie, MultiPie

\textit{Example:}

\begin{verbatim}
PieSpotlights = (blue_0,blue_105,RIGHT,1,1,120,-120,440);
\end{verbatim}

\textbf{Attributes}

\begin{verbatim}
start stop center
centeroffsetx centeroffsety focusoffsetx focusoffsety radius
\end{verbatim}

\underline{Start}

\begin{verbatim}
Start specifies the first of the two colors which will be interpolated to produce a gradient spotlight.
\end{verbatim}

\textit{Example:}

\begin{verbatim}
PieSpotlights = (\textbf{purple} \_40,\textbf{blue} \_155,LEFT,20,10,120,-120,250);
\end{verbatim}

\underline{Stop}

\begin{verbatim}
Stop specifies the second of two colors which will be interpolated to produce a gradient spotlight.
\end{verbatim}

\textit{Example:}

\begin{verbatim}
PieSpotlights = (\textbf{purple} \_40,\textbf{blue} \_155,LEFT,20,10,120,-120,250);
\end{verbatim}

\underline{Center}

\begin{verbatim}
Center specifies the position around of the center of the chart where the spotlight center will be placed.
\end{verbatim}

\textit{Example:}

\begin{verbatim}
PieSpotlights = (\textbf{purple} \_40,\textbf{blue} \_155,\textbf{LEFT},20,10,120,-120,250);
\end{verbatim}
Common CDL Attributes

Values

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>Offsets the center point of the spotlight to the right of the center point of the chart.</td>
</tr>
<tr>
<td>LEFT</td>
<td>Offsets the center point to the left.</td>
</tr>
<tr>
<td>TOP</td>
<td>Offsets the center point to the top.</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Offsets the center point to the bottom.</td>
</tr>
<tr>
<td>CENTER</td>
<td>Uses the chart center point for the spotlight center point.</td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td>Offsets the center point of the spotlight to the top right.</td>
</tr>
<tr>
<td>TOPLEFT</td>
<td>Offsets the center point of the spotlight to the top left.</td>
</tr>
<tr>
<td>BOTTOMRIGHT</td>
<td>Offsets the center point of the spotlight to the bottom right.</td>
</tr>
<tr>
<td>BOTTOMLEFT</td>
<td>Offsets the center point of the spotlight to the bottom left.</td>
</tr>
</tbody>
</table>

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:*

```plaintext
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:*

```plaintext
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:*

```plaintext
Pie3DDepth = 12;
```

*Attributes*

None
PlotArea

PlotArea = \( (x_{location}, y_{location}, width, height) \);

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.

\[x_{location} \text{ – } \text{x coordinate of the upper left corner of the plot area within the chart. If } x_{location} \text{ is a number between 0 and 1, it is interpreted as a percentage of the total available chart width. If } x_{location} \geq 1 \text{, it is interpreted as an absolute location in pixels.}\]

\[y_{location} \text{ – } \text{y coordinate of the upper left corner of plot area within the chart. If } y_{location} \text{ is a number between 0 and 1, it is interpreted as a percentage of the total available chart height. If } y_{location} \geq 1 \text{ it is interpreted as an absolute location in pixels.}\]

\[width \text{ – width of plot area. If } width \text{ is a number between 0 and 1, it is interpreted as a percentage of the total available chart area. If } width \geq 1 \text{ it is interpreted as an absolute size in pixels.}\]

\[height \text{ – width of plot area. If } height \text{ is a number between 0 and 1, it is interpreted as a percentage of the total available chart height. If } height \geq 1 \text{ it is interpreted as an absolute size in pixels.}\]

**Used in These Charts**

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

**Example:**

\[\text{PlotArea} = (1,1,1,1);\]

**Attributes**

\[x_{location} \quad y_{location} \quad width \quad height\]

PlotType

PlotType = \( \text{Type}; \)
**PlotType** defines the type of plot to be displayed for a BoxChart.

**Used in These Charts**

Box

**Example:**

```plaintext
PlotType = STANDARD | EDA | GAUSSIAN | PERCENTN | TENNINETY;
```

**Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PlotType</strong></td>
<td>defines the type of plot to be displayed for a BoxChart.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>refers to the manner in which lines are stacked in a line chart, allowing the creation of area graph.</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
PlotType = STANDARD;
```

**Values**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>In the Standard type of box plot, the box represents the IQR. The Inter-Quartile Range is the difference between the upper hinge (the 75th percentile) and the lower hinge (the 25th percentile).</td>
</tr>
</tbody>
</table>
| 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 lower quartile, and the upper part of the box is then referred to as the upper quartile.) However, this type of plot features additional elements called fences. The fences are defined as follows: 
  - **Lower Outer Fence** = lower quartile - 3 * IQR
  - **Lower Inner Fence** = lower quartile - 1.5 * IQR
  - **Upper Inner Fence** = upper quartile + 1.5 * IQR
  - **Upper Outer Fence** = 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, Nth percentile, median, 100-Nth 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, 10th percentile, median, 90th 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
**Common CDL Attributes**

---

**Mode**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Show tic labels for this axis</td>
</tr>
<tr>
<td>OFF</td>
<td>Don’t show tic labels on this axis</td>
</tr>
</tbody>
</table>

**Default**

ON

---

**interiorAlignment**

interiorAlignment dictates the alignment of text lines in multi-line labels.

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td></td>
</tr>
<tr>
<td>CENTER</td>
<td></td>
</tr>
</tbody>
</table>

**Default**

CENTER

---

**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**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

---

**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**

<table>
<thead>
<tr>
<th>DataType</th>
<th>FormatString</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataType</strong></td>
<td>FormatString</td>
</tr>
</tbody>
</table>

**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
### Common CDL Attributes

<table>
<thead>
<tr>
<th>DataType</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PolarScale = (0,10,1);</td>
</tr>
</tbody>
</table>

#### 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:**

**Attributes**

- **minWidth**
- **minHeight**
- **maxWidth**
- **maxHeight**

### PolyActiveLabels

**PolyActiveLabels** = (“Label1”, “URL1”, “Target1”)...

PolyActiveLabels define the active labels associated with defined polygons.
**Common CDL Attributes**

<table>
<thead>
<tr>
<th>DataType</th>
<th>Used in These Charts</th>
<th>Example</th>
<th>Default</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PolyActiveLabels</strong></td>
<td>Diagram Chart</td>
<td>PolyActiveLabels = (“west region”, “javascript:doSelection()”);</td>
<td>Use tag name as label</td>
<td>Label, URL, Target</td>
</tr>
<tr>
<td><strong>PolyColor</strong></td>
<td>Diagram Chart</td>
<td>PolyColor = (&quot;Tag&quot;, &quot;Color&quot;),...;</td>
<td>None</td>
<td>TagName, Color</td>
</tr>
<tr>
<td><strong>PolySet</strong></td>
<td>Diagram Chart</td>
<td>PolySet = (“Tag”, x1,y1,x2,y2,...);...;</td>
<td>None</td>
<td>TagName, X/Y Pairs</td>
</tr>
</tbody>
</table>
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),
Common CDL Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>StepSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>RadialAxesAngles</td>
<td>angle1, angle2 ...;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RadialAxesColors</td>
<td>color1, color2 ...;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RadialAxesFormat</td>
<td>(dataType, formatString),...;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>DataType</th>
<th>FormatString</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataType</strong></td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>%d</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>$,%d</td>
<td>$1,000</td>
</tr>
<tr>
<td>1000</td>
<td>%d%</td>
<td>1000%</td>
</tr>
<tr>
<td>1000</td>
<td>%f</td>
<td>1000.0</td>
</tr>
<tr>
<td>1000</td>
<td>%.2f</td>
<td>1000.00</td>
</tr>
<tr>
<td>1000</td>
<td>$,.2f</td>
<td>$1,000.00</td>
</tr>
</tbody>
</table>

If the format type is DECIMAL, the format syntax is consistent with those defined in the Java DecimalFormat spec.

**RadialAxesLabel**

```plaintext
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:**

```plaintext
RadialAxesLabel = ("ON", black, "sansserif", 10, 0), ("ON", red, "sansserif", 12, 0);
```

**Attributes**

<table>
<thead>
<tr>
<th>Font</th>
<th>Color</th>
<th>Label</th>
<th>Mode</th>
</tr>
</thead>
</table>

**RadialAxesLabels**

```plaintext
RadialAxesLabels = ("AxisLabel1", AxisLabel2...), (AxisLabel1, AxisLabel2...) ...;
```

`RadialAxesLabels` specifies the text to use as axis tie labels for each axis.
Common CDL Attributes

FormatString

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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Target</th>
<th>URL</th>
</tr>
</thead>
</table>

### 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**

<table>
<thead>
<tr>
<th>GridRadius</th>
<th>GridLineType</th>
<th>GridLineWidth</th>
<th>GridLineColor</th>
<th>GridAreaColor</th>
</tr>
</thead>
</table>
**RelativeBoxSymbolWidth**

RelativeBoxSymbolWidth = OFF | LINEAR | SQRT;

RelativeBoxSymbolWidth instructs the Box Chart to assign widths to the boxes in the chart based on a function of the relative number of points in each data series. For example, if a value of LINEAR is used, a data series with 100 data points will have a box width twice that of a box representing a data series with only 50 data points. If a value of SQRT is used, the relative box sizes will be calculated as a function of the square root of the number of points in each series.

**Used in These Charts**
- Box Chart

**Example:**
RelativeBoxSymbolWidth = SQRT;

**Default**
OFF

**Attributes**
- Mode

---

**RightActiveLabels**

RightActiveLabels[N] = ("Label", "URL", "Target");

The right axis labels become active labels when RightActiveLabels parameter is used. Each set in parenthesis has a corresponding set within a Dataset parameter.

**Used in These Charts**
- All

**Attributes**
- Label
- URL
- Target

---

**RightAxis**

RightAxis[N] = (Label, Color, FontName, FontSize, Angle, interiorAlignment);

If TopAxis is defined for a Combo chart, then the right axis will be used to map the Y data values for all line sets, unless otherwise specified using the LineAxis parameter.

**interiorAlignment**
- Specifies the alignment to use in text strings that contain multiple lines.

The legal values for interiorAlignment are LEFT, RIGHT, or CENTER.
**Common CDL Attributes**

**FormatStr**

**Used in These Charts**

Combo

**Example:**

```
RightAxis = ("Milliseconds", black, "TimesRoman", 16, 0);
```

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

**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 Head 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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
<th>exteriorAlignment</th>
</tr>
</thead>
</table>

**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");
```
### Attributes

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

### RightAxisTitleBox

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

The `RightAxisTitleBox` 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:

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

### Attributes

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ImageFormat</th>
<th>BorderColor</th>
<th>TRCornerStyle</th>
<th>BRCornerStyle</th>
<th>BLCornerStyle</th>
<th>CornerColor</th>
</tr>
</thead>
</table>

### 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.

### RightColor

```
RightColor = Color;
```

`RightColor` 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:

```
RightColor = xB5D5F0;
```

### Attributes

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
</table>

### RightDrawMinorTics

```
RightDrawMinorTics = ON|OFF;
```

`RightDrawMinorTics` controls whether or not right tics are drawn. The default value is ON.
Common CDL Attributes

**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**

<table>
<thead>
<tr>
<th>FormatType</th>
<th>FormatExpr</th>
<th>TimeBase</th>
<th>TimeUnit</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>DATE</th>
<th>Axis value are shown as date and/or time values. See Appendix A: Date and Time Values for further detail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOAT</td>
<td>Axis values are shown with decimal parts.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Axis values are shown as integers, and are rounded if necessary.</td>
</tr>
</tbody>
</table>

**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.
Common CDL Attributes

**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**

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:**
```
"Steve", "Paul";
```

**Attributes**
Label
**RightMargins**

\[ \text{RightMargins} = (\text{BottomSideMargin}, \text{TopSideMargin}) \]

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:*
\[ \text{RightMargins} = (20, 20) \]

**RightScroll**

\[ \text{RightScroll} = (\text{ScrollMin}, \text{ScrollMax}) \]

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.

*Example:*
\[ \text{RightScroll} = (0, 98) \]

**Attributes**

*ScrollMin* \hspace{1cm} *ScrollMax*

**ScrollMin**

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

*Example:*
\[ \text{RightScroll} = (0, 98) \]

**Values**

<MinValue

**Default**

None

**ScrollMax**

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

*Example:*
\[ \text{RightScroll} = (0, 98) \]
**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**
  - 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:**

```plaintext
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.
Common CDL Attributes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
<th>BackgroundColor</th>
<th>RotationPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>interiorAlignment</td>
<td>Specifies the alignment to use in text strings that contain multiple lines.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Background color of the tic label area</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotationPoint</td>
<td>For rotated axis labels, anchor point for the rotation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:**
```javascript
RightTics = ("OFF", black, "Helvetica", 10);
```

**Attributes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
<th>BackgroundColor</th>
<th>RotationPoint</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
<th>BackgroundColor</th>
<th>RotationPoint</th>
</tr>
</thead>
</table>

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

**Example:**
```javascript
RightTics = ("OFF", black, "Helvetica", 10);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Show tic labels for this axis</td>
</tr>
<tr>
<td>OFF</td>
<td>Don’t show tic labels on this axis</td>
</tr>
</tbody>
</table>

**Default**

ON

**RightTitle**

```javascript
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:**
```javascript
RightTitle = ("Financial Status", royalblue, Helvetica, 14, 0, LEFT);
```
**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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th><em>interiorAlignment</em></th>
<th><em>exteriorAlignment</em></th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>BorderColor</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td>TRCornerStyle</td>
<td>BRCornerStyle</td>
</tr>
<tr>
<td>BLCornerStyle</td>
<td>CornerColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.
Common CDL Attributes

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

<table>
<thead>
<tr>
<th>LINE_TYPE</th>
<th>LINE_WIDTH</th>
<th>LINE_COLOR</th>
</tr>
</thead>
</table>

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
**Common CDL Attributes**

### SashMode

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
RubberBandFill = YELLOW_100);
```

**Sash**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Width</th>
<th>SashMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
Sash[N] = (Color, Width, SashMode);
```

By default, all charts within a multi-chart are allocated a fixed area within the applet based on the Charts and Layout parameters, with no dividing “sashes.” The Sash parameter allows the specification of the color and width of a dividing line between pair of charts similar to that seen in HTML frames.

**Used in These Charts**

Multi-chart

**Example:**

```plaintext
<PARAM NAME=Sash VALUE="(xe3e3e3, 4, DYNAMIC)">
```

**SashMode**

In some situations, it may be desired to allow users to change the relative size of each chart interactively. SashMode switches the ability to move the sashes in Multi-charts.

**Example:**

```plaintext
<PARAM NAME=Sash VALUE="(xe3e3e3, 4, DYNAMIC)">
```

**Values**

- **DYNAMIC** Viewer will be allowed to move each sash line in order to change the amount of space allocated to each row/column or chart
- **STATIC** Viewer will not be able to move the sash lines

**Default**

STATIC

**Scale**

```plaintext
TopScale[N] = mode, ...;
BottomScale[N] = mode, ...;
LeftScale[N] = mode, ...;
```
RightScale[N] = \textit{mode}, \ldots;

The \textit{Scale} parameter specifies the minimum and maximum data values which will be displayed along the axis. If the \textit{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.

\textbf{ScaleFactor}

\texttt{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).

\textbf{Attributes}

\texttt{Number}

\textbf{ScaleMode}

\texttt{TopScaleMode[N] = \textit{(mode,logBase)}, \textit{(mode,logBase)}, \ldots;}
\texttt{BottomScaleMode[N] = \textit{(mode, logBase)}, \textit{(mode,logBase)}, \ldots;}
\texttt{LeftScaleMode[N] = \textit{(mode, logBase)}, \textit{(mode, logBase)}, \ldots;}
\texttt{RightScaleMode[N] = \textit{(mode,logBase)}, \textit{(mode, logBase)}, \ldots;}

The \textit{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.

\textbf{Example:}

\texttt{TopScaleMode = \textit{(LOG,16)};
Common CDL Attributes

**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.

**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**

Name    Label    URL    Target
**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>LineWidth</th>
<th>Color</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>StartValue</th>
<th>StopValue</th>
</tr>
</thead>
</table>
**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A string that names a hand</td>
</tr>
<tr>
<td>ALL</td>
<td>All the hands</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Allows the user to drag the sector around the dial</td>
</tr>
<tr>
<td>OFF</td>
<td>Stops the user from dragging the sector around the dial</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Built-In Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>no pattern, do default fill, if any</td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>GRADIENTHORIZTONTAL</td>
<td>left to right style gradient</td>
</tr>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONAT</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
<tr>
<td>IMAGE</td>
<td>use an image specified in the optional imageURL element</td>
</tr>
</tbody>
</table>

**Used in These Charts**

**Dial**

**Example:**

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

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Color1</th>
<th>Color2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageFormat</td>
<td>ImageURL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The legal values for <code>interiorAlignment</code> are LEFT, RIGHT, or CENTER.</td>
</tr>
</tbody>
</table>

**Used in These Charts**

**Dial**
**Example:**

```plaintext
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>(Switch)</th>
<th>LabelPos</th>
<th>Color</th>
<th>FontName</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontSize</td>
<td>(Switch)</td>
<td>LabelPos</td>
<td>Color</td>
<td>FontName</td>
</tr>
<tr>
<td>interiorAlignment</td>
<td>Angle</td>
<td></td>
<td></td>
<td>FontName</td>
</tr>
</tbody>
</table>

**Switch**

This switch turns the tic labels’ visibility on and off.

**Example:**

```plaintext
DialTicLabelStyles = ("Hour Dial","ON",1.1,black,"Helvetica",14,0),
("Minute Dial","ON",1.1,black,"Helvetica",14,0);
```

**Values**

<table>
<thead>
<tr>
<th>ON</th>
<th>Show the tic labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Hide the tic labels</td>
</tr>
</tbody>
</table>

**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:**

```plaintext
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
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:**

```python
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Color</th>
<th>DialName</th>
<th>OuterRadius</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnerRadius</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DialName** is the name of the dial in which the sector resides.

**Example:**

```python
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:**

```python
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**

\[
\text{ShowDataPoints} = \textit{ON} | \textit{OFF};
\]

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:**

\[
\begin{align*}
\text{ShowDataPoints} &= \text{ON}; \\
\text{ShowDataPoints} &= \text{OFF};
\end{align*}
\]

**Default**

OFF

**Attributes**
- Mode

**SliceAnimationStyle**

\[
\text{SliceAnimationStyle} = \text{GROW} | \text{FADE} | \text{NONE}
\]

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:**

\[
\text{LineAnimationStyle} = \textit{BEND};
\]

**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, MultiPie

**Example:**

```plaintext
SliceBorder = (DOTTED, 2, blue);
```

**Attributes**

<table>
<thead>
<tr>
<th>LineType</th>
<th>Width</th>
<th>Color</th>
</tr>
</thead>
</table>

**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:**

```plaintext
SliceColor = red, white, blue;
```

**Attributes**

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
</table>

**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:**

```plaintext
SliceData = 15, 33, 44, 102, 19.45, 88, 47.9;
```
Common CDL Attributes

Attributes
None

SliceFillPattern

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

The SliceFillPattern parameter provides a visual pattern fill for pie slices in a chart.

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

Used in These Charts
Pie MultiPie

Example:
   SliceFillPattern = (GRADIENTVERTICAL, blue, white);

Attributes
Type          Color1          Color2
ImageFormat    ImageURL
SliceFormat

SliceFormat = (FormatType, "FormatExpr");

In pie charts, SliceFormat affects the numeric labels that are automatically generated for each slice.

Used in These Charts
Pie MulitPie

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**

\[
\text{SliceLabel} = (\text{State}, \text{Color}, \text{FontName}, \text{FontSize}, \text{Angle}, \text{interiorAlignment});
\]

**SliceLabel** controls the appearance of all the slice labels in a pie chart. This parameter is a standard “tuple.”

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
</table>

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

**Used in These Charts**

Pie MulitPie

**Example:**

\[
\text{SliceLabel} = (\text{ON}, \text{Black}, \text{Helvetica}, 12, 270, \text{LEFT});
\]

**Attributes**

State  Color  FontName  FontSize  Angle  interiorAlignment

**State**

State indicates if this parameter is in effect or not.

**Example:**

\[
\text{SliceLabel} = (\text{ON}, \text{Black}, \text{Helvetica}, 12, 270);
\]

**Values**

ON or null

**Default**

ON

**SliceLabelBox**

\[
\text{SliceLabelBox} = (\text{Color}, \text{BorderType}, \text{BorderWidth});
\]

**SliceLabelBox** controls the appearance of the area under the slice label in a pie chart.

**Used in These Charts**

Pie MulitPie

**Example:**

\[
\text{SliceLabelBox} = (\text{fuschia}, \text{RECESS}, 4);
\]
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 MulitPie

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 SliceFormat parameter
LABEL The label passed as part of the Slices or SliceLabels parameter.

Default
LABEL

SliceLabelContentDelimiter
SliceLabelContentDelimiter = "delimiter";

SliceLabelContentDelimiter 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 MulitPie

Example:
SliceLabelContentDelimiter = ",";
SliceLabelContentDelimiter = "rrr";
SliceLabelContentDelimiter = "\n";
SliceLabelContentDelimiter = " ";
Common CDL Attributes

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.
Common CDL Attributes

**Used in These Charts**
Pie MultiPie

**Example:**
```plaintext
SliceLabelStyle = EXTERIOR;
SliceLabelStyle = LEGEND;
SliceLabelStyle = RADIAL;
```

**Attributes**

**Style**

Only one **Style** attribute is allowed for **SliceLabelStyle**.

**Example:**
```plaintext
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**

```plaintext
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:**
```plaintext
SlicePos = 0, 1.2, 1.5, 0, 0, 0, 0;
```
**Attributes**

**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");
```
Common CDL Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontName</td>
<td></td>
</tr>
<tr>
<td>FontSize</td>
<td></td>
</tr>
<tr>
<td>LabelBgBorder</td>
<td></td>
</tr>
<tr>
<td>LabelAngle</td>
<td></td>
</tr>
<tr>
<td>LabelBgColor</td>
<td></td>
</tr>
<tr>
<td>LabelColor</td>
<td></td>
</tr>
</tbody>
</table>

**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:**

```plaintext
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:**

```plaintext
Slices = (2300, x6C5D94, "Mon", black, "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, x03e3e3, NONE, 0),
         (2700, x515F23, "Sun", x515F23, "Helvetica", 12, 0, x03e3e3, 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:**

```plaintext
Slices = (2300, NULL, "Mon", black, "TimesRoman", 14),
        (3700, "Tue"),
        (1200, "Wed"),
        (2500, "Thu"),
        (4300, "Fri"),
        (1900, "Sat"),
        (2700, "Sun");
```

```plaintext
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:**

```plaintext
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, 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, and is in all respects specified as any other **Color** attribute.

**Example:**

```plaintext
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 **BorderType** attribute.

**Example:**

```plaintext
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**

```plaintext
SliceSet = Value1, Value2, Value3, Value4, ..., Valuen;
```

**SliceSet** defines the numeric values for each slice set.

**Used in These Charts**

MultiPie
Common CDL Attributes

SliceSets

SliceSets = ("Name", Color, "State");

SliceSet defines a list of SliceSet tuples with the following attributes in each tuple:

<table>
<thead>
<tr>
<th>Name</th>
<th>Color</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Color</td>
<td>State</td>
</tr>
<tr>
<td>Name</td>
<td>Color</td>
<td>State</td>
</tr>
</tbody>
</table>

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 = BOTTUMUP | 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
Common CDL Attributes

**Example:**

```
StackDisplayOrder = TOPDOWN;
```

---

**StackedBarConnectors**

```text
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**

```text
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;
```
Common CDL Attributes

<table>
<thead>
<tr>
<th>Style</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>Uses the accumulated numeric total for the label</td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>Uses the individual item’s numeric value for the label</td>
<td></td>
</tr>
<tr>
<td>PERCENT</td>
<td>Uses the accumulated numeric total for the label, displayed as a percentage of the total of all values.</td>
<td></td>
</tr>
</tbody>
</table>

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

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), (BOTTOM, RIGHT);
```

**Values**

<table>
<thead>
<tr>
<th>Bottom</th>
<th>X values will be plotted along the chart’s bottom axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>X values will be plotted along the chart’s top axis</td>
</tr>
</tbody>
</table>

**Default**

BOTTOM

---

**YAxis**

*YAxis*, 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), (BOTTOM, RIGHT);
```

**Values**

<table>
<thead>
<tr>
<th>Left</th>
<th>X values will be plotted along the chart’s left axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>X values will be plotted along the chart’s right axis</td>
</tr>
</tbody>
</table>

**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).
**Common CDL Attributes**

**Used in These Charts**
Stock

*Example:*
```
StockColorTable2 = , , red;
```

**Attributes**
None

**StockData[n]**

```
StockData[1-50] = (High1, Low1, Open1, Close1), (High2, 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**

<table>
<thead>
<tr>
<th>High</th>
<th>Low</th>
<th>Open</th>
<th>Close</th>
</tr>
</thead>
</table>

**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**), (32, 30.25, **29.75** , 31.5);
```

**Values**

- Any stock number
- **null** No tic mark will be drawn for this data point

---

**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

---

**StockFillPattern**

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

The `StockFillPattern` parameter provides a visual pattern fill for stock sets in a stock chart.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NONE</strong></td>
<td>no pattern, do default fill, if any</td>
</tr>
<tr>
<td><strong>Built-In Patterns</strong></td>
<td></td>
</tr>
<tr>
<td>FSLASH</td>
<td>front slash type</td>
</tr>
<tr>
<td>BSLASH</td>
<td>back slash type</td>
</tr>
<tr>
<td>DGRID</td>
<td>diagonal grid lines, (front and back slash lines)</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>horizontal lines</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>vertical lines</td>
</tr>
<tr>
<td>GRID</td>
<td>grid lines, (horizontal and vertical lines)</td>
</tr>
<tr>
<td><strong>Gradients</strong></td>
<td></td>
</tr>
<tr>
<td>GRADIENTVERTICAL</td>
<td>bottom to top style gradient</td>
</tr>
</tbody>
</table>
Common CDL Attributes

<table>
<thead>
<tr>
<th>GRADIENTHORIZONTAL</th>
<th>left to right style gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENTFDIAG</td>
<td>top right to bottom left style gradient</td>
</tr>
<tr>
<td>GRADIENTBDIAG</td>
<td>top left to bottom right style gradient</td>
</tr>
<tr>
<td>GRADIENTRADIAL</td>
<td>radial style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERHORIZONTAL</td>
<td>center out horizontal style gradient</td>
</tr>
<tr>
<td>GRADIENTCENTERVERTICAL</td>
<td>center out vertical style gradient</td>
</tr>
</tbody>
</table>

Images

| IMAGE | use an image specified in the optional imageURL element |

<table>
<thead>
<tr>
<th>color1</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Foreground color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Starting color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color 2</th>
<th>This color is used in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Background color for patterns</td>
</tr>
<tr>
<td></td>
<td>- Stopping color for gradients</td>
</tr>
<tr>
<td></td>
<td>- Ignored in images</td>
</tr>
</tbody>
</table>

| imageURL | 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"),


Common CDL Attributes

\[
\text{StockSets}[N] = (\text{Label}_1, \text{Color}_1, \text{Width}_1, \text{TicLen}_1), (\text{Label}_2, \text{Color}_2, \text{Width}_2, \text{TicLen}_2), \ldots;
\]

StockSets defines the display attributes for one or more stock data sets.

\textbf{Used in These Charts}

\textbf{Stock}

\textbf{Example:}

\begin{verbatim}
StockSets = ("DCX", azure, 1, 3),
            ("MSFT", darkblue, 1, 3),
            ("SEBL", moccasin, 1, 3);
\end{verbatim}

\textbf{Attributes}

\begin{tabular}{llll}
Label & Color & Width & TicLen \\
\end{tabular}

\textbf{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.

\textbf{Example:}

\begin{verbatim}
StockSets = ("DCX", azure, 1, 3),
            ("MSFT", darkblue, 1, 3),
            ("SEBL", moccasin, 1, 3);
\end{verbatim}

\textbf{Values}

Any pixel value

\begin{verbatim}
0 StockWidth attributes are used to determine the size of the bar
\end{verbatim}

\textbf{Default}

0

\textbf{TicLen}

TicLen defines length of the open/close tic marks in pixels. These pixel values override any StockWidth attributes for the current stock set.

\textbf{Example:}

\begin{verbatim}
StockSets = ("DCX", azure, 1, 3),
            ("MSFT", darkblue, 1, 4),
            ("SEBL", moccasin, 1, 5);
\end{verbatim}

\textbf{Values}

Any pixel value

\begin{verbatim}
0 StockWidth attributes are used to determine the width of the tic mark
\end{verbatim}
**Common CDL Attributes**

<table>
<thead>
<tr>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Width</th>
<th>TicLen</th>
</tr>
</thead>
</table>

**Width**

Width controls the width of the bars for stock data points.

**Example:**

StockWidth = (5, 80);
StockWidth = (.2, .5);

**Values**

<table>
<thead>
<tr>
<th>0 to 100 or 0.0 to 1.0</th>
<th>Width of bar in percentage of space allocated for the stock symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 or 1.0</td>
<td>The bar will occupy all of the space allocated for each stock symbol, leaving no room for the tic marks</td>
</tr>
</tbody>
</table>

**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);
**Common CDL Attributes**

**NumSlots**

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 100 or</td>
<td>Width of bar in percentage of space allocated for the stock symbol</td>
</tr>
<tr>
<td>0.0 to 1.0</td>
<td>The tic mark will fill up the remaining space not used by the bar</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**StripLayout**

\[
\text{StripLayout}[N] = (\text{NumSlots}, \text{InitialFill}, \text{MaxFill}, \text{UndefinedString})
\]

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**

<table>
<thead>
<tr>
<th>NumSlots</th>
<th>InitialFill</th>
<th>MaxFill</th>
<th>UndefinedString</th>
</tr>
</thead>
</table>

**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, "*"));
```
**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:*

```
```

*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
Common CDL Attributes

**Tics**

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.

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

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:**

```plaintext
TopTics = ("ON", black, "Helvetica", 10, LEFT);
```

**Attributes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>InteriorAlignment</td>
<td>BackgroundColor</td>
<td>RotationPoint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mode**

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

**Example:**

```plaintext
TopTics = ("ON", black, "Helvetica", 10);
```

**Values**

<table>
<thead>
<tr>
<th>ON</th>
<th>Show tic labels for this axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Don’t show tic labels on this axis</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Mode</th>
<th>SkipCount</th>
<th>StaggerLevels</th>
</tr>
</thead>
</table>

**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 staggering 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* is the number of visual "levels" to which axis labels can or should be staggered for good visibility.

**Example:**

```
TopTicLayout = (AUTOSKIP, 5, 3);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or 1</td>
<td>No stagger occurs</td>
</tr>
<tr>
<td>&gt;=2</td>
<td>Integer Number of text lines staggered</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No skipping occurs</td>
</tr>
<tr>
<td>&gt;0</td>
<td>Integer Number of axis tics skipped</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
</table>

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

Used in These Charts
Combo

Example:
TopAxis = ("", black, "TimesRoman", 16, 0);
Common CDL Attributes

**Attributes**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
</tr>
</thead>
</table>

**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.

<table>
<thead>
<tr>
<th>interiorAlignment</th>
<th>Specifies the alignment to use in text strings that contain multiple lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>exteriorAlignment</td>
<td>Specifies the alignment for the entire Title object.</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Label</th>
<th>Color</th>
<th>FontName</th>
<th>FontSize</th>
<th>Angle</th>
<th>interiorAlignment</th>
<th>exteriorAlignment</th>
</tr>
</thead>
</table>

**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**

<table>
<thead>
<tr>
<th>Label</th>
<th>URL</th>
<th>Target</th>
</tr>
</thead>
</table>
Common CDL Attributes

**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**

<table>
<thead>
<tr>
<th>Color</th>
<th>BorderType</th>
<th>BorderWidth</th>
<th>ImageURL</th>
<th>ImageFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderColor</td>
<td>TRCornerStyle</td>
<td>BRCornerStyle</td>
<td>BLCornerStyle</td>
<td>CornerColor</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
</table>

**TopDrawMinorTics**

```
TopDrawMinorTics = ON| OFF;
```

**TopDrawMinorTics** controls whether or not Top tics are drawn. The default value is ON.
Common CDL Attributes

**Example:**

```plaintext
TopDrawMinorTics = OFF;
```

**Attributes**

(Switch)

---

**TopFormat**

```plaintext
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:**

```plaintext
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**

<table>
<thead>
<tr>
<th>FormatType</th>
<th>FormatExpr</th>
<th>TimeBase</th>
<th>TimeUnit</th>
</tr>
</thead>
</table>

**FormatType**

FormatType specifies the type of number being displayed on the top axis.

**Example:**

```plaintext
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 values 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.

**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.
**Common CDL Attributes**

**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:**

```
```

**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

MinValue MaxValue StepValue

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:**

```plaintext
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:**

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

**Values**
Any real number, date, or time between MinValue and MaxValue

**Default**
1

**TopScroll**

```plaintext
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:**

```plaintext
TopScroll = (0, 98);
```

**Attributes**

- **ScrollMin**
- **ScrollMax**

**ScrollMin**

ScrollMin sets the lower visible limit for a scrollbar defined with TopScroll
Common CDL Attributes

ScrollMax

ScrollMax sets the upper visible limit for a scrollbar defined with TopScroll.

Example:

    TopScroll = (0, 98);

Values

<MinValue

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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interiorAlignment</td>
<td>Specifies the alignment to use in text strings that contain multiple lines.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Background color of the tic label area</td>
</tr>
<tr>
<td>rotationPoint</td>
<td>For rotated axis labels, anchor point for the rotation</td>
</tr>
</tbody>
</table>

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
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.

Usage in These Charts
Bar, Box, Combo, Bubble, Line, Pareto, Stock, Strip, Time, X-Y

Example:
```
TopTicLayout = (AUTOSTAGGER, 5, 3);
```

Attributes
Mode SkipCount StaggerLevels

Mode
controls the type of layout mode that should be used with the axis labels.

Example:
```
TopTicLayout = (AUTOSKIP, 5, 3);
```
Common CDL Attributes

StaggerLevels

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

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*

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOX</td>
<td>BOX is 40% of the main box width</td>
</tr>
<tr>
<td>Line</td>
<td>Line whiskers are 1 pixel</td>
</tr>
</tbody>
</table>

*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 viewpoint 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**

<table>
<thead>
<tr>
<th>CoordinateSystem</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Radius</th>
<th>Phi</th>
<th>Theta</th>
</tr>
</thead>
</table>

### 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 3D barchart 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 viewpoint to (0, 0, a). To look at the chart from a 45 degree angle, set the viewpoint 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 magnitude 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:**

```plaintext
LeftTitle = ("Rotated Label", black, "TimesRoman", 16, 90, LEFT);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>degrees counterclockwise from horizontal</td>
</tr>
<tr>
<td>90</td>
<td>degrees counterclockwise from horizontal</td>
</tr>
<tr>
<td>180</td>
<td>degrees counterclockwise from horizontal</td>
</tr>
<tr>
<td>270</td>
<td>degrees counterclockwise from horizontal</td>
</tr>
</tbody>
</table>

**Default**

0

**BorderColor**

*BorderColor* defines the color used to draw a region border when the BOX value of *BorderType* is used.

**Example:**

```plaintext
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:**

```plaintext
LeftTitleBox = (yellow, RAISED, 10);
```
Common CDL Attributes

BorderWidth

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No border</td>
</tr>
<tr>
<td>BOX</td>
<td>Simple Box outline</td>
</tr>
<tr>
<td>SHADOW</td>
<td>Shadow border</td>
</tr>
<tr>
<td>RAISED</td>
<td>Raised border</td>
</tr>
<tr>
<td>RECESS</td>
<td>Recessed border</td>
</tr>
</tbody>
</table>

Default
NONE

BorderWidth

<table>
<thead>
<tr>
<th>.BorderWidth defines the width of the edges of a visual region such as a background or label.</th>
</tr>
</thead>
</table>

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:

- 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 DataSets parameter in the NFBarchartApp accepts a NULL parameter, whereas the Background parameter does not accept NULL colors.
Translucency can be added to any color specification by appending a translucency qualifier between 0 and 250. For example \texttt{red\_0} or \texttt{xFF0000\_0} is completely translucent, while \texttt{red\_250} and \texttt{xFF0000\_250} both represent solid red.

\textit{Default}

Black for text, gray for backgrounds and chart elements

\textbf{Table 1: Predefined Color Names Used in Visual Mining Applets}

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Name</th>
<th>Color Name</th>
<th>Color Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>antiquewhite</td>
<td>dimgray</td>
<td>magenta</td>
<td>salmon</td>
</tr>
<tr>
<td>aquamarine</td>
<td>dodgerblue</td>
<td>maroon</td>
<td>slategray</td>
</tr>
<tr>
<td>azure</td>
<td>firebrick</td>
<td>mediumaquamarine</td>
<td>sandybrown</td>
</tr>
<tr>
<td>beige</td>
<td>floralwhite</td>
<td>mediumblue</td>
<td>saddlebrown</td>
</tr>
<tr>
<td>bisque</td>
<td>forestgreen</td>
<td>mediumpurple</td>
<td>seagreen</td>
</tr>
<tr>
<td>black</td>
<td>fuchsia</td>
<td>mediumseagreen</td>
<td>seashell</td>
</tr>
<tr>
<td>blue</td>
<td>gainsboro</td>
<td>mediumslateblue</td>
<td>sienna</td>
</tr>
<tr>
<td>blueviolet</td>
<td>ghostwhite</td>
<td>mediumturquoise</td>
<td>slateblue</td>
</tr>
<tr>
<td>brown</td>
<td>goldenrod</td>
<td>mediumturquoise</td>
<td>silver</td>
</tr>
<tr>
<td>burlywood</td>
<td>gold</td>
<td>midnightblue</td>
<td>snow</td>
</tr>
<tr>
<td>cadetblue</td>
<td>gray</td>
<td>mintcream</td>
<td>springgreen</td>
</tr>
<tr>
<td>chartreuse</td>
<td>green</td>
<td>moccasin</td>
<td>steelblue</td>
</tr>
<tr>
<td>chocolate</td>
<td>greenyellow</td>
<td>midnightblue</td>
<td>snow</td>
</tr>
<tr>
<td>coral</td>
<td>honeydew</td>
<td>olive</td>
<td>teal</td>
</tr>
<tr>
<td>cornflowerblue</td>
<td>hotpink</td>
<td>olivedrab</td>
<td>turquoise</td>
</tr>
<tr>
<td>cornsilk</td>
<td>indigo</td>
<td>orange</td>
<td>violet</td>
</tr>
<tr>
<td>crimson</td>
<td>indianred</td>
<td>orangered</td>
<td>wheat</td>
</tr>
<tr>
<td>cyan</td>
<td>ivory</td>
<td>orchid</td>
<td>whitesmoke</td>
</tr>
<tr>
<td>darkblue</td>
<td>khaki</td>
<td>palegoldenrod</td>
<td>white</td>
</tr>
<tr>
<td>darkcyan</td>
<td>lavenderblush</td>
<td>palevioletred</td>
<td>yellow</td>
</tr>
<tr>
<td>darkgoldenrod</td>
<td>lawngreen</td>
<td>palegreen</td>
<td>yellowgreen</td>
</tr>
<tr>
<td>darkgray</td>
<td>lemonchiffon</td>
<td>paleturquoise</td>
<td></td>
</tr>
<tr>
<td>darkgreen</td>
<td>lightblue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>darkkhaki</td>
<td>lightcoral</td>
<td></td>
<td></td>
</tr>
<tr>
<td>darkmagenta</td>
<td>lightgoldenrodyellow</td>
<td>papayawhip</td>
<td></td>
</tr>
<tr>
<td>darkorchid</td>
<td>lightgrey</td>
<td>peachpuff</td>
<td></td>
</tr>
<tr>
<td>darkorange</td>
<td>lightgreen</td>
<td>peru</td>
<td></td>
</tr>
<tr>
<td>darkred</td>
<td>lightpink</td>
<td>pink</td>
<td></td>
</tr>
<tr>
<td>darksalmon</td>
<td>lightsalmon</td>
<td>plum</td>
<td></td>
</tr>
<tr>
<td>darkseagreen</td>
<td>lightseagreen</td>
<td>powderblue</td>
<td></td>
</tr>
<tr>
<td>darkslateblue</td>
<td>lightskyblue</td>
<td>purple</td>
<td></td>
</tr>
<tr>
<td>darkslategray</td>
<td>lightslategray</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td>darkturquoise</td>
<td>lightsteelblue</td>
<td>rosybrown</td>
<td></td>
</tr>
<tr>
<td>darkviolet</td>
<td>lightyellow</td>
<td>royalblue</td>
<td></td>
</tr>
<tr>
<td>darkolivegreen</td>
<td>lime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deeppink</td>
<td>limegreen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deepskyblue</td>
<td>linen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textit{FontName}

\textit{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

<table>
<thead>
<tr>
<th>Java font</th>
<th>X-11 font</th>
<th>Windows Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>misc-fixed</td>
<td>Arial</td>
</tr>
<tr>
<td>Helvetica</td>
<td>adobe-helvetica</td>
<td>Arial</td>
</tr>
<tr>
<td>TimesRoman</td>
<td>adobe-times</td>
<td>Times New Roman</td>
</tr>
<tr>
<td>Courier</td>
<td>adobe-courier</td>
<td>Courier New</td>
</tr>
<tr>
<td>Dialog</td>
<td>b&amp;h-lucida</td>
<td>MS Sans Serif</td>
</tr>
</tbody>
</table>

**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 ‘%’.
**Common CDL Attributes**

**Example:**
```
LeftFormat = (FLOAT,"%$\%9.2f" ,);
SliceFormat = (FLOAT,"%5.2f" );
```

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>%g</td>
<td>1234.456</td>
</tr>
<tr>
<td>%f</td>
<td>1234.46</td>
</tr>
<tr>
<td>%,f</td>
<td>1,234.46</td>
</tr>
<tr>
<td>%d</td>
<td>1234</td>
</tr>
<tr>
<td>%,d</td>
<td>1,234</td>
</tr>
<tr>
<td>%8.1f</td>
<td>1234.5</td>
</tr>
<tr>
<td>%08.1f</td>
<td>001234.5</td>
</tr>
<tr>
<td>Cost=$%,.1fM</td>
<td>Cost=$1,234.5M</td>
</tr>
</tbody>
</table>

**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:

- 0WWW.DD

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.

**Default**

| %.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:

$DOCBASE  document base
$CODEBASE  code base
$NETCHARTS  $CODEBASE/netcharts
$ICONS  $NETCHARTS/icons
$IMAGES  $NETCHARTS/images
$SYMBOLS  $NETCHARTS/symbols
$PATTERNS  $NETCHARTS/patterns

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:**

```plaintext
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

---

**LineType**

LineType 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:**

```plaintext
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:**

```plaintext
GridLine = (VERTICAL, BAR), (HORIZONTAL, DOTTED, 2);
```

**Values**

Integer number of pixels of line width

**Default**

1
**Common CDL Attributes**

**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 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:

- **_blank** — Browser creates a new window for the destination document.
- **_parent** — Destination document replaces the current frame’s framesetting document, if one exists; otherwise, defaults to _self.
Common CDL Attributes

_self
Destination document replaces the current document in its window or frame.

_top
Destination document is to occupy the entire browser window, replacing any and all framesets that may be loaded.

LOADPARAMS
Points to a NetCharts parameter file, which is specified by the URL 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.
Common CDL Attributes

YAxis

Example:

\texttt{GridAxis = (BOTTOM, LEFT), (TOP, RIGHT);} 

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>top of chart</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>bottom of chart</td>
</tr>
</tbody>
</table>

Default

BOTTOM

YAxis

Specifies which side of the chart becomes the Y axis.

Example:

\texttt{GridAxis = (BOTTOM, LEFT), (TOP, RIGHT);} 

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT</td>
<td>left of chart</td>
</tr>
<tr>
<td>RIGHT</td>
<td>right of chart</td>
</tr>
</tbody>
</table>

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 20th 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 DataSets / LineSets / StockSets / BubbleSets 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:
Appendix A: Date and Time Values

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:

```plaintext
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:

<table>
<thead>
<tr>
<th>Time Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Years</td>
</tr>
<tr>
<td>M</td>
<td>Months</td>
</tr>
<tr>
<td>d</td>
<td>Days</td>
</tr>
<tr>
<td>h</td>
<td>Hours</td>
</tr>
<tr>
<td>m</td>
<td>Minutes</td>
</tr>
<tr>
<td>s</td>
<td>Seconds</td>
</tr>
</tbody>
</table>
Appendix A: Date and Time Values

Date/Time Data Format and TimeBase

Example:

"?d"   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:

\[ \text{Absolute DateTime} = \text{Axis TimeBase} + N \times \text{Axis TimeUnit} \]

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 tick 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Attribute</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>yyyy</td>
<td>1999</td>
</tr>
<tr>
<td>Year</td>
<td>yy</td>
<td>99</td>
</tr>
<tr>
<td>Month</td>
<td>MMM</td>
<td>January</td>
</tr>
<tr>
<td>Month</td>
<td>MM</td>
<td>01</td>
</tr>
<tr>
<td>Month</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>Day of week</td>
<td>EEEE</td>
<td>Monday</td>
</tr>
<tr>
<td>Day of week</td>
<td>EE</td>
<td>Mon</td>
</tr>
<tr>
<td>Day of month</td>
<td>dd</td>
<td>01</td>
</tr>
<tr>
<td>Day of month</td>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>Hour (1-12)</td>
<td>hh</td>
<td>01</td>
</tr>
<tr>
<td>Hour (1-12)</td>
<td>h</td>
<td>1</td>
</tr>
<tr>
<td>Hour (0-23)</td>
<td>HH</td>
<td>00</td>
</tr>
<tr>
<td>Hour (0-23)</td>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>Hour (0-11)</td>
<td>KK</td>
<td>00</td>
</tr>
<tr>
<td>Hour (0-11)</td>
<td>K</td>
<td>0</td>
</tr>
<tr>
<td>Hour (1-24)</td>
<td>kk</td>
<td>01</td>
</tr>
<tr>
<td>Hour (1-24)</td>
<td>k</td>
<td>1</td>
</tr>
<tr>
<td>Minute</td>
<td>mm</td>
<td>00</td>
</tr>
<tr>
<td>Second</td>
<td>ss</td>
<td>00</td>
</tr>
<tr>
<td>Millisecond</td>
<td>SSS</td>
<td>001</td>
</tr>
<tr>
<td>AM/PM</td>
<td>a</td>
<td>AM</td>
</tr>
<tr>
<td>Time zone</td>
<td>zzzz</td>
<td>EST</td>
</tr>
<tr>
<td>Time zone</td>
<td>zz</td>
<td>ET</td>
</tr>
<tr>
<td>Day of week in month</td>
<td>F</td>
<td>3rd</td>
</tr>
<tr>
<td>Day in year</td>
<td>DDD</td>
<td>001</td>
</tr>
<tr>
<td>Day in year</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>Week in year</td>
<td>ww</td>
<td>1</td>
</tr>
<tr>
<td>Era</td>
<td>G</td>
<td>AD</td>
</tr>
</tbody>
</table>

**DATE Format Expression Attributes**

The following attributes are supported for the DATE format type:

<table>
<thead>
<tr>
<th>Field</th>
<th>Attribute</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locale form</td>
<td>%L</td>
<td>Mon Jan 01 00:00:00 2000</td>
</tr>
<tr>
<td>GMT form</td>
<td>%G</td>
<td>1 Jan 2001 00:00:00 GMT</td>
</tr>
<tr>
<td>Year</td>
<td>%Y</td>
<td>1999</td>
</tr>
<tr>
<td>Year</td>
<td>%y</td>
<td>99</td>
</tr>
</tbody>
</table>
Appendix A: Date and Time Values

Date/Time Data Format and TimeBase

<table>
<thead>
<tr>
<th>Month</th>
<th>%M</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Month</td>
<td>%N</td>
<td>January</td>
</tr>
<tr>
<td>Month</td>
<td>%n</td>
<td>Jan</td>
</tr>
<tr>
<td>Weekday</td>
<td>%W</td>
<td>Monday</td>
</tr>
<tr>
<td>Weekday</td>
<td>%w</td>
<td>Mon</td>
</tr>
<tr>
<td>Day of month</td>
<td>%d</td>
<td>01</td>
</tr>
<tr>
<td>Hour</td>
<td>%h</td>
<td>00</td>
</tr>
<tr>
<td>Minute</td>
<td>%m</td>
<td>00</td>
</tr>
<tr>
<td>Second</td>
<td>%s</td>
<td>00</td>
</tr>
</tbody>
</table>

**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/Jun/01 -30 days from 15 Jul 2001
- 25/Jun/01 -20 days from 15 Jul 2001
- 05/Jul/01 -10 days from 15 Jul 2001
- 15/Jul/01 0 days from 15 Jul 2001
- 25/Jul/01 +10 days from 15 Jul 2001
- 04/Aug/01 +20 days from 15 Jul 2001
- 14/Aug/01 +30 days from 15 Jul 2001
- 24/Aug/01 +40 days from 15 Jul 2001
Appendix A: Date and Time Values

Date/Time Data Format and TimeBase

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
A

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
AxisScale, 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], 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
BottomTickLength, 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
CellTextAutoColorThreshold, 102
ChartElementSpacing, 103
ChartHeight, 105
ChartName, 105
Charts, 103
ChartScript[n], 106
ChartType, 106
ChartURL[n], 107
ChartWidth, 107
color, 151
decimal format, 314
hexadecimal format, 314
Index

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
CumulativeLineSetName, 108
CumulativeLineStyle, 109
CumulativeLineSymbol, 110
CumulativeLineValueLabel, 111
CumulativeLineValueLabelBox, 112
CumulativeLineValueLabelStyle, 112
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
dividng 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
DialTickLabels, 132
DialTickLabelStyles, 133
DialTics, 134
DrawFences, 57, 101, 135
DrawOrder, 135
drill-down
parameters associated with, 44
dwell label
box, 53
box background, 137
dwell labels, 44
DwellAnimationHighlightBorderStyle, 135
DwellAnimationHighlightFill, 136
DwellAnimationStyle, 137
DwellBox, 137
DwellLabel, 55, 138
DwellOffset, 138

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
dividng 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
DialTickLabels, 132
DialTickLabelStyles, 133
DialTics, 134
DrawFences, 57, 101, 135
DrawOrder, 135
drill-down
parameters associated with, 44
dwell label
box, 53
box background, 137
dwell labels, 44
DwellAnimationHighlightBorderStyle, 135
DwellAnimationHighlightFill, 136
DwellAnimationStyle, 137
DwellBox, 137
DwellLabel, 55, 138
DwellOffset, 138
Index

E

Edges, 139
EightyLineSetName, 142
EightyTwentyLineStyle, 142
EightyTwentyLineSymbol, 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, 152
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, 49
lines in, 49
parameters specific to, 49
GroupStackLabels, 161
GroupStackSegmentLabels, 161

H

HandActiveLabels, 161
HandBorders, 161
HandButtonDownBorder, 162
HandButtonDownEdgeHighlights, 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 NFParamScript, 106
sashes dividing charts, 260
multipie
specifying pie edge highlights, 167, 231, 267
multipie 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
NFParamURL, 11, 12
NodeBox, 215
NodeDrag, 216, 221
Index

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
NumberFormatException, 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
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

Target attribute, 320
TaskColorTable, 294
TaskHeigth, 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
underline, 50
UniqueTaskColors, 310
Update, 309
URL attribute, 321
WhiskerType, 310
Width attribute, 321
Windows font names, 316
WW, 317
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
YAxis attribute, 322
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
General Phone   800.308.0731
International   +1.301.795.2200
Fax             301.947.8293
General Inquiries  info@visualmining.com
Customer Support support@visualmining.com
Sales           sales@visualmining.com
Press and Media Inquiries marketing@visualmining.com

Web
http://www.visualmining.com