Information Builders

WebFOCUS

Creating HTML5 Charts With WebFOCUS Language

Release 8205

February 26, 2019

Contents

Preface

This content explains how to use the JavaScript® Chart Engine (Js:Chart). Js:Chart draws a chart in an HTML environment using a defined data set and JavaScript® Object Notation (JSON) definitions. The JSON object(s) define chart properties that control the appearance of the chart and chart objects.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	**Contents**
1	Introduction to WebFOCUS HTML5 Charts	Introduces HTML5 charts and how to generate them in WebFOCUS.
2	Understanding Chart Types	Describes the chart types available for creating HTML5 charts with examples of each type.
3	Introduction to JSON Properties for HTML5 Charts	Describes JSON properties and how to specify colors, numbers, fonts, and callbacks.
4	WebFOCUS Chart Attribute Syntax	Describes chart attribute syntax for creating HTML5 charts.
5	Chart-Wide Properties	Describes JSON properties that are not specific to a particular chart type or component.
6	Chart Title Properties	Describes JSON properties that format the chart title, subtitle, and footnote.
7	Legend Properties	Describes how to control the format of the legend area.
8	Axis Properties	Describes JSON properties used to format chart axes.
9	Series-Specific Properties	Describes JSON properties that format series in a chart request.
10	Chart-Specific Properties	Describes JSON properties that are specific to each chart type.
11	Generating Narrative Charts	Describes how to generate Narrative Charts

	Chapter/Appendix	Contents
12	Special Topics	Describes special features for HTML5 charts.
13	Map Support	Describes how to generate map charts.
14	Adding Your Own Chart Types to the Chart Library	Describes how to add custom chart types written by the user to the chart library.
A	WebFOCUS Chart Parameters	Describes common WebFOCUS parameters used with chart requests.
B	Converting Requests to Chart Attribute Syntax	Describes how to convert legacy chart requests to chart attribute syntax.
C	Cumulative List of Changes to JSON Properties	Lists JSON properties that have been deprecated or changed.

Conventions

The following table describes the conventions that are used in this manual.

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices. Type one of them, not the braces.

Convention	Description
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Content Library at *http://documentation.informationbuilders.com*. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at *http://forums.informationbuilders.com/eve/forums*.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, *http://www.informationbuilders.com*. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of *www.informationbuilders.com* also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions effectively, be prepared to provide the following information when you call:

- ❏ Your six-digit site code (*xxxx.xx*).
- ❏ Your WebFOCUS configuration:
  - ❏ The front-end software you are using, including vendor and release.
  - ❏ The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
  - ❏ The software release.
  - ❏ Your server version and release. You can find this information using the Version option in the Web Console.
- ❏ The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- ❏ The Master File and Access File.
- ❏ The exact nature of the problem:
  - ❏ Are the results or the format incorrect? Are the text or calculations missing or misplaced?
  - ❏ Provide the error message and return code, if applicable.
  - ❏ Is this related to any other problem?
- ❏ Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- ❏ What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?

- ❑ Is this problem reproducible? If so, how?
- ❑ Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- ❑ Do you have a trace file?
- ❑ How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website, *http://documentation.informationbuilders.com/connections.asp*.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (*http://education.informationbuilders.com*) or call (800) 969-INFO to speak to an Education Representative.

Chapter 1

Introduction to WebFOCUS HTML5 Charts

The JavaScript® Chart Engine (FORMAT JSCHART) draws a chart in a web browser (HTML environment) using a WebFOCUS GRAPH request, optionally extended with JavaScript® Object Notation (JSON) definitions. The JSON objects define chart properties that refine the appearance of the chart and chart objects.

In this chapter:

WebFOCUS Charting Overview

Charts allow you to present information graphically, using such visual cues as color, size, and position to convey relationships between measures (numeric fields to be aggregated) and dimensions (categories) and to identify trends and outliers. WebFOCUS can create a wide variety of charts, generated in your choice of output formats.

One way that WebFOCUS produces charts is by having the Reporting Server generate them, in which case they are delivered to the browser as bitmap images (such as .png, .gif, or .jpg) or in a vector format embedded in a PDF document.

HTML5 charts, in contrast, are created as JavaScript code that runs directly in the browser, producing highly-interactive visualizations. These charts conform to all current web standards and are supported by all modern browsers. For a complete list of browsers and browser releases supported, see the *WebFOCUS Release Guide*.

Although you can use WebFOCUS tools to create charts by dragging and dropping elements onto a canvas and making selections from a toolbar or ribbon, the basis of this process is the familiar and easily comprehensible WebFOCUS language. In addition to these simple WebFOCUS commands, HTML5 charts support a JSON API that allows you to completely customize the final output. Multiple charts can even be linked.

Also built into the WebFOCUS HTML5 charting product is map support. You can create choropleth maps, in which areas are color-coded, or proportional symbol maps that display bubbles of different sizes over relevant areas of the map.

In order to create an HTML5 chart using the WebFOCUS language, all you have to do is include the PCHOLD FORMAT JSCHART command in your request.

In addition, there is a powerful chart attribute syntax that uses WebFOCUS StyleSheet commands to place chart objects into attribute categories, giving you additional options and chart types for generating meaningful visualizations. For complete information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

The data source referenced in a WebFOCUS chart request can be a joined or cross-referenced data source, and can be stored in the Hyperstage cache or in any supported DBMS, across all operating environments. Dialogue Manager variables and commands can be used in the request to make the chart customizable based on user selections at run-time. In addition, WebFOCUS charts support a wide variety of WebFOCUS styling options.

Each data source referenced in a request needs a WebFOCUS description, called a synonym. A synonym consists of a Master File (which describes the data) and, in most cases, an associated Access File (which describes how to access the data). You can use the Synonym Editor to create synonyms automatically for most types of data sources. For information about creating synonyms, see the *Describing Data With WebFOCUS Language* manual.

JSON objects define values for enhanced chart properties. In addition to complete JSON objects, format JSCHART can accept JSON objects that only include one or a few properties. These property values are used in place of any default or existing settings.

Note: The terms chart and graph are used interchangeably unless otherwise noted.

Chart Types

Following are descriptions of some popular types of charts you can create using JSCHART format:

- **Line charts.** Line charts are useful for emphasizing the movement or trend of numeric data over time, since they allow a viewer to trace the evolution of a particular point by working backwards or interpolating. Highs and lows, rapid or slow movement, or a tendency toward stability are all types of trends that are well suited to a line chart.

  Line charts can also be plotted with two or more scales to suggest a comparison of the same value, or set of values, in different time periods. The number of scales your chart has depends on the type of chart you select. For details, see *Controlling Chart Type Using LOOKGRAPH* on page 40.

- **Bar charts.** A bar or column chart plots numeric data by displaying rectangular blocks against a scale. The length of a bar corresponds to a value or amount. Viewers can develop a clear mental image of comparisons among data series by distinguishing the relative heights of the bars. Use a bar chart to display numeric data when you want to present distributions of data. You can create horizontal as well as vertical bar charts.

- **Pie charts.** A pie chart emphasizes where your data fits in relation to a larger whole. Keep in mind that pie charts work best when your data consists of several large sets. Too many variables divide the pie into small segments that are difficult to see. Use color or texture on individual segments to create visual contrast.

- **Scatter charts.** Scatter charts show the relationship between two different numeric measures. Scatter plots give you a sense of trends, concentrations, and outliers that pinpoint where to focus further investigation efforts.

- **Area charts.** Area charts are similar to line charts except that the area between the data line and the zero line (or axis) is usually colored or textured. Area charts allow you to stack data on top of each other. Stacking allows you to highlight the relationship between data series, showing how some data series approach or shadow a second series.

- **3D charts.** 3D charts add dimension to your presentation. Dimensionality allows your viewers to recognize trends based on two or more data sets. 3D charts also add impact to your presentation.

- **Polar charts.** A polar chart is a circular form of scatter plot in which each point is defined by an angle and distance from the center of the chart.

- **Radar charts.** A radar chart is a circular form of line or area chart. Radar charts work well with any data that are cyclical, such as the months of a year.

- **Stock charts.** A stock chart plots the trend or changes in financial instruments over time.

- **Bubble charts.** A bubble chart is an enhanced scatter plot in which the size of each marker is proportional to the value of a third measure.

- **Spectral charts.** A spectral chart, also known as a *heatmap*, is a table in which the color of each cell is dependent on the value in the cell.

- **Other charts.** Many additional chart types are available with the LOOKGRAPH parameter, including waterfall charts (which are used for understanding how an initial value is affected by a series of intermediate positive or negative values).

- ❏ **HTML5-specific charts.** Some chart types are only available with FORMAT JSCHART. Among these are map charts (both choropleths that use color to differentiate between value groups, and proportional symbol maps that use the size of a bubble as the differentiator), tagclouds (which display a word in a size proportional to its frequency), and treemaps (which display hierarchical data as a set of nested rectangles).

For a complete list of the types of charts you can create and the syntax for generating each type, see *Controlling the Chart Type* on page 40.

Creating an HTML5 Chart

To generate an HTML5 chart, include the following command In your GRAPH FILE request:

```
ON GRAPH PCHOLD FORMAT JSCHART
```

If the ON GRAPH PCHOLD FORMAT JSCHART command is not issued, server-side graphics are generated.

WebFOCUS styling commands and libraries of JSON objects and legacy graph API calls are available to customize the final output to make it conform to your needs and standards.

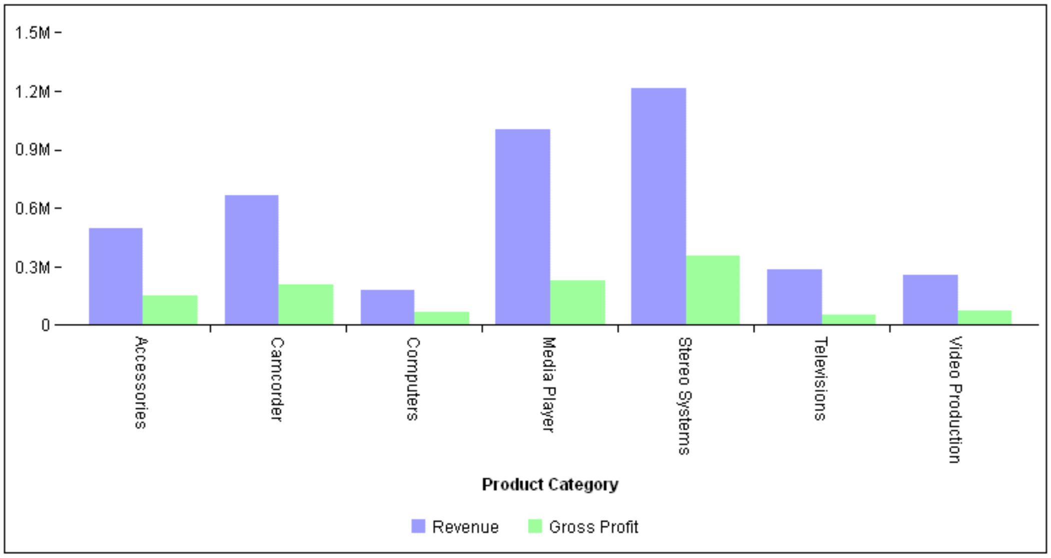
If you do not specify JSON properties, default properties will be used. If you do not specify a WebFOCUS StyleSheet to be used, default styles will be used.

Example: Creating an HTML5 Vertical Bar Graph

The following request against the WF_RETAIL_LITE data source creates an HTML5 vertical bar chart. As it includes no StyleSheet or JSON properties, default properties are used:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
END
```

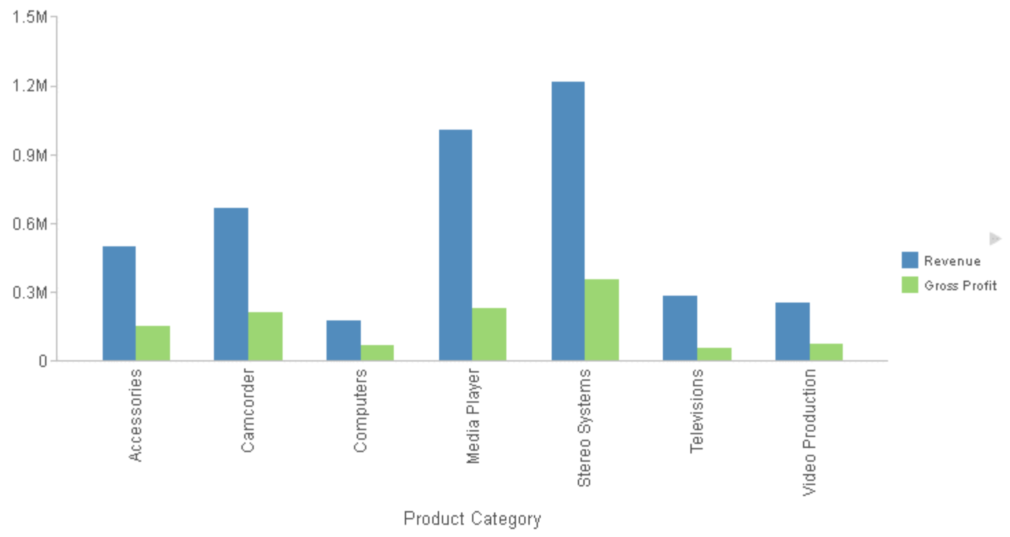
The output is:



The following version of the request uses chart attribute syntax and applies a StyleSheet.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=X-AXIS,$
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=Y-AXIS,$
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=Y-AXIS,$
ENDSTYLE
END
```

The output is shown in the following image.



Syntax: How to Resize HTML5 Graph Output to Fit Its Container

You can use the SET AUTOFIT command to make the HTML5 output resize to fit into the container in which it is placed.

```
ON GRAPH SET AUTOFIT {OFF|ON|RESIZE}
```

where:

OFF

Respects the dimensions specified by the HAXIS and VAXIS parameters.

ON

Always resizes the HTML5 chart output to fit its container.

RESIZE

Respects the dimensions specified by the HAXIS and VAXIS parameters initially, but resizes the chart output if the container is resized.

Note:

- With AUTOFIT enabled, the output from one GRAPH FILE request is sized to fit the entire draw area. Therefore, if you have a procedure that includes multiple GRAPH FILE requests, set the AUTOFIT parameter to OFF.

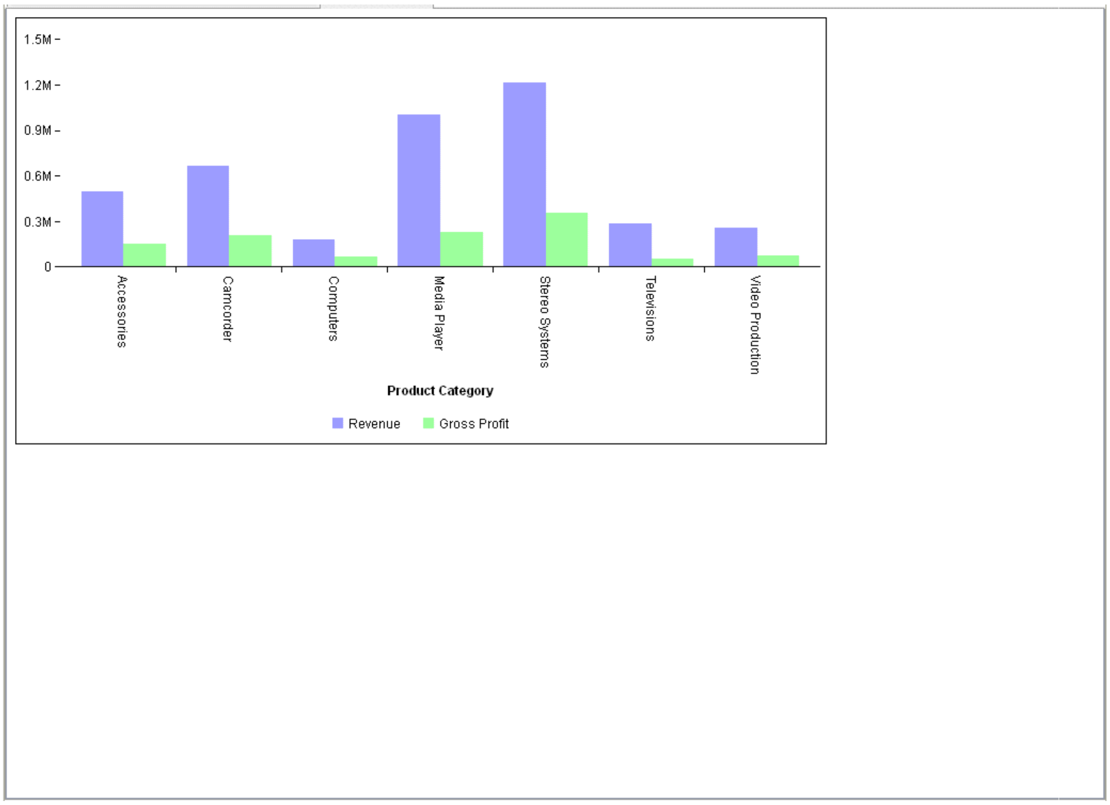
- AUTOFIT ON is ignored for a multi-graph request. For more information about multi-graph requests, see *Creating Multiple Graphs* on page 983.
- If a procedure has multiple GRAPH FILE requests, only the last one will display on the output with SET AUTOFIT=ON.

Example: Using SET AUTOFIT

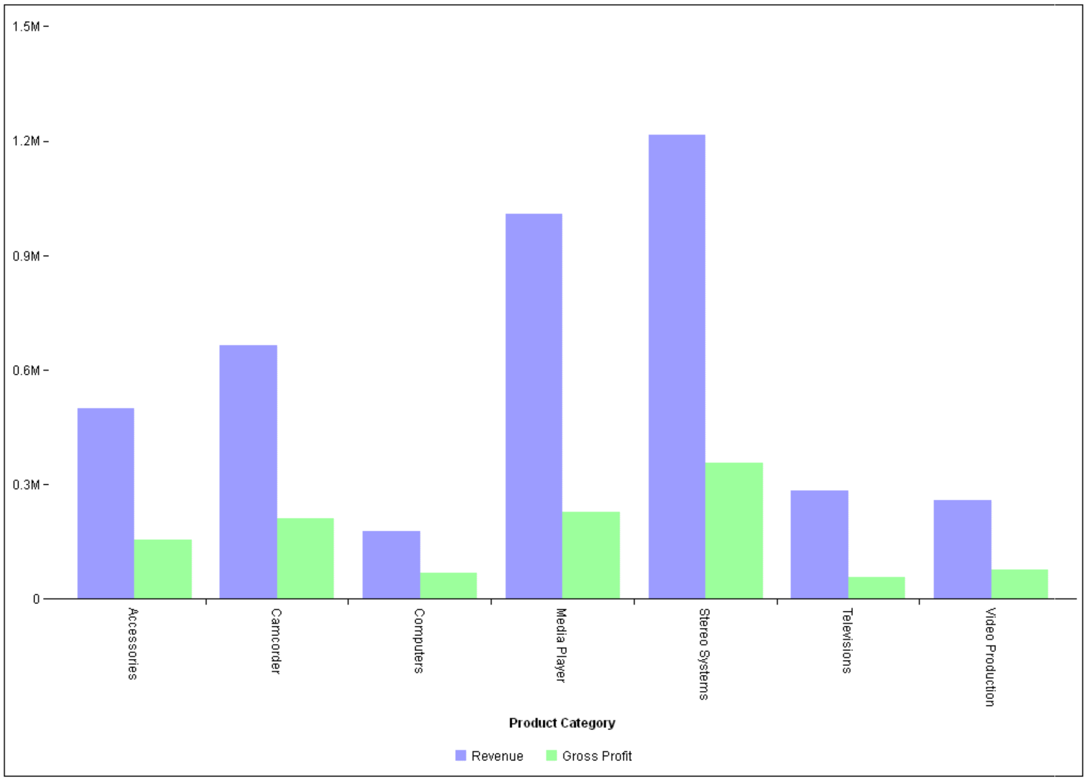
The following request created a vertical bar chart. the AUTOFIT parameter is set to OFF:

```
SET AUTOFIT = OFF
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
END
```

The size of the chart is controlled by the default settings for the HAXIS and VAXIS parameters:



If you change the AUTOFIT parameter value to ON, the chart is drawn using the entire page, ignoring the HAXIS and VAXIS values.



It then resizes if you resize the container.

Changing the AUTOFIT setting to RESIZE initially renders the chart as with the OFF setting, but resizes it if the container is reduced or enlarged.

Reference: Scrollbar Generation for Bar, Line, and Area Charts

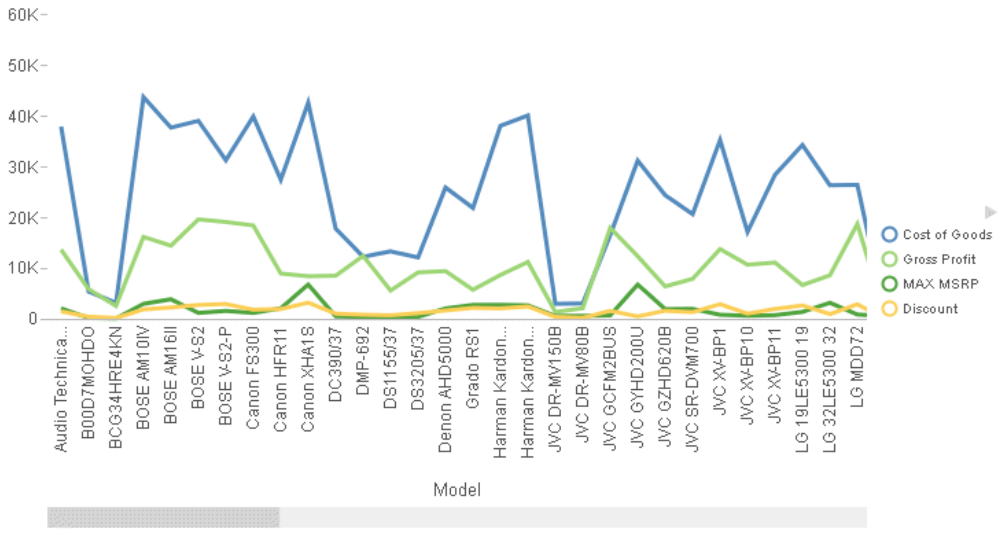
When bar, line, and area charts have a large number of group entries, a scrollbar will be generated so that all groups can be seen by scrolling.

Example: **Generating a Scroll Bar for a Line Chart**

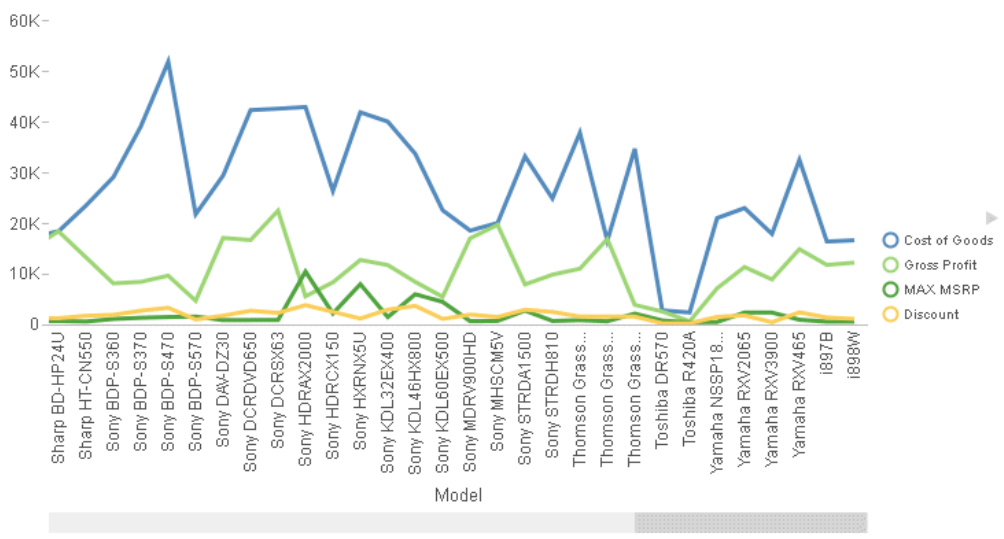
In the following request, the number of group entries is too large to display them all in the draw area legibly at one time:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MAX.MSRP_US DISCOUNT_US
BY MODEL
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=model, bucket=x-axis,$
type=data, column=COGS_US , bucket=y-axis,$
type=data, column=GROSS_PROFIT_US, bucket=y-axis,$
type=data, column=MAX.MSRP_US, bucket=y-axis,$
type=data, column=DISCOUNT_US, bucket=y-axis,$
ENDSTYLE
END
```

The output is shown in the following image. Not all the group entries are visible, but the scroll bar enables you to scroll to all group values:



The following image shows the scroll bar dragged to the last set of group entries:



Introduction to the WebFOCUS GRAPH FILE Command

GRAPH FILE request syntax is similar to TABLE FILE request syntax. To produce a chart instead of a tabular report, you need only substitute the command GRAPH for TABLE in the request. Thus, in some cases, you can produce charts by simply converting TABLE requests to GRAPH requests.

However, not every TABLE facility has a GRAPH counterpart, and there are some practical limitations on the amount of information that you can effectively display on a chart. When a TABLE request is converted in this manner, the phrases that make up the body of the request take on special meanings that determine the format and layout of the chart. The type of chart produced by a GRAPH FILE request depends on the display command used (SUM or PRINT), the sort phrases used (ACROSS, BY), and the LOOKGRAPH parameter setting.

For complete information about the GRAPH FILE command, see the *Creating Reports With WebFOCUS Language* manual.

Structure of a WebFOCUS HTML5 GRAPH Request

The following shows the structure of a GRAPH FILE request:

```
GRAPH FILE filename

[HEADING ...]
```

```
{PRINT|SUM} field ...
{BY|ACROSS} sortfield ...


[{WHERE|IF} expression ...[WHEN condition]] ...


[FOOTING ...]


ON GRAPH PCHOLD FORMAT JSCHART


[ON GRAPH SET LOOKGRAPH charttype]
[ON GRAPH SET optional_parameter ...]


ON GRAPH SET STYLE *
[*GRAPH_JS
 JSON properties
*END]
[*GRAPH_SCRIPT
 Legacy Graph API methods and properties
 *END]
[WebFOCUS StyleSheet declarations ]
[*GRAPH_JS_FINAL
 JSON properties to override server properties
*END]
ENDSTYLE
END
```

The following table describes each command :

Request Syntax	Description
`GRAPH FILE filename`	Required start of charting request.
`[HEADING ...]`	Optional title for the chart. By default, the heading is placed on the HTML page above the chart. To embed the heading in the chart, include the ON GRAPH SET EMBEDHEADING ON command in the request. With this setting, the heading replaces the JSON chart title object. If you also include a JSON title object in the request, it will override the WebFOCUS heading.

Request Syntax	Description
`{PRINT|SUM}` *`field`* `...`	Required display command. Fields must be numeric. They can be real fields, virtual fields, or calculated values. PRINT displays individual field values, SUM aggregates them within the sort values.
`{BY|ACROSS}` *`sortfield`* `...`	Required sort phrase.
`[{WHERE|IF}` *`expression`* `...[WHEN` *`condition`*`]] ...`	Optional filtering criteria.
`[FOOTING ...]`	Optional footing for the chart. By default, the footing is placed below the chart on the HTML page. To embed the footing in the chart, include the ON GRAPH SET EMBEDHEADING command in the request. With this setting, the footing replaces the JSON chart footnote object. If you also include a JSON footnote object in the request, it will override the WebFOCUS footing.
`ON GRAPH PCHOLD FORMAT JSCHART`	Required SET command for creating HTML5 output.

`[ON GRAPH SET LOOKGRAPH` *`charttype`*
 	
`ON GRAPH SET AUTOFIT ON`
`ON GRAPH SET GRAPHDEFAULT OFF`
`ON GRAPH SET VZERO OFF`
`ON GRAPH SET GRMERGE ADVANCED`
`ON GRAPH SET GRMULTIGRAPH 0`
`ON GRAPH SET GRLEGEND 0`
`ON GRAPH SET GRXAXIS 1]`
	Optional SET commands. For information about the optional SET commands, see *WebFOCUS Chart Parameters* on page 981 and the *Developing Reporting Applications* manual.

Request Syntax	Description
`ON GRAPH SET STYLE *`	Start of the style section of the request. The style section can include JSON blocks, legacy graph API blocks, and WebFOCUS StyleSheet declarations. They can appear in any order and there can be multiple blocks of each type. With conflicting properties, the last one in the style section of the request takes precedence.
`[*GRAPH_JS` *`JSON properties`* `*END]`	Start and end of an optional JSON block of the style section.
`[*GRAPH_SCRIPT` *`Legacy graph methods and properties`* `*END]`	Start and end of an optional legacy graph API block of the style section. These blocks are supported for backward compatibility, so that existing requests that contain these API calls can be converted to HTML5 charts without any changes except to include the ON GRAPH PCHOLD FORMAT JSCHART command.
`[`*`WebFOCUS StyleSheet declarations`*`]`	Chart attribute syntax declarations, as described in *WebFOCUS Chart Attribute Syntax* on page 143, and other optional WebFOCUS StyleSheet declarations. For information about StyleSheet declarations, see the *Creating Reports With WebFOCUS Language* manual.

Request Syntax	Description

`[*GRAPH_JS_FINAL`
` `*`JSON properties to override server properties`*
`*END]`
	Start and end of an optional JSON block of the style section that can override properties set by the server.
	
	The server sets certain properties, such as titles and tooltips, when using chart attribute syntax. Some of these properties can be customized in the WebFOCUS request. For example, column titles can be customized using an AS name in the request. If a property cannot be overridden with request syntax, you can add the properties to a *GRAPH_JS_FINAL block in the StyleSheet.

`ENDSTYLE`	End of the style section of the request.
`END`	Required end of the request.

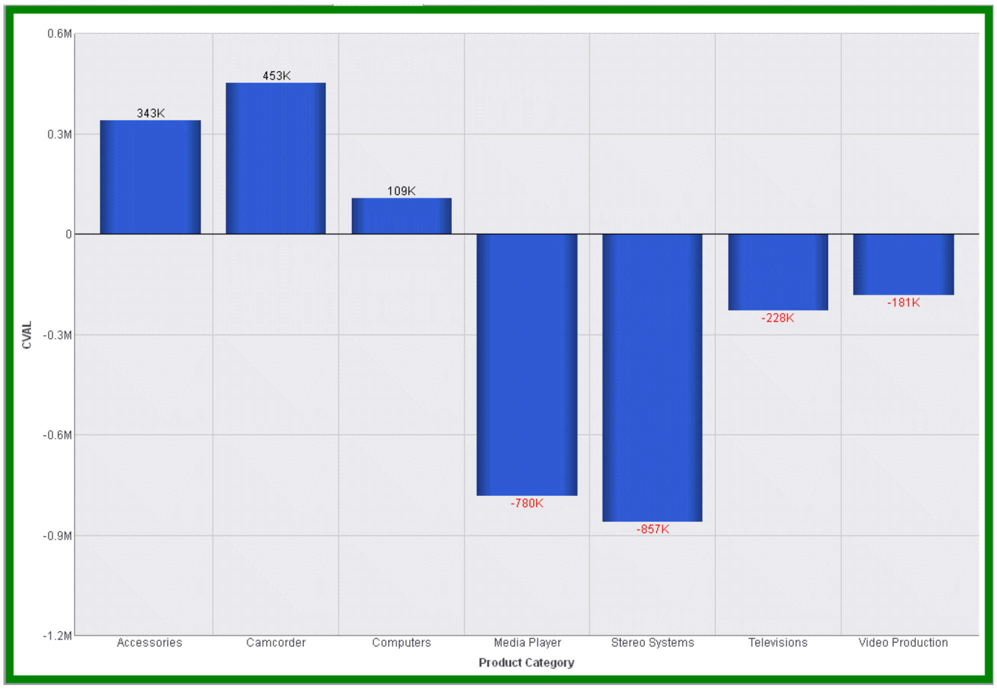
Example: Sample HTML5 GRAPH FILE Request

The following request creates a vertical bar chart (ON GRAPH SET LOOKGRAPH BAR). The y-axis represents the virtual field named CVAL, and the x-axis represents the sort field PRODUCT_CATEGORY:

```
-* Create virtual field
DEFINE FILE WF_RETAIL_LITE
CVAL/D12.2= IF PRODUCT_CATEGORY GT 'M' THEN -COGS_US ELSE COGS_US;
END
-*Define default values for amper variables referenced in the request
-DEFAULTH &WF_STYLE_UNITS='PIXELS';
-DEFAULTH &WF_STYLE_HEIGHT='405.0';
-DEFAULTH &WF_STYLE_WIDTH='770.0';
-* GRAPH request starts
GRAPH FILE WF_RETAIL_LITE
SUM CVAL
BY PRODUCT_CATEGORY
-*Required SET parameter for HTML5 output
ON GRAPH PCHOLD FORMAT JSCHART
-*Optional SET parameters
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET UNITS &WF_STYLE_UNITS
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
-*Style section starts
ON GRAPH SET STYLE *
-*WebFOCUS StyleSheet declarations start
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, PAGECOLOR = Green,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=X-AXIS,$
TYPE=DATA, COLUMN=CVAL, BUCKET=Y-AXIS,$
-*JSON block starts
*GRAPH_JS
"mouseOverIndicator": {
"enabled": true,
"color": " "
},
"introAnimation": {
"enabled": true,
"duration": 1000
}
*END


-*Legacy graph API block starts
*GRAPH_SCRIPT
setUseNegativeDataTextColor(true);
setDataTextDisplay(true);
*END
ENDSTYLE
END
```

The output is shown in the following image. On the output:

- The JSON block in the style section causes the drawing of the chart to be animated. It also causes a riser to change color and display a tooltip when you mouse over it,
- The WebFOCUS StyleSheet declarations assign the fields in the request to chart attribute categories, cause the chart styles to be taken from the included StyleSheet file, and cause the color of the page on which the chart is drawn to be green.
- The legacy graph API block causes the data text for negative risers to be displayed in red (which can also be done using JSON properties).



Similarities Between GRAPH and TABLE

The GRAPH request elements generally follow the same rules as their TABLE counterparts:

- The word FILE and the file name must immediately follow the GRAPH command, unless they were previously specified in a SET FILE command:

  ```
  SET FILE=filename
  ```

  You can specify any data source available to WebFOCUS, including joined or cross-referenced structures.

- You can concatenate unlike data source files in a GRAPH request with the MORE command.
- You can use the HEADING and FOOTING phrases to create headings and footings for the charts.
- You can use many WebFOCUS StyleSheet options for customizing the look of the chart, including conditional styling. For complete information about WebFOCUS StyleSheet options, see the *Creating Reports With WebFOCUS Language* manual.
- The order of the phrases in the request does not affect the format of the chart. For example, the selection phrase may follow or precede the display command and sort phrases. The order of sort phrases does affect the format of a chart, just as the order of sort phrases in TABLE requests affects the appearance of reports.
- The word END must be on a line by itself to complete a request.
- All dates are displayed in MDY format unless they are changed to alphanumeric fields.

Differences Between GRAPH and TABLE

There are a few notable syntactical differences between TABLE and GRAPH. Specifically, the following restrictions apply:

- A GRAPH request must contain a display command with at least one display field. It almost always requires at least one sort phrase (BY or ACROSS), in order to generate a meaningful chart.
- In GRAPH requests, the object of the display command must always be a numeric field (measure).
- Several BY phrases can be used in a request, in which case multiple graphs are created. A single ACROSS phrase is allowed in a GRAPH request, and requests for certain chart forms can contain both ACROSS and BY phrases.
- The number of ACROSS values cannot exceed 64.
- The RUN option is not available as an alternative to END.

Generating Sample Files

The examples in this manual use the WF_RETAIL_LITE Master File, which is part of the WF_RETAIL sample. To generate these sample files:

1. Go to the Reporting Server Web Console.

2. On the Applications page, click the *New* button and click *Tutorials*, or right-click an application folder, point to *New*, and click *Tutorials* from the context menu.

   This opens the *Create Tutorial Framework* page.

3. Select *WebFOCUS - Retail Demo* from the *Tutorial* drop-down list.

   The files can be generated as flat files, if you do not have a relational DBMS configured, or as relational tables, if you do have a relational DBMS configured. If you choose to create relational tables, you may have to create a database in the relational DBMS to receive the sample files prior to generating them.

4. If you do not have a relational adapter configured, select DATREC as the DBMS. If you have a relational adapter configured, select a relational DBMS and connection.

5. Enter a prefix to apply to the name of each table created in the DBMS, or accept the default that is already entered.

6. You can select *Medium* from the *Tutorial Data Volume Limit* drop-down list in order to avoid generating massive amounts of data.

   Note: The output for the examples in this manual were generated using the Medium volume limit.

7. Either enter an application name, accept the default already entered, or click ... to select an application from the Application tree.

8. Click *Create*.

The sample files will be created.

Note: In some of the sample requests in this manual, the page width may cause a string to be separated onto multiple lines. To run the request, you must remove any line breaks within strings. That is, make sure that the beginning quotation mark and ending quotation mark that enclose the string are on the same line in the request.

Chapter 2

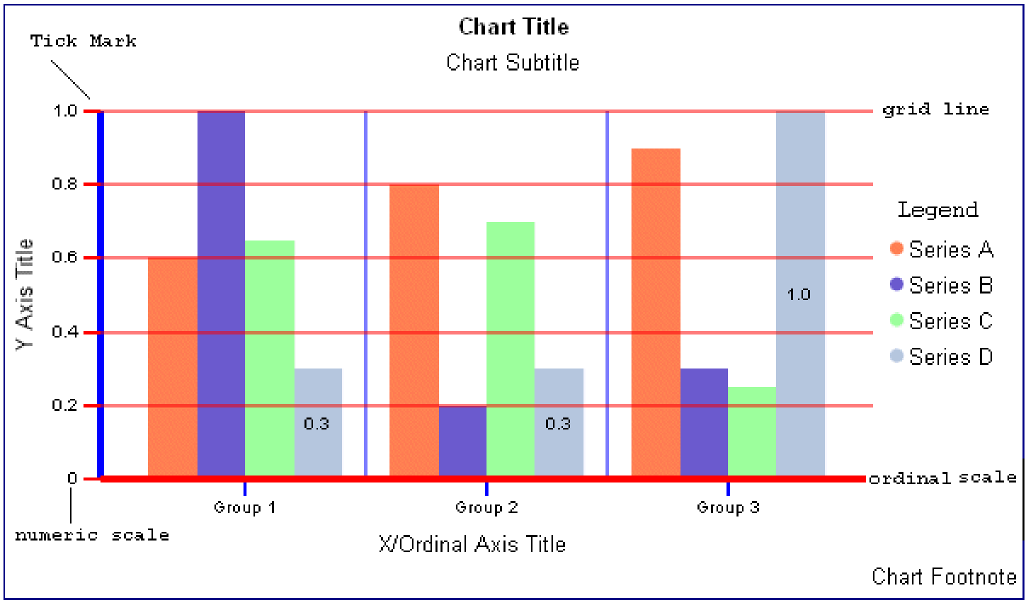
Understanding Chart Types

This chapter describes the chart types available for creating HTML5 charts and how to generate them.

In this chapter:

Chart Components

The following diagram shows the components of a chart.



With a few exceptions, each series represents the set of values for one field in the request, and each group represents all series values related to one sort field value.

Controlling the Chart Type

The recommended way to establish the chart type is to include the LOOKGRAPH parameter in your request.

WebFOCUS provides parameters for controlling many other chart properties, as well. Most of these also have JSON equivalents. In all cases, the best practice is to use WebFOCUS syntax where possible and JSON properties when a WebFOCUS parameter does not exist for the desired property.

Controlling Chart Type Using LOOKGRAPH

With a few exceptions, each GRAPH request must include a sort phrase (dimension) and at least one display field (measure).

The fields that are the subjects of the chart can be real or virtual fields, with or without direct operator prefixes (for example, AVE., MIN., MAX.). They can also be calculated values.

Note: Display fields used only for calculations need not appear in the chart. You can use the NOPRINT phrase to suppress the display of such fields.

The LOOKGRAPH parameter enables you to change the format of the graph without having to set individual control parameters or restructure the graph request. However, even if you use LOOKGRAPH, you can choose to set individual control parameters (for example, SET GRID=ON).

Note: There are two types of syntax for generating HTML5 charts, traditional syntax and chart attribute syntax. In some cases, they use different LOOKGRAPH values. This chapter describes the LOOKGRAPH values for the traditional syntax. For information about chart types and LOOKGRAPH values for chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

Syntax: How to Specify a Chart Type Using LOOKGRAPH

Prior to the request, use the following syntax

```
SET LOOKGRAPH= type
```

Or, within the request, use the following syntax

```
ON GRAPH SET LOOKGRAPH type
```

where:

type

Is the chart type you want to generate.

For a complete list of LOOKGRAPH values for the traditional chart syntax, see the *Creating Reports With WebFOCUS Language* manual. The following sections list the most useful LOOKGRAPH values for creating HTML5 charts.

Reference: Line Chart Styles

The following are LOOKGRAPH values for the most useful types of connected point plots:

SET LOOKGRAPH	Description
`VLINE` `LINE`	A vertical absolute connected point plot graph.
`VLINE2`	A vertical connected point plot graph with two axes.
`HLINE`	A horizontal absolute connected point plot graph.
`HLINE2`	A horizontal connected point plot graph with two axes.

Reference: Bar Chart Styles

The following are LOOKGRAPH values for the most useful types of bar charts:

SET LOOKGRAPH	Description
`VBAR` `BAR`	A bar graph with absolute vertical side-by-side bars (columns).
`HBAR`	An absolute bar graph with horizontal bars.
`VBRSTK1` `STACK`	A stacked vertical bar graph.
`VBRSTKPC`	A stacked vertical bar graph that shows percentages.
`HBRSTK1`	A stacked horizontal bar graph.

SET LOOKGRAPH	Description
HBRSTKPC	A stacked horizontal bar graph that shows percentages.

Reference: Pie Chart Styles

The following is the LOOKGRAPH value for the most useful type of pie chart:

SET LOOKGRAPH	Description
PIE	A pie graph.

Reference: Scatter Chart Styles

The following are LOOKGRAPH values for the most useful types of scatter chart:

SET LOOKGRAPH	Description
SCATTER	Produces a scatter graph. The numeric sort field becomes the x-axis.
SCATTERN	Produces a scatter graph using a sort field of any data type. The measures (there must be two) become the axes, and the points plotted represent the sort field values.
SCATTERS	Produces a scatter graph using a sort field of any data type. Each marker represents a value of one of the measures (there can be any number), and the sort field becomes the x-axis.

Reference: Three-Dimensional Chart Styles

The following are LOOKGRAPH values for the most useful types of three-dimensional charts:

SET LOOKGRAPH	Description
3DAREAG	A three-dimensional connected group area chart.

SET LOOKGRAPH	Description
3D_BAR	A three-dimensional chart with bars.
3DSURFCE	A three-dimensional surface chart that graphs all data points as a three-dimensional surface, like a rolling wave.

Reference: Area Chart Styles

Choose one of the following LOOKGRAPH values to change the style of area charts:

SET LOOKGRAPH	Description
VAREA	A vertical area graph.
VAREASTK	A stacked vertical area graph.
VAREAR2	A vertical area graph with two axes.
VARESTK2	A stacked vertical area graph with two axes.
VARESTKP	A stacked vertical area graph that shows percentages.
HAREA	A horizontal area graph.
HAREAR2	A horizontal area graph with two axes.
HAREASTK	A stacked horizontal area graph.
HARESTK2	A stacked horizontal area graph with two axes.
HARESTKP	A stacked horizontal area graph that shows percentages.

Reference: Polar Chart Styles

Choose one of the following LOOKGRAPH values to change the style of polar charts:

SET LOOKGRAPH	Description
`POLAR`	A polar chart that displays data points on a circle.
`POLAR2`	A dual polar chart. Values from an additional data set are displayed on a second value (Y) axis.

Reference: Radar Chart Styles

Choose one of the following LOOKGRAPH values to change the style of radar charts:

SET LOOKGRAPH	Description
`RADARA`	A radar area chart.
`RADARL`	A radar line chart.
`RADARL2`	A dual radar line chart. Values from an additional data set are displayed on a second value (Y) axis.

Reference: Bubble Chart Styles

Choose one of the following LOOKGRAPH values to change the style of bubble charts:

SET LOOKGRAPH	Description
`BUBBLE`	A bubble chart.
`BUBBLED`	A bubble chart with a dual axis.
`BUBBLEDL`	A bubble chart with a dual axis and labels.
`BUBBLEL`	A bubble chart with labels.

Reference: Spectral Chart Styles

Choose the following LOOKGRAPH value to generate a spectral chart (or heatmap):

SET LOOKGRAPH	Description
`SPECTRAL`	A spectral map chart. This is a chart with a row or column matrix of markers that is colored according to the data values.

Reference: Other Chart Types

Use the following LOOKGRAPH values for these advanced chart types:

SET LOOKGRAPH	Description
`VWATERFL`	Vertical waterfall chart. A waterfall chart does not require a sort field.
`HWATERFL`	Horizontal waterfall chart. A waterfall chart does not require a sort field.
`PARETO`	Displays data following Pareto 80:20 rule. Pareto charts require only one display field.

`MULTI3Y`
`MULTI4Y`
`MULTI5Y`
	Stacks charts in order to make it easier to read, analyze and manage them.

Reference: HTML5-Only Chart Types

The following LOOKGRAPH values are valid only when generating an HTML5 chart:

SET LOOKGRAPH	Description
`BUBBLEMAP`	A bubblemap is a chart in which proportionally sized bubbles are displayed on relevant areas of the map.
`CHOROPLETH`	A choropleth is a chart in which areas on a map are shaded or patterned in proportion to the value of the measure being represented.

SET LOOKGRAPH	Description
`MEKKO`	A Mekko chart is a variant of a stacked bar chart, in which the width of the bars is adjusted relative to its value in the data set.
`PARABOX`	A Parabox (or parallel coordinates chart) is similar to a regular line chart, except that each group in the line chart has a unique and interactive numeric axis. Each line represents one series of data. Each vertical bar represents a numeric axis. You can click and drag along each of the axes to select (filter) the lines that pass through that part of the axis
`STREAM`	A streamgraph is a simplified version of a stacked area chart. In a streamgraph, there are no axes, gridlines, or frames. The baseline is free, which makes it easier to perceive the thickness of any given layer across the data.
`TAGCLOUD`	A tagcloud is a visual representation of frequency. It displays only group labels. The size of each label is proportional to its data value.
`TREEMAP`	A treemap chart displays hierarchical data as a set of nested rectangles.

Chart Type Notes and Sample Charts

This section shows simple WebFOCUS requests using traditional WebFOCUS chart syntax that create different chart types. Subsequent chapters describe the JSON methods and properties available for each type of chart. For chart types and sample charts using chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

Line Charts

Line charts are useful for emphasizing the movement or trend of numeric data over time, since they allow a viewer to trace the evolution of a particular point by working backwards or interpolating. Highs and lows, rapid or slow movement, or a tendency toward stability are all types of trends that are well suited to a line chart.

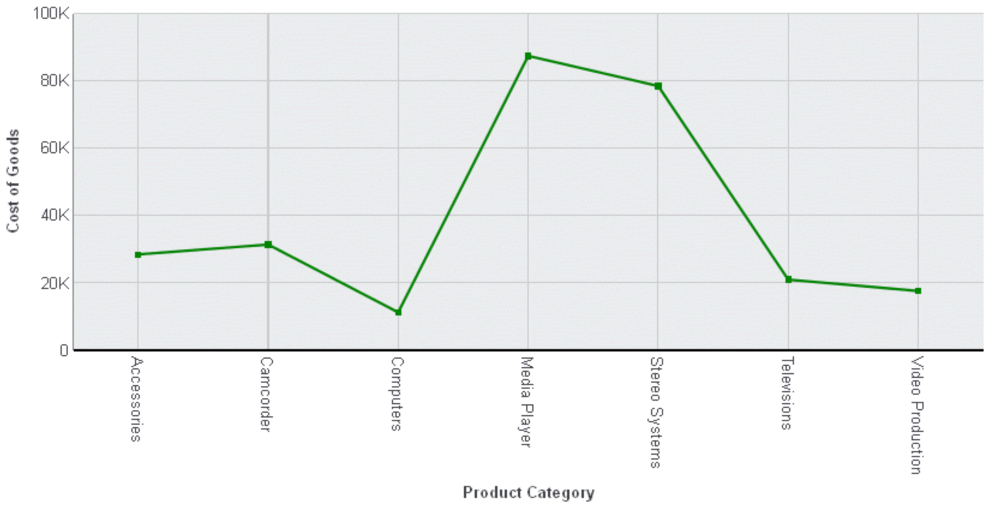
Line charts can also be plotted with two or more scales to suggest a comparison of the same value, or set of values, in different time periods. The number of scales your chart has depends on the type of chart you select.

Example: **Sample Line Chart**

The following request creates a line chart (ON GRAPH SET LOOKGRAPH LINE). The y-axis represents cost of goods sold (COGS_US), and the x-axis represents the product category. The JSON block of the style section sets the series color to green:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"series": [{"series": 0, "color": "green"}]
*END
ENDSTYLE
END
```

Since there is only one display field in the request, it represents series 0 and it is displayed in green:



For more information about setting color properties, see *Introduction to JSON Properties for HTML5 Charts* on page 83.

Bar Charts

A bar chart plots numeric data by displaying rectangular blocks against a scale. The length of a bar corresponds to a value or amount. Viewers can develop a clear mental image of comparisons among data series by distinguishing the relative heights of the bars. Use a bar chart to display numeric data when you want to present comparisons of data.

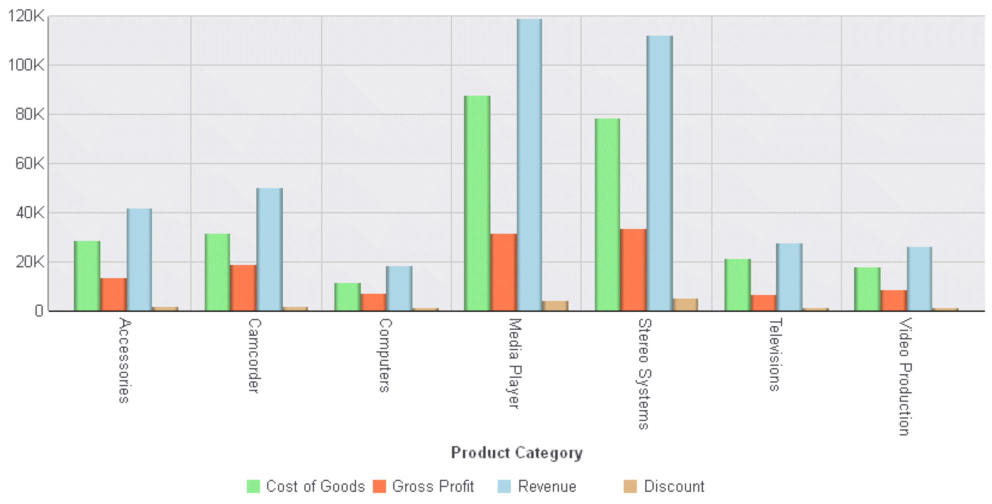
WebFOCUS supports a variety of bar chart styles. For a list of bar chart styles available, see *Bar Chart Styles* on page 41 and *Three-Dimensional Chart Styles* on page 42.

Example: Sample Bar Chart

The following request creates a vertical bar chart (ON GRAPH SET LOOKGRAPH VBAR). The JSON block in the style section of the request sets the series colors:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:

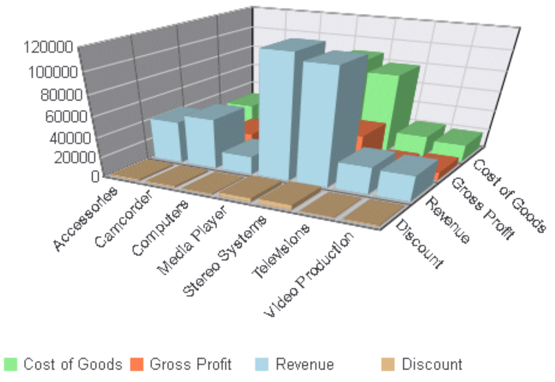


Example: Sample 3D Bar Chart

The following request creates a 3D bar chart (ON GRAPH SET LOOKGRAPH 3D_BAR). The JSON block in the style section of the request sets the series colors:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3D_BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Pie Charts

A pie chart emphasizes where your data fits in relation to a larger whole. Each slice represents a percentage of the whole. Keep in mind that pie charts work best when your data falls into a limited number of groups. Too many groups divide the pie into small segments that are difficult to see. Use color or texture on individual segments to create visual contrast.

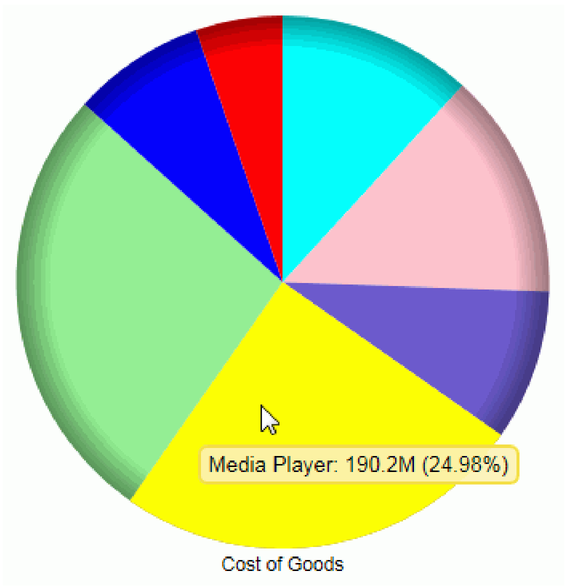
Note: If a pie chart request that does not use chart attribute syntax has multiple BY fields and multiple measures, only the last measure referenced in the request is represented on the legend, and the tooltip information is incorrect.

Example: Sample Pie Chart

The following request creates a pie chart (ON GRAPH SET LOOKGRAPH PIE). Each slice of the pie shows the percentage of cost of goods sold contributed by each product category. The JSON block in the style section of the request sets the slice colors:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visble": false},
"mouseOverIndicator": {
    "enabled": true, "color": "yellow"},
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "pink"},
    {"series": 2, "color": "slateblue"},
    {"series": 3, "color": "burlywood"},
    {"series": 4, "color": "lightgreen"},
    {"series": 5, "color": "blue"},
    {"series": 6, "color": "red"}
    ]
*END
ENDSTYLE
END
```

The output shows the default labels displayed when the mouse hovers over a pie slice:



Scatter Charts

Scatter charts show the relationship between two different numeric measures. Use a scatter plot to visualize the density of individual data values around particular points or to demonstrate patterns in your data.

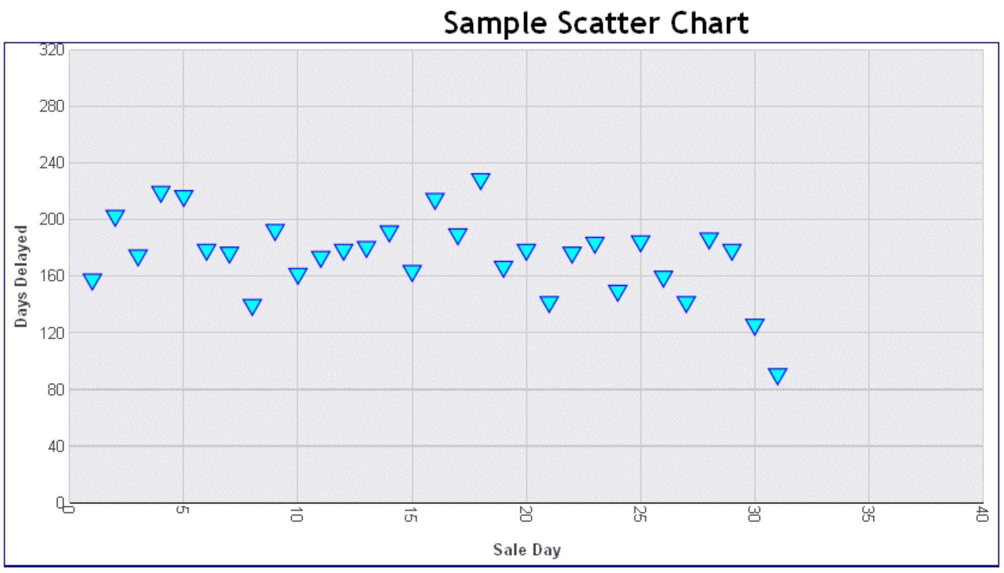
A scatter chart has a numeric x-axis, or sort field. Scatter charts and line charts are distinguishable from one another only by virtue of their x-axis data type. Line charts can appear without connecting lines (making them look like scatter charts) and scatter charts can appear with connecting lines (making them look like line charts).

Example: Sample Scatter Chart

The following request creates a scatter chart (ON GRAPH SET LOOKGRAPH SCATTER). The HEADING phrase in the request creates a heading for the chart. The JSON block in the style section of the request sets the chart border color and the marker shape, border, and color:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"Sample Scatter Chart"
SUM DAYSDELAYED
ACROSS TIME_DAY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"color": "navy"},
"series": [{"series": 0, "color": "cyan",
    "marker": {"shape": "triangle", "size": 12,
    "border": {"width": 1, "color": "blue"}}}
    ]
*END
ENDSTYLE
END
```

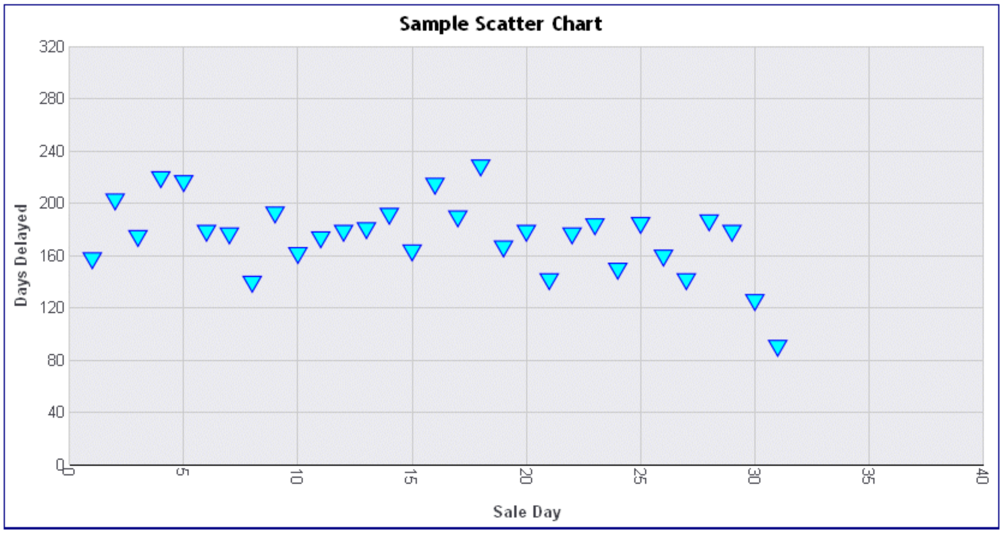
The output is:



Note that the heading is positioned on the HTML page outside of the chart frame. It is also centered on the HTML page. If you want to place the heading inside of the chart and have it centered on the chart, add the ON GRAPH SET EMBEDHEADING ON command to your request:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"Sample Scatter Chart"
SUM DAYSDELAYED
ACROSS TIME_DAY
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"color": "navy"},
"series": [{"series": 0, "color": "cyan",
    "marker": {"shape": "triangle", "size": 12,
    "border": {"width": 1, "color": "blue"}}}
    ]
*END
ENDSTYLE
END
```

The output is:



Area Charts

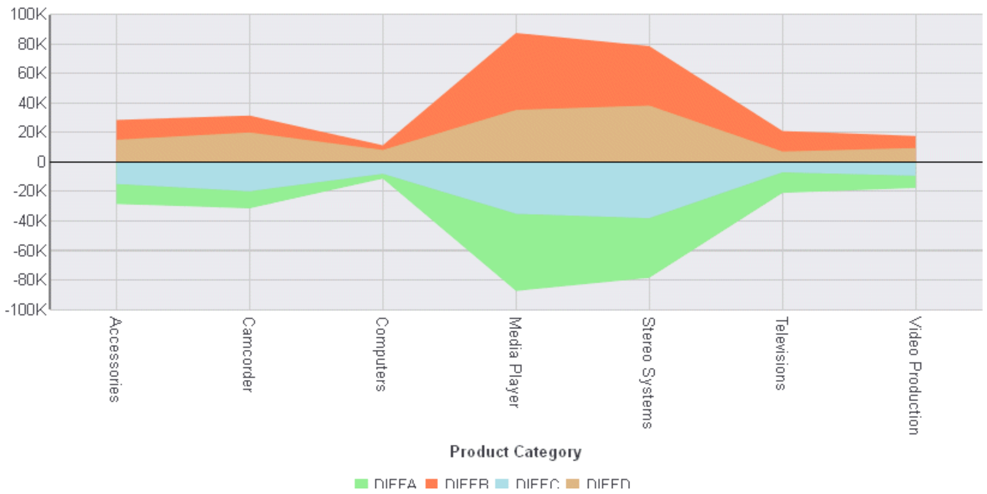
Area charts are similar to line charts, except that the area between the data line and the zero line (or axis) is usually colored or textured. Area charts allow you to stack data on top of each other. Stacking allows you to highlight the relationship between data series, showing how some data series approach or shadow a second series.

Example: Sample Area Chart

The following request creates four virtual fields and then uses them to produce a vertical area chart (ON GRAPH SET LOOKGRAPH VAREA). The JSON block in the style section of the request sets the series colors:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - MSRP_US;
DIFFD = MSRP_US - COGS_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VAREA
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:

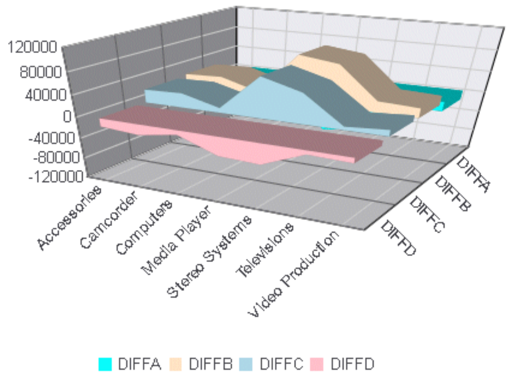


Example: **Sample 3D Area Chart**

The following request creates four virtual fields and uses them to produce a 3D area chart (ON GRAPH SET LOOKGRAPH 3DAREAS). The JSON block in the style section of the request sets the legend position and the series colors:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - (COGS_US * DISCOUNT_US)/100;
DIFFD = COGS_US - MSRP_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DAREAS
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true, "position": "bottom"},
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "bisque"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "pink"}
    ]
*END
ENDSTYLE
END
```

The output is:



Boxplots

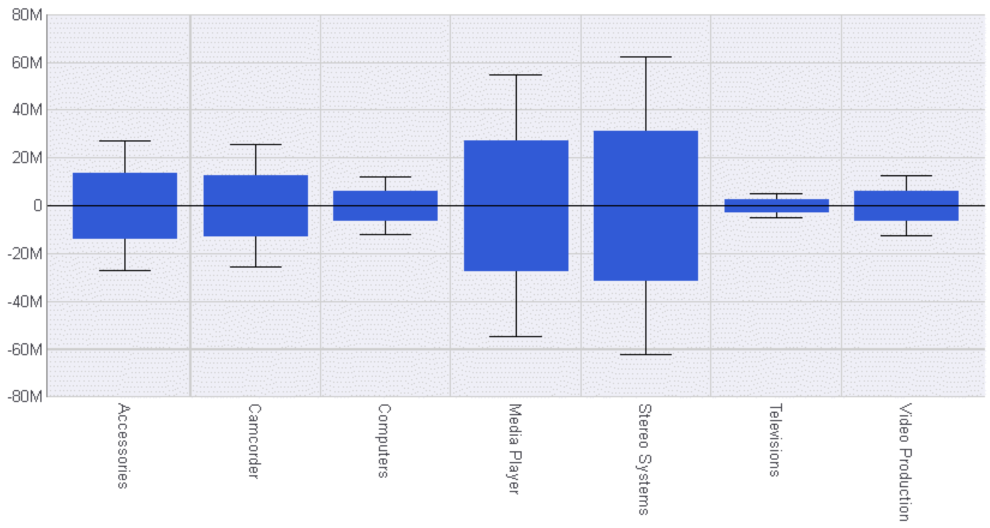
In a boxplot, each series and group requires five values. For a given series and group box, the first value is the minimum (lower *hat*), the second defines the box bottom, the third value defines the median line, the fourth value defines the box top, and the fifth value defines the location of the top hat. Values must be in ascending order. Box plot fill color and border are defined by the series color and border.

Example: Sample Boxplot

The following request calculates the five values needed for a boxplot and then generates a boxplot (ON GRAPH SET LOOKGRAPH BOXPLOT). The JSON block in the style section of the request sets the series colors and border width.

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US -100000;
DIFF2 = COGS_US -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"Series": [
    {"series": 0, "color": "lightblue", "border": {"width": 1}}
    ]
*END
ENDSTYLE
END
```

The output is:



Bubble Charts

A bubble chart is an enhanced scatter plot in which the size of each marker is proportional to the value of a third measure. Therefore, a bubble chart requires three values, (x-position, y-position, and size) to draw each bubble marker.

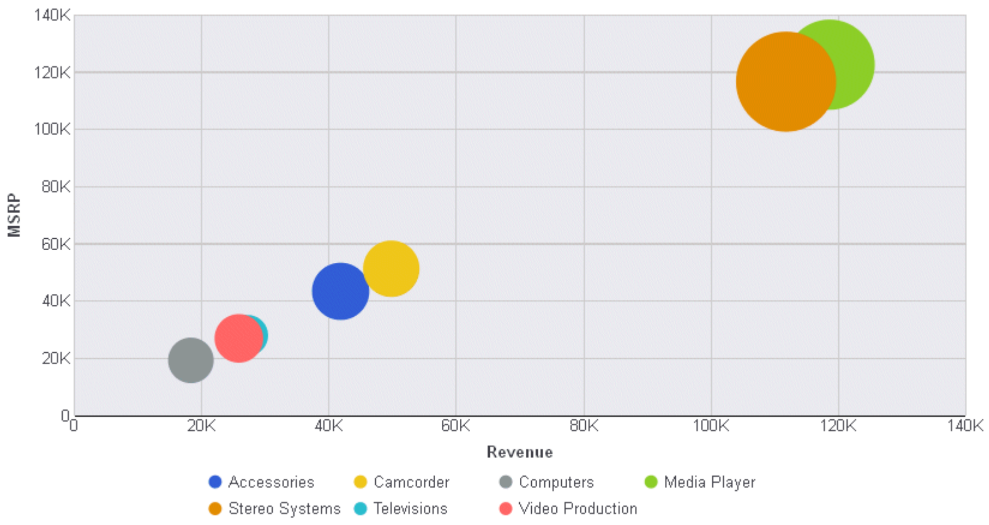
A negative size value is treated as positive (that is, it uses the absolute value of the data). A null, undefined, or zero size will eliminate the marker. Sizes are proportional according to the range of size values in the data set. Disparate marker size values will draw a large marker that exceeds the draw area.

Example: Sample Bubble Chart

The following request generates a bubble chart (ON GRAPH SET LOOKGRAPH BUBBLE). The chart uses the included StyleSheet and default settings, so the style section of the request does not include a JSON block.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```

On the output, the bubble size depends on the discount, while the axes show the manufacturer suggested retail price and revenue:



Funnel Charts

A funnel chart is basically a pie chart that shows only one group of data at a time. The series in the group are stacked in the funnel with the first series at the top and the last series at the bottom. In the funnel chart, the display field functions as the group, and the sort fields function as the series.

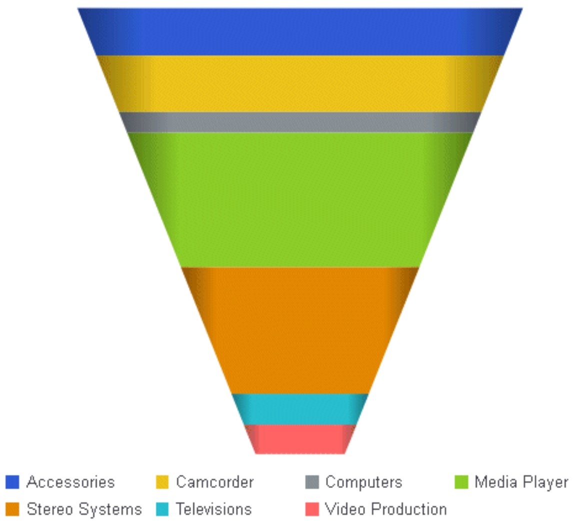
Series-specific properties control the color of funnel segments.

Example: Sample Funnel Chart

The following request generates a funnel chart (ON GRAPH SET LOOKGRAPH FUNNEL). The output uses the included StyleSheet file and accepts all other defaults, so there is no JSON block in the style section:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```

The output is:



Gauge Charts

Gauges identify a single value along an axis that is usually displayed in a circle. They are often used in dashboards to display progress or quantity.

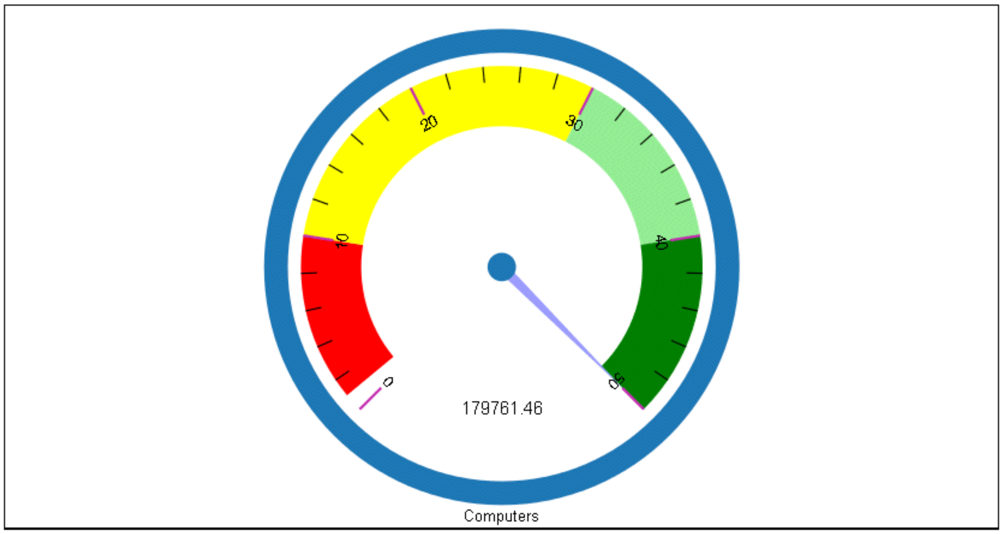
The numeric y-axis properties control scaling, tick marks, and colors assigned to segments of the gauge axis.

Example: Sample Gauge Chart

The following request generates a gauge chart (ON GRAPH SET LOOKGRAPH GAUGE1). The JSON block in the style section of the request sets the color scale and major grid line style:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"yaxis": {"min": 0, "max": 50,
"colorBands": [
    {"start": 1, "stop": 10, "color": "red"},
    {"start": 10, "stop": 30, "color": "yellow"},
    {"start": 30, "stop": 40, "color": "lightgreen"},
    {"start": 40, "stop": 50, "color": "green"}],
"majorGrid": {
    "visible": true, "lineStyle": {"width": 2, "color": "rgb(196,48,178)"}}}
*END
ENDSTYLE
END
```

The output is:



Heatmaps

Heatmaps (or spectral charts) contain a row or column matrix of rectangles that are displayed in different colors depending on the data values. They use the same kind of data as bar, line, and area charts (an array of arrays). Each internal array forms a row in the heatmap table.

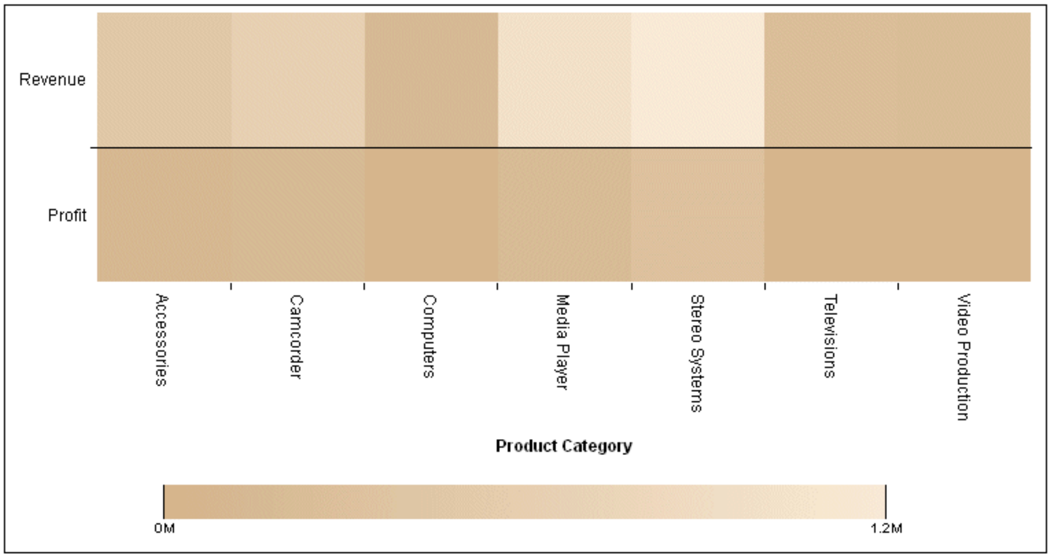
Series labels are plotted on the left edge according to settings in z-axis properties. Group labels are plotted on the bottom edge according to x-axis properties. The colorScale property controls the color of the cells in the heatmap table.

Example: **Sample Heatmap**

The following request generates a heatmap (ON GRAPH SET LOOKGRAPH SPECTRAL). The JSON block in the style section of the request sets the color scale.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US AS 'Revenue'
GROSS_PROFIT_US AS 'Profit'
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SPECTRAL
ON GRAPH SET STYLE *
*GRAPH_JS
"colorScale": {"colors": ["tan", "antiquewhite"]}
*END
ENDSTYLE
END
```

The output is:



Histograms

A histogram is a graphical representation that visually depicts the distribution of data. The sort field for a histogram is numeric, while for bar chart, it is usually categorical.

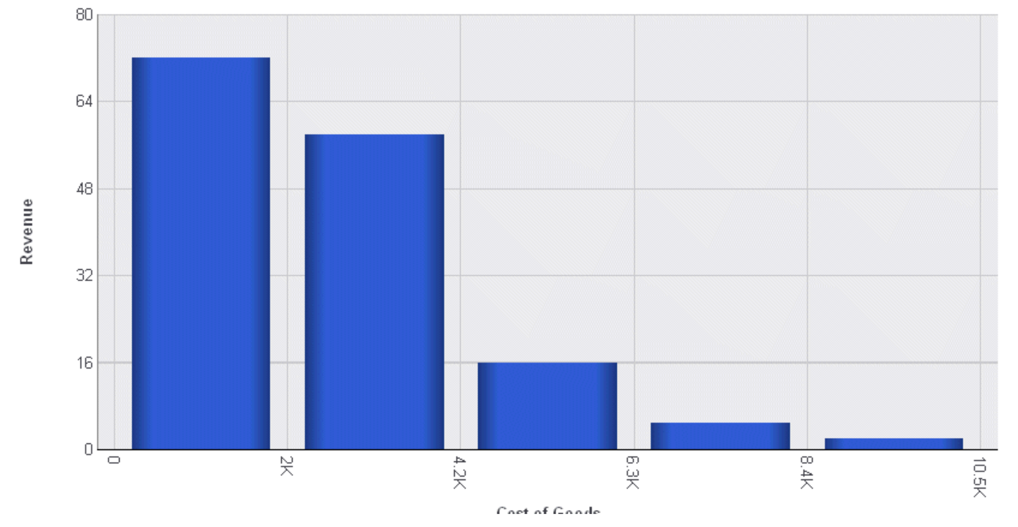
In a vertical histogram, the y-axis is on the left side of the chart and the x-axis is drawn on the bottom of the chart. In a horizontal histogram, the x-axis is on the left side of the chart and the y-axis axis is drawn on the bottom of the chart.

Example: Sample Histogram

The following request generates a vertical histogram (ON GRAPH SET LOOKGRAPH VHISTOGR). The JSON block in the style section of the request sets the number of x-axis groups to use and makes the legend not visible.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VHISTOGR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"histogramProperties": {"binCount": 5},
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is:



Mekko Charts

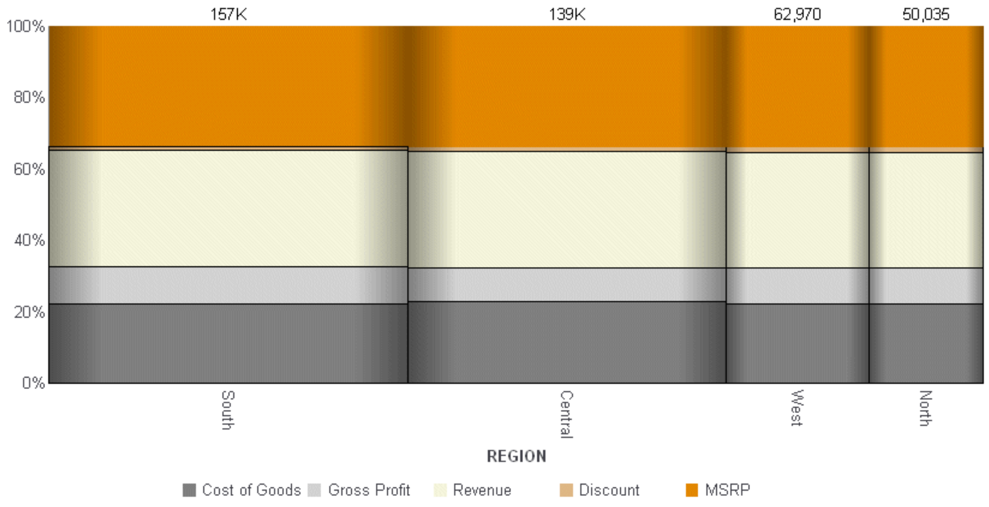
A mekko (also called marimekko) chart is a percent bar chart, except that the width of each bar riser is based on the overall value of the stack. You can use any data set that works for a regular or stacked bar chart.

Example: Sample Mekko Chart

The following request creates a mekko chart (ON GRAPH SET LOOKGRAPH MEKKO). The virtual field REGION groups the data into a reasonable number of categories. The JSON block in the style section of the request sets the series colors and borders:

```
DEFINE FILE WF_RETAIL_LITE
REGION/A12 = IF STATE_PROV_CODE_ISO_3166_2 EQ 'CT' OR 'MA' OR 'ME'
                     OR 'NH' OR 'RI' OR 'VT'
                     OR 'NJ' OR 'NY' OR 'PA' THEN 'North'
             ELSE IF STATE_PROV_CODE_ISO_3166_2 EQ 'IA' OR 'ID'
                     OR 'IN' OR 'IL' OR 'OK'
                     OR 'KS' OR 'OH' OR 'MI' OR 'MN'
                     OR 'OK' OR 'MO' OR 'MS' OR 'ND'
                     OR 'NE' OR 'SD' OR 'WI' THEN 'Central'
             ELSE IF STATE_PROV_CODE_ISO_3166_2 EQ 'AL' OR 'DC'
                     OR 'DE' OR 'FL' OR 'GA' OR 'KY'
                     OR 'LA' OR 'MD' OR 'NC' OR 'SC'
                     OR 'TN' OR 'TX' OR 'VA' OR 'WV' THEN 'South'
             ELSE IF STATE_PROV_CODE_ISO_3166_2 EQ 'CA' OR 'OR'
                     OR 'WA' OR 'AR' OR 'AZ' OR 'CO'
                     OR 'MN' OR 'MT' OR 'ID' OR 'NM'
                     OR 'WY' OR 'NV' OR 'UT' OR 'WY' THEN 'West';
END

GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US MSRP_US
BY REGION
WHERE COUNTRY_NAME EQ 'United States'
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH MEKKO
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"series": [
{"series": 0, "color": "grey", "border": {"width": 1, "color": "black"}},
{"series": 1, "color": "lightgrey", "border": {"width": 1, "color": "black"}},
{"series": 2, "color": "beige", "border": {"width": 1, "color": "black"}},
{"series": 3, "color": "burlywood", "border": {"width": 1, "color": "black"}}
]
*END
ENDSTYLE
END
```

The output is:



Parabox Charts

Parallel coordinates is a popular method of visualizing high-dimensional data using dynamic queries. The parabox chart type is similar to a regular line chart, except that each group in the line chart has a unique and interactive numeric axis.

Each colored line represents one series of data. Each vertical bar represents a numeric axis. You can click first, then you can drag along each of the axes to select (filter) the lines that pass through that part of the axis.

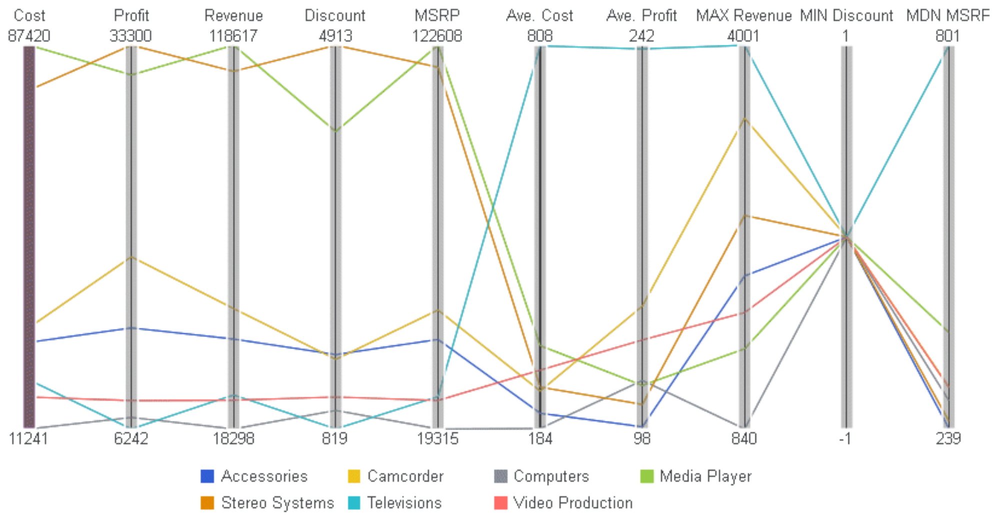
FORMAT JSCHART also supports categorical (alphanumeric) vertical axes. These are drawn with a bubble marker on the vertical axis. The size of the bubble corresponds to the number of lines passing through it. Whether an axis is numeric or categorical is defined by the first series of data. If a value in the first array is a number, that axis is numeric. If a value in the first array is a string, that axis is categorical. Categorical axes are sorted automatically, such that the biggest bubble is on top and bubble size descends from there.

Example: Sample Parabox Chart

The following request creates a parabox chart (ON GRAPH PCHOLD FORMAT PARABOX). The JSON block of the style section makes the legend visible:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US AS 'Cost'
GROSS_PROFIT_US AS 'Profit'
REVENUE_US DISCOUNT_US MSRP_US
AVE.COGS_US AS 'Ave. Cost'
AVE.GROSS_PROFIT_US AS 'Ave. Profit'
MAX.REVENUE_US
MIN.DISCOUNT_US
MDN.MSRP_US AS 'MDN MSRP'
BY PRODUCT_CATEGORY
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PARABOX
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true}
*END
ENDSTYLE
END
```

The output is:



Pareto Charts

A pareto chart is similar to a combo chart. The first series (series zero) draws as an absolute bar chart on the y-axis. The second series (series one, derived using the same field as series 0) draws as a cumulative percent line on the y2-axis. This enables you to see not only the amount contributed by each group, but also which groups contribute the highest percentage to the total.

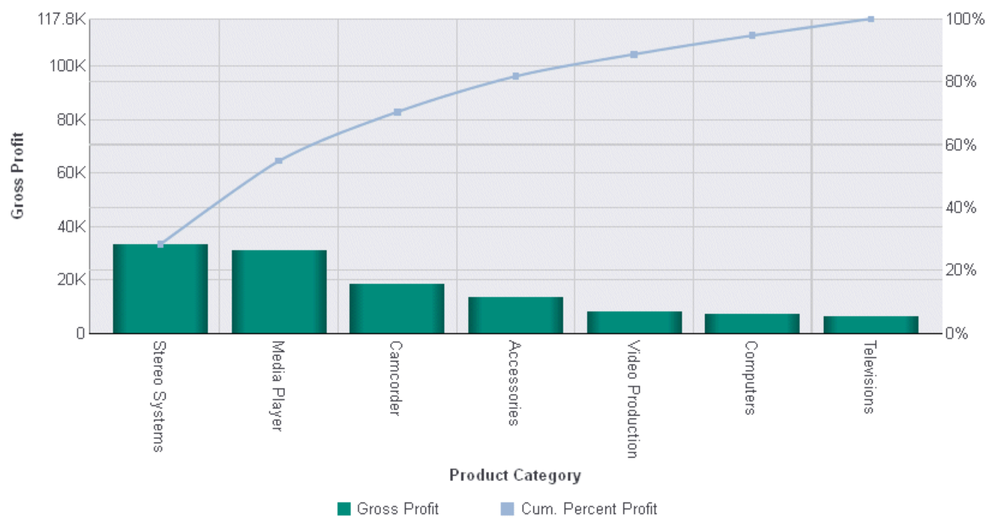
Note: The Y1 axis maximum value in a Pareto chart is set as the total of all of the risers. We recommend that you not customize the maximum y-axis value. It should be left as Automatic (the default setting).

Example: Sample Pareto Chart

The following request creates a pareto chart (ON GRAPH SET LOOKGRAPH PARETO). Both the bars and the line are generated from the single display field, GROSS_PROFIT_US. The JSON block in the style section sets the series colors and marker shapes and labels:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PARETO
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"mouseOverIndicator": {
    "enabled": true, "color": " "},
"legend": {"visible": true},
"blaProperties": {"orientation": "vertical", "lineConnection": "curved"},
"series": [
    {"series": 0, "color": "rgb(0,142,126)", "marker": {"visible": true}},
    {"series": 1, "color": "rgb(152,181,211)", "marker": {"visible": true},
            "label": "Cum. Percent Profit"}
]
*END
ENDSTYLE
END
```

The output is:



Polar Charts

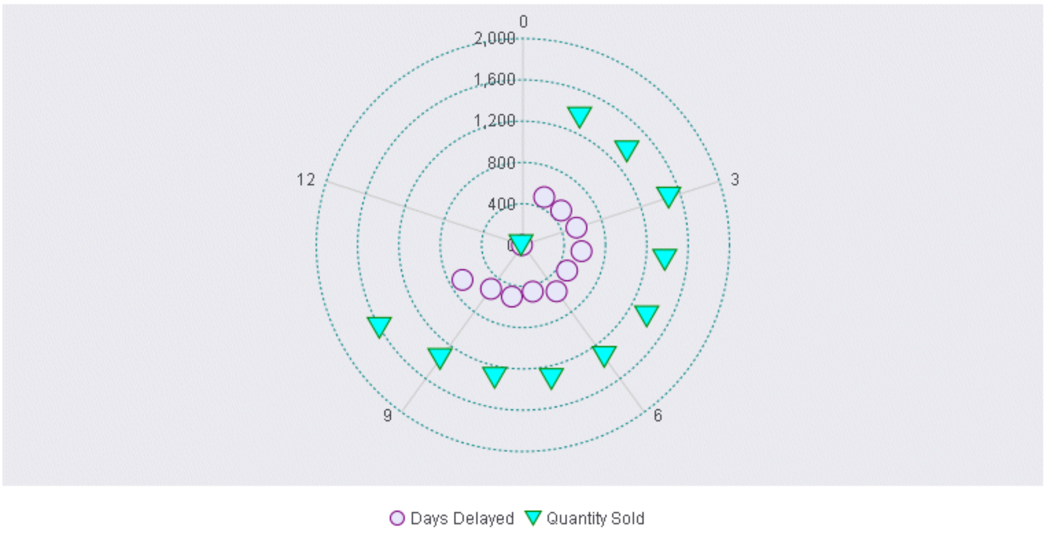
A polar chart is a circular scatter chart. Like scatter charts, a polar chart requires two values to draw each marker.

Example: Sample Polar Chart

The following request creates a polar chart (ON GRAPH SET LOOKGRAPH POLAR). The JSON block in the style section sets the series colors and line styles for the major gridlines of the y-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
ACROSS TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH POLAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"yaxis": {"majorGrid": {"lineStyle": {"width": 1, "color": "teal", "dash": "2 2"}}},
"series": [
{"series": 0, "color": "lavender", "marker": {"size": 15, "shape": "circle",
    "border": {"width": 1,"color": "purple"}}},
{"series": 1, "color": "cyan", "marker": {"size": 15, "shape": "triangle",
    "border": {"width": 1, "color": "green"}}}
]
*END
ENDSTYLE
END
```

The output is:



Radar Charts

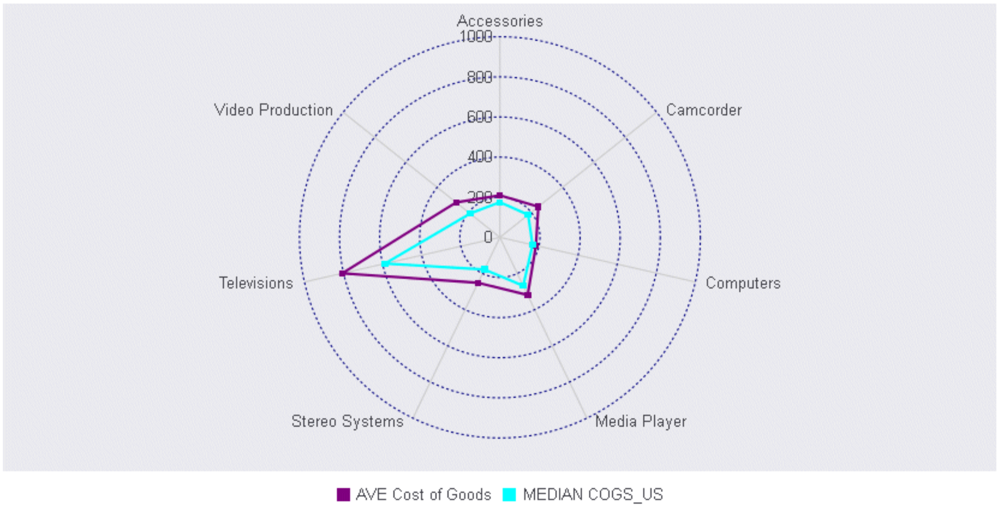
A radar chart is a circular line chart. Radar charts require one value for each line segment (and marker if shown).

Example: Sample Radar Chart

The following request creates a radar chart (ON GRAPH SET LOOKGRAPH RADARL). The JSON block in the style section sets the color and style for the y-axis major grid lines, and sets the series colors and borders:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH RADARL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"yaxis": {"majorGrid": {"lineStyle": {
    "width": 1, "color": "navy", "dash": "2 2"}}},
"series": [
    {"series": 0, "color": "purple", "border": {"width": 2}},
    {"series": 1, "color": "cyan", "border": {"width": 2}}
    ]
*END
ENDSTYLE
END
```

The output is:



Streamgraph Charts

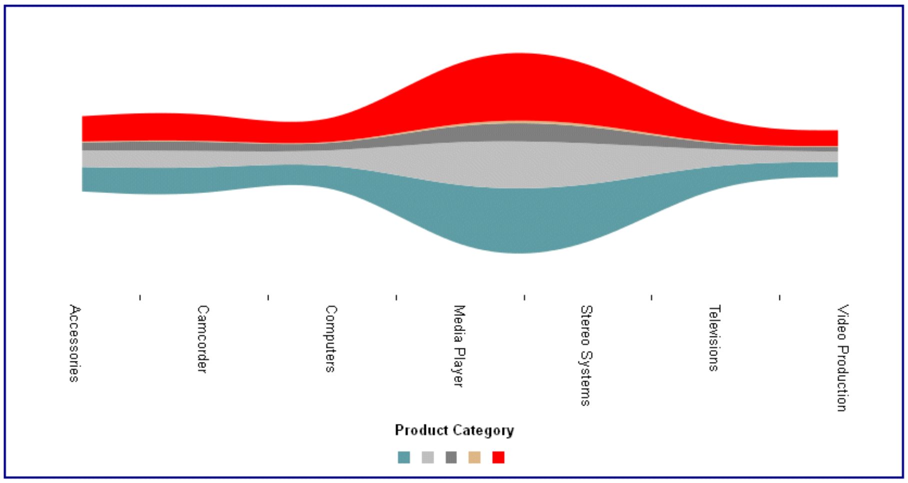
A streamgraph is a simplified version of a stacked area chart. In a streamgraph, there are no axes, gridlines, or frames. The baseline is free, which makes it easier to perceive the thickness of any given layer across the data. A streamgraph does not use data text labels. The data required to draw a streamgraph is the same format required to draw an area chart. However, streamgraphs are normally given many series (10 or more), each with many data point points (100 or more). A typical streamgraph would include 20 series with 400 data points in each series.

Example: Sample Streamgraph

The following request creates a streamgraph (ON GRAPH SET LOOKGRAPH STREAM). The JSON block in the style section sets the border and series colors and widths:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US COGS_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH STREAM
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "navy"},
"series": [
    {"series": 0, "color": "cadetblue","label": " "},
    {"series": 1, "color": "silver", "label": " "},
    {"series": 2, "color": "grey", "label": " "},
    {"series": 3, "color": "burlywood", "label": " "},
    {"series": 4, "color": "red", "label": " "}
    ]
*END
ENDSTYLE
END
```

The output is:



Surface3D Charts

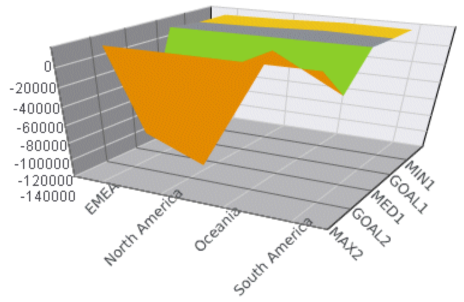
A Surface3D chart shows a three dimensional surface that connects a set of data points. The colors on the chart represent similar values, rather than different series.

Example: Sample Surface3D Chart

The following request creates a Surface3D chart (ON GRAPH SET LOOKGRAPH 3DSURFCE). The JSON block in the style section hides the legend. For all other properties, the defaults are used:

```
DEFINE FILE WF_RETAIL_LITE
DIFF = GROSS_PROFIT_US - REVENUE_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF NOPRINT AND COMPUTE
MIN1 = MIN.DIFF;
GOAL1 = MIN1 +200;
MED1 =MDN.DIFF;
GOAL2 =DIFF;
MAX2 = MAX.DIFF;
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DSURFCE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is:



Tagcloud Charts

A tagcloud chart is a visual representation of group labels. The size of each label is proportional to its data value. Tagclouds are used by social media to measure the frequency of words in order to quantify sentiments.

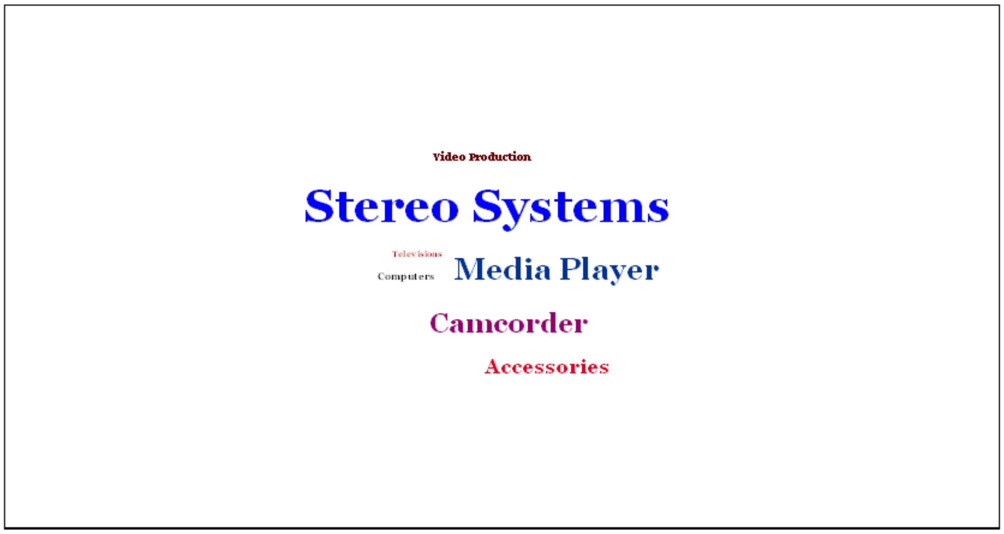
The tagcloud chart also supports an optional second series of data, which is used to drive the color of each label. Colors are chosen from the colorScale property.

Example: Sample Tagcloud

The following request creates a tagcloud chart (ON GRAPH SET LOOKGRAPH TAGCLOUD). Each product category label is sized based on the value of its gross profit and is colored depending on the value of its cost. The JSON block in the style section sets the color scale:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
ON GRAPH SET STYLE *
*GRAPH_JS
"colorScale": {"colors": ["black", "red", "purple", "green", "blue"]}
*END
ENDSTYLE
END
```

The output is:



Treemap Charts

A treemap chart displays hierarchical data as a set of nested rectangles.

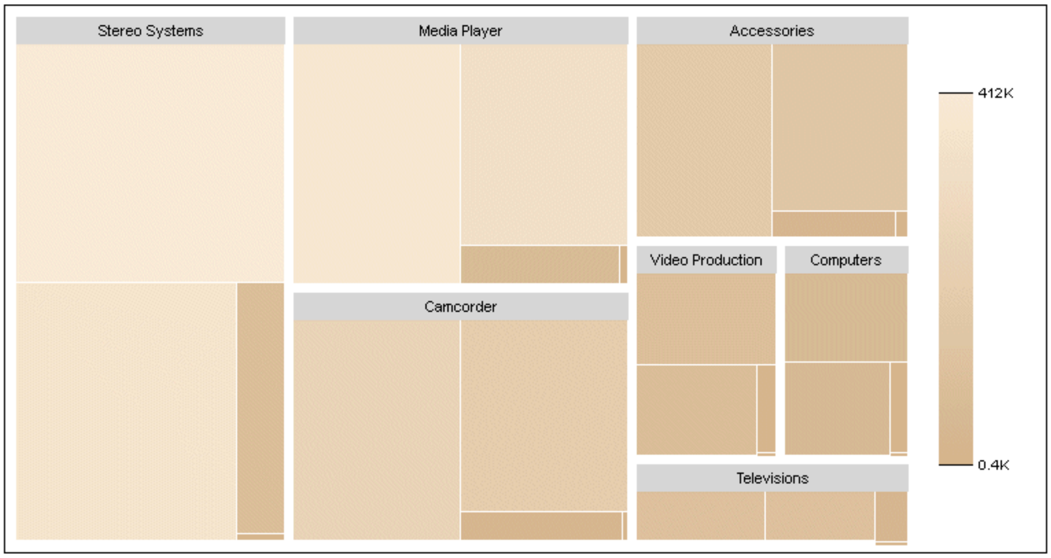
An object can be nested arbitrarily deep. Any properties in the data object with numeric values are drawn as a single rectangle in the treemap.

Example: **Sample Treemap**

The following request generates a treemap chart (ON GRAPH SET LOOKGRAPH TREEMAP). The JSON block in the style section sets the color scale and colors for the treemap:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
    "colorScale":
        {"colors": ["tan", "antiquewhite"]}
*END
ENDSTYLE
END
```

The output is:



Waterfall Charts

Waterfall charts illustrate the cumulative effect of sequentially introducing positive or negative values to an initial value. The initial and final (or total) values are represented by whole columns drawn from the ordinal (x) axis baseline. Intermediate positive and negative values are drawn as floating columns.

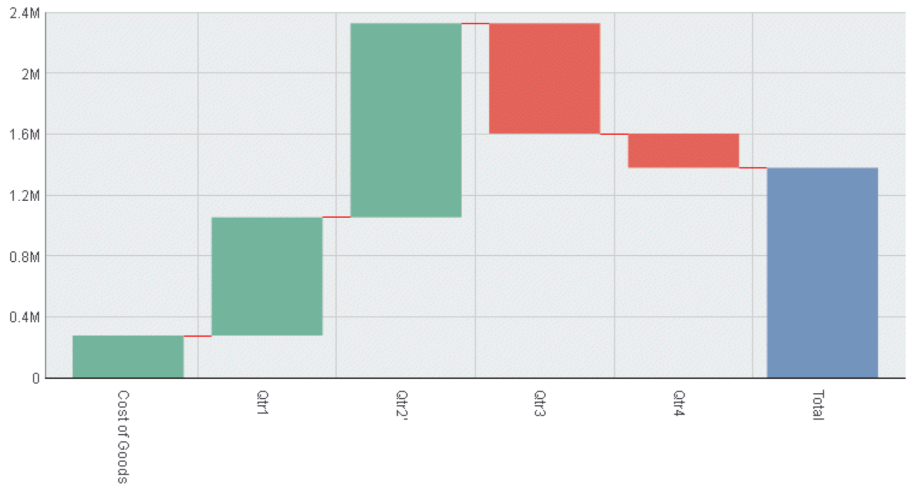
A waterfall chart does not require a sort phrase.

Example: Sample Waterfall Chart

The following request creates a waterfall chart (ON GRAPH SET LOOKGRAPH VWATERFL). The JSON block in the style section sets the riser colors, generates a riser for the total, and sets the connector line properties:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
DECR1 = (COGS_US - 1000);
DECR2 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 AS 'Qtr1'
INCR2 AS Qtr2'
DECR1 AS 'Qtr3'
DECR2 AS 'Qtr4'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"waterfallProperties": {
    "appendTotalRiser": true,
    "positiveRiserColor": "#77b39a",
    "negativeRiserColor": "#e2675b",
    "zeroRiserColor": "#7593bd",
    "connectorLine": {
        "width": 1,
        "color": "red",
        "dash": ""}
}
*END
ENDSTYLE
END
```

The output is:



Bullet Charts

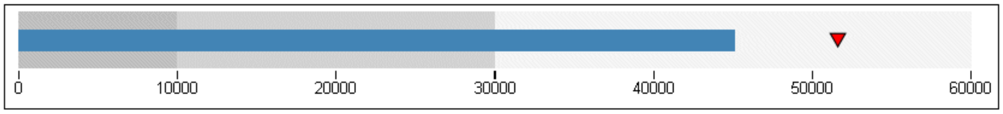
A bullet chart is a microchart that is a variation of the bar chart and is sometimes used as a replacement for a gauge chart. It is designed to show a quantitative measure against qualitative ranges. For example, a quantitative measure such as profit or revenue could be visualized against quality (good, better, best) to show progress toward a target.

Example: Sample Bullet Chart

The following request creates a bullet chart. It charts the cost of goods sold for each weekend day. There is no LOOKGRAPH value for a bullet chart, so the LOOKGRAPH value is set to CUSTOM, and the JSON block in the style section of the request contains the chartType: 'bullet' property. It also sets the color bands for the y-axis and the colors for each day (group). The VAXIS parameter sets the height for the chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'SAT' OR 'SUN'
ON GRAPH SET VAXIS 80
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CUSTOM
ON GRAPH SET STYLE *
*GRAPH_JS
"chartType": "bullet",
"yaxis": {"colorBands": [
    {"start": 0, "stop": 10000, "color": "silver"},
    {"start": 10000, "stop": 30000, "color": "lightgrey"},
    {"start": 30000, "stop": 60000, "color": "whitesmoke"}]},
"series": [
    {"series": 0, "group": 0, "color": "steelblue"},
    {"series": 0, "group": 1, "color": "red"}
    ]
*END
ENDSTYLE
END
```

On the output, the blue bar represents the quantity for Saturday, and the red triangle represents the quantity for Sunday:



Sparkline Charts

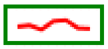
Sparkline is a microchart that has no titles, labels, or legends. It is a single line chart that is intended to be drawn in a very small area to show the shape of the variation without axes or coordinates. A sparkline can be embedded in text or tables.

Example: Sample Sparkline Chart

The following request creates a sparkline chart. Since there is no LOOKGRAPH value for a sparkline chart, the LOOKGRAPH parameter is set to CUSTOM, and the JSON block in the style section of the request contains the chartType: 'sparkline' property. It also sets the border color and width and the series (line) color. The GRAPH parameters HAXIS and VAXIS set the height and width of the chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH SET HAXIS 50
ON GRAPH SET VAXIS 20
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CUSTOM
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"chartType": "sparkline",
"series": [{"series": 0, "color": "red"}],
"border": {"width": 2, "color": "green"}
*END
ENDSTYLE
END
```

The output is:



Chapter 3

Introduction to JSON Properties for HTML5 Charts

This chapter describes basic JSON syntax, color and gradient definitions, and number formats.

In this chapter:

WebFOCUS HTML5 JSON Syntax Basics

JSON is an open standard format that uses attribute-value pairs written in plain text to define methods and properties. JSON is language-independent.

Reference: JSON Object Definitions

An object is described using an unordered set of name-value pairs that define the properties of the object.

Each object name is followed by a colon (:), followed by a list of properties. The list is enclosed within a left brace ({) and a right brace (}).

Each property is a name-value pair. For example, a property may be the color of the object. That property has the name *color*. Its value is a color. For information about specifying colors, see *Colors and Gradients* on page 85.

A value can be a number, a string, a Boolean value (true or false), a null value, an object, or an array. String values must be enclosed in single quotation marks (') or double quotation marks (").

Note: Strict JSON syntax requires that all property and object names and all string values be enclosed in double quotation marks. Strict JSON syntax is required for all WebFOCUS map chart requests.

A string value must not be broken onto multiple lines in the request. For example:

```
"color": "green"
```

Note: In some of the sample requests in this manual, the line length may cause a string to be separated onto multiple lines. To run the request, you must remove any line breaks within strings.

Each property is separated from the next property by a comma (,). Each object definition is also separated from the next object definition by a comma.

An object can consist of an array. For example, a chart can have multiple series (fields displayed by the request). Series within a request are numbered, starting from zero (0). Therefore, the series object consists of an array in which each member represents one series number. Arrays are enclosed within a left bracket ([) and a right bracket (]).

For example, the following series object describes the color property of first two series in a chart (series 0 and series 1). The numbers 0 and 1 are values for the series property, and the strings ‘cyan’ and ‘bisque’ are values for the color property:

```
"series":
     [
        {"series": 0, "color": "cyan"},
        {"series": 1, "color": "bisque"}
     ]
```

An object can consist of multiple properties and objects.

For example, the following definitions for series 0 and series 1 define the color for each series and its marker and border objects. The marker object defines the shape and size properties of the marker. The border object defines the width and color properties of the border:

```
"series":
     [
      {"series": 0, "color": "cyan",
           "marker": {"shape": "triangle", "size": 12,
           "border": {"width": 1, "color":"blue"}}},
      {"series": 1, "color": "bisque",
           "marker": {"shape": "square", "size": 13,
           "border": {"width": 1, "color":"brown"}}}
    ]
```

White space is not required within or between name-value pairs, but is recommended for clarity.

Note that brackets do not have to be placed on their own lines, but doing so and using indentations makes object definitions easier to identify. Similarly, each object in the array does not have to start on a new line, but starting each on a new line makes them easier to find.

Colors and Gradients

Color properties define the color of an object. All objects can be assigned a color and transparency setting. Area and line objects can be assigned a color definition or a gradient definition.

Syntax: How to Specify Color Properties

```
"color": "string"
```

where:

`"color": "string"`

Can be one of the following:

- ❏ A color name (for example, "red").

  For a list of supported color names, see *http://www.w3.org/TR/css3-color/#svg-color*.

- ❏ Three RGB values, or three RGB values and a transparency setting:

  ```
  "rgb (r,g,b)"
  ```

  or

  ```
  "rgba(r,g,b,a)"
  ```

  The values *r*, *g*, and *b* represent the intensity (from 0 to 255) of red, green, and blue.

  Transparency defines how the object blends into the background, expressed as a number between 0.0 (fully transparent) and 1.0 (fully opaque).

  For example, the color black can be described as "rgb(0,0,0)"

- ❏ Three Hue-Saturation-Lightness (HSL) values, or three HSL values and a transparency setting:

  ```
  "hsl(h,s,l)"
  ```

  or

`"hsla(`*`h, s, l, a`*`)"`

- ❏ Hue is expressed as an angle on the color wheel. Red is at the top and is defined as 0 or 360°, green is 120°, and blue is 240°.
- ❏ Saturation defines how pure the hue is, as a percentage, including a percent symbol (%). A pure color is 100% saturated, while grays are unsaturated.
- ❏ Lightness defines how light or dark the hue is, as a percentage, including a percent symbol. White is 100% lightness, while black is 0% lightness.
- ❏ Transparency defines how the object blends into the background, expressed as a number between 0.0 (fully transparent) and 1.0 (fully opaque).

❏ A hexadecimal color value:

`"#`*`hexvalue`*`"`

The hexadecimal color value starts with a pound sign (#), then has two hexadecimal digits each for the combination of red, green, and blue color values (RGB). The lowest value for each component is 0 (hex 00). The highest value is 255 (hex FF).

For example, black is #000000, which corresponds to rgb(0,0,0). Red is #FF0000, which corresponds to rgb(255,0,0). White is #FFFFFF, which corresponds to rgb(255,255,255).

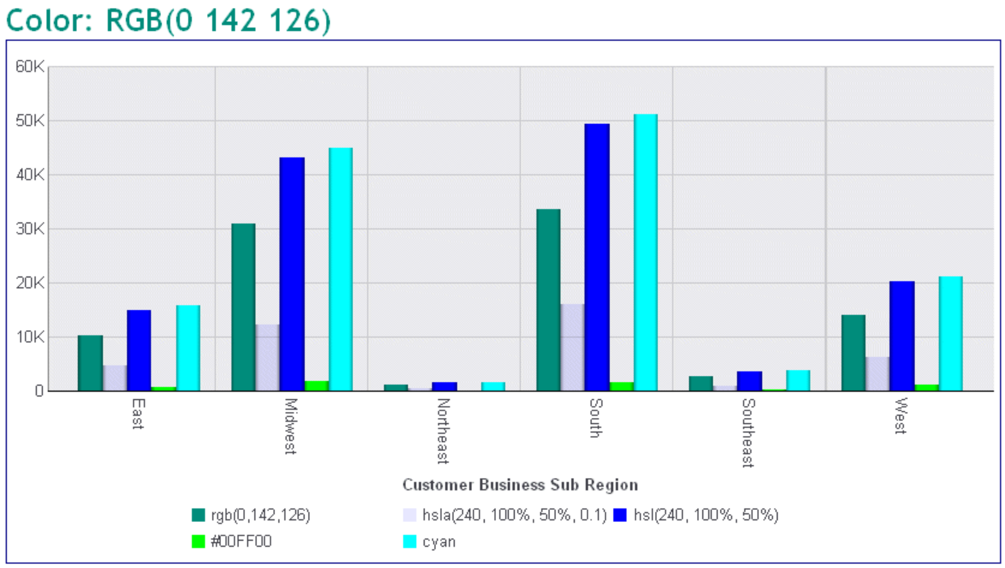
The World Wide Web Consortium (W3C) website at *http://www.w3.org/TR/css3-color/#colorunits* provides details about color specifications.

Example: Defining Colors

The following request has five series. The JSON definitions for each series illustrate one type of color definition, and display that color definition in the legend as the series label. The heading and border also have color definitions:

```
GRAPH FILE WF_RETAIL_LITE
HEADING
"Color: RGB(0 142 126)"
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US MSRP_US
BY BUSINESS_SUB_REGION
WHERE COUNTRY_NAME EQ 'United States'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"border": {"width": 1, "color": "navy"},
"series": [
   {"series": 0, "color": "rgb(0,142,126)", "label": "rgb(0,142,126)"},
   {"series": 1, "color": "hsla(240, 100%, 50%, 0.1)",
     "label": "hsla(240, 100%, 50%, 0.1)"},
   {"series": 2, "color": "hsl(240, 100%, 50%)", "label": "hsl(240, 100%, 50%)"},
   {"series": 3, "color": "#00FF00", "label": "#00FF00"},
   {"series": 4, "color": "cyan", "label": "cyan"}
   ]
*END
TYPE=HEADING, COLOR = rgb(0 142 126), JUSTIFY=LEFT,$
ENDSTYLE
END
```

The output is:



Gradient Definitions

Gradients are transitions between colors. The transitions can be applied linearly across the chart or radially from a central point outward.

Gradients can be defined as a string or a JSON object.

Syntax: How to Define Linear Gradient Properties as a String

```
"color": "linear-gradient(startX, startY, endX, endY,
 stopsOffset  stopsColor,...stopsOffset  stopsColor)"
```

where:

startX, starty

Is the starting x-axis and y-axis position for the color gradient.

The axis positions can be specified as numbers (for example, 0) or as strings (for example, "5%"). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, start: x:0, y:0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero. Numeric start and end coordinates are only useful for objects where you can set the area dimensions (such as the chart background area). For example, if the chart background area is set to 200 by 200 pixels, and start x:0, y:0 and stop x:100, y:0 are used to define a linear gradient, the gradient will be applied from the left edge to the center of the rectangle. If the size of the object is calculated by the library (for example, legend area, chart frame, risers), string percentages are typically used.

`endX, endy`

Applies to linear gradients only. Is the ending x-axis and y-axis position for the color gradient.

The axis positions can be specified as numbers (for example, 20) or as strings (for example, "20%"). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, start: x:0, y:0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero. Numeric start and end coordinates are only useful for objects where you can set the area dimensions (such as the chart background area). For example, if the chart background area is set to 200 by 200 pixels, and start x:0, y:0 and stop x:100, y:0 are used to define a linear gradient, the gradient will be applied from the left edge to the center of the rectangle. If the size of the object is calculated by the library (for example, legend area, chart frame, risers), string percentages are typically used.

`stopsOffset`

Offset values define where the color changes. They can be numbers, or they can be strings enclosed in single quotation marks with a percent symbol (%). Only numbers between 0 and 1 are valid, and are treated as percentages. A value 0.5 is the same as 50%. Numbers less than zero are treated as zero, and numbers greater than one are treated as 1.

`stopsColor`

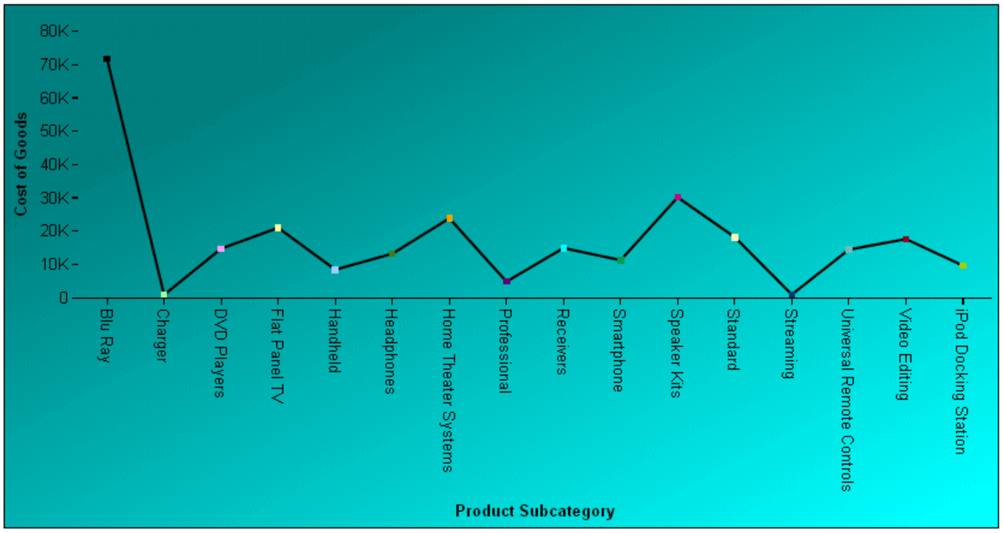
Specifies a color for the corresponding offset.

Example: Sample Linear Gradient String

The following request specifies a linear gradient for the background of the chart. The start and end coordinates describe the entire chart area, and the gradient transitions from teal to cyan. The linear gradient is specified as a string:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
 *GRAPH_JS
"fill": {
   "color": "linear-gradient(0,0,100%,100%, 20% teal, 95% cyan)"
   },
"series": [{"series": 0, "color": "black"}]
*END
ENDSTYLE
END
```

The output is:



Syntax: How to Define Radial Gradient Properties as a String

```
"color": "radial-gradient(startX, startY, radius,
 stopsOffset stopsColor,...stopsOffset stopsColor)"
```

where:

startX, starty

Is the starting x-axis and y-axis position for the color gradient.

The axis positions can be specified as numbers (for example, 0) or as strings (for example, "5%"). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. Numeric start and end coordinates are only useful for objects where you can set the area dimensions (such as the chart background area). If the size of the object is calculated by the library (for example, legend area, chart frame, risers), string percentages are typically used. For radial gradients, start: x/y defines the center of the gradient (for example, start: x: "50%", y: "50%" draws the gradient starting from the center of the object).

radius

Applies to radial gradients only. Defines the distance from the center to the outermost edge of the gradient. The start x and y coordinate defines the center of the gradient (for example, "start": "x": "50%", "y": "50%" draws the gradient in the center of the object). Therefore, if an object is 200 by 200 pixels with a gradient start: "x":"50%", "y":"50%" and "radius": "100%", the outermost gradient edge will be 100 pixels beyond the edges of the object. In this example, 100% of 200 pixels is 200 pixels, so the gradient is actually 400 pixels from extreme left to extreme right. For a radial gradient that starts in the center ("start": "x":"50%"/"y":"50%"), "radius": "50%" is normally used to draw the gradient from the center to the outermost edges of the object.

stopsOffset

Offset values define where the color changes. They can be numbers, or they can be strings with a percent symbol (%). Only numbers between 0 and 1 are valid, and are treated as percentages. A value 0.5 is the same as 50%. Numbers less than zero are treated as zero, and numbers greater than one are treated as 1.

stopsColor

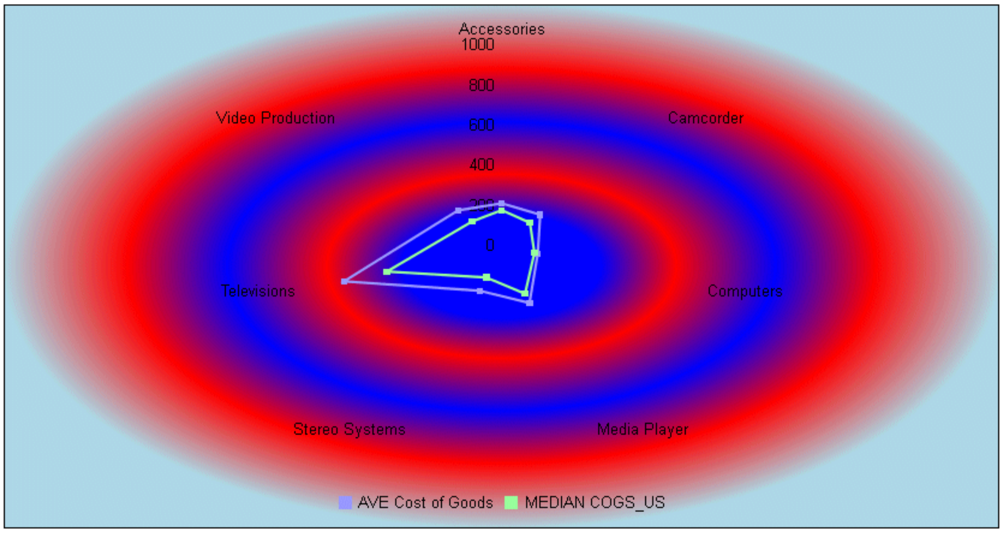
Specifies a color for the corresponding offset.

Example: Sample Radial Gradient String

The following request specifies a radial gradient for the background of the chart. The start position and radius describe the entire chart background. From inside to outside, the colors transition from blue to red to blue to red to lightblue. The gradient is specified as a string:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH RADARL
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {
   "color":
"radial-gradient(50%,50%,50%, 20% blue, 35% red, 55% blue, 75% red, 1 lightblue)"
   }
*END
ENDSTYLE
END
```

The output is:



Syntax: How to Define Gradient Properties as a JSON Object

```
"color": {
   "type": "string",
   "start":{
      "x": sx1,
      "y": sy1   },
   "end":{
      "x": ex1,
      "y": ey1   },
   "radius": "rstring",
   "stops": [
         {
            "offset": off1,
            "color": "color1"
         },
         ...
      ]
}
```

where:

`"type": "string"`

Is a string that defines the type of gradient. Valid values are "linear" or "radial".

`"start": "x", "start": "y"`

Specify the gradient coordinates in the chart.

`"x": x`

Specifies the starting x-coordinate in the chart. It can be specified as a number or as a percent value enclosed in single quotation marks and including the percent symbol.

Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, "start": "x":0, "y":0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero.

Percentage coordinates are most useful when the size of an object is calculated by the library or, for radial gradients, where 50% specifies the center of the object.

`"y": y`

Specifies the starting y-coordinate in the chart. It can be specified as a number or as a percent value enclosed in single quotation marks and including the percent symbol.

Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, "start": "x":0, "y":0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero.

Percentage coordinates are most useful when the size of an object is calculated by the library or, for radial gradients, where 50% specifies the center of the object.

The start: x/y and end: x/y parameters can be specified as numbers (for example, 0/20) or as strings (for example, "5%"/"20%"). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, "start": "x":0/"y":0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero. Numeric start/end coordinates are only useful for objects where you can set the area dimensions (such as the chart background area). For example, if the chart background/draw area is set to 200 by 200 pixels and "start" "x":0/"y":0 and "stop" "x":100/"y":0 is used to define a linear gradient, the gradient will be applied from the left edge to the center of the rectangle. If the size of an object is calculated by the library (such as for the legend area, chart frame, or risers), string percentages are typically used. For radial gradients, "start": "x"/"y" defines the center of the gradient (for example, "start": "x": "50%"/"y": "50%" draws the gradient in the center of the object).

`"end": "x", "end": "y"`

Specify the gradient coordinates in the chart.

`"x":` *x*

Applies to linear gradients only. Specifies the ending x-coordinate in the chart. It can be specified as a number or as a percent value enclosed in single quotation marks and including the percent symbol.

Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. Negative numbers are not valid. Any number less than zero is treated as zero.

Percentage coordinates are most useful when the size of an object is calculated by the library or for radial gradients, where 50% specifies the center of the object.

`"y":` *y*

Applies to linear gradients only. Specifies the ending y-coordinate in the chart. It can be specified as a number or as a percent value enclosed in single quotation marks and including the percent symbol.

Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. Negative numbers are not valid. Any number less than zero is treated as zero.

Percentage coordinates are most useful when the size of an object is calculated by the library or for radial gradients, where 50% specifies the center of the object.

The start: x/y and end: x/y parameters can be specified as numbers (for example, 0/20) or as strings (for example, "5%"/"20%"). Numbers are interpreted as raw Scalable Vector Graphics (SVG) pixel coordinates. For example, "start": "x":0/"y":0 is the top-left corner of the object. Negative numbers are not valid. Any number less than zero is treated as zero. Numeric start/end coordinates are only useful for objects where you can set the area dimensions (such as the chart background area). For example, if the chart background/ draw area is set to 200 by 200 pixels and start x:0/y:0 and stop x:100/y:0 is used to define a linear gradient, the gradient will be applied from the left edge to the center of the rectangle. If the size of an object is calculated by the library (such as for the legend area, chart frame, or risers), string percentages are typically used. For radial gradients, start: x/y defines the center of the gradient (for example, "start": "x": "50%"/"y": "50%" draws the gradient in the center of the object).

`"radius": "`*`string`*`"`

Applies to radial gradients only. Defines the distance from the center to the outermost edge of the gradient.

For example, if an object is 200 by 200 pixels with a gradient "start": "x":"50%", "y":"50%" and "radius": "100%", the outermost gradient edge will be 100 pixels beyond the edges of the object. In this example, 100% of 200 pixels is 200 pixels, so the gradient is actually 400 pixels from extreme left to extreme right. For a radial gradient that starts in the center ("start": "x":"50%", "y":"50%"), "radius": "50%" is normally used to draw the gradient from the center to the outermost edges of the object.

`"stops":`

Is a comma-separated list of offset-color pairs. This array can be any length, and can be a list of objects or arrays.

`"offset": `*`offset`*

Is an offset expressed as a number or a percent.

If expressed as a number, it must be between 0 and 1 and represents a percentage of the gradient area at which the color changes. The value 0.5 is the same as 50%. Numbers less than zero are treated as zero, and numbers greater than one are treated as 1.

If expressed as a percentage, it must be enclosed in double quotation marks and include the percent symbol. It specifies a percentage of the gradient area at which the color changes.

`"color": "`*`string`*`"`

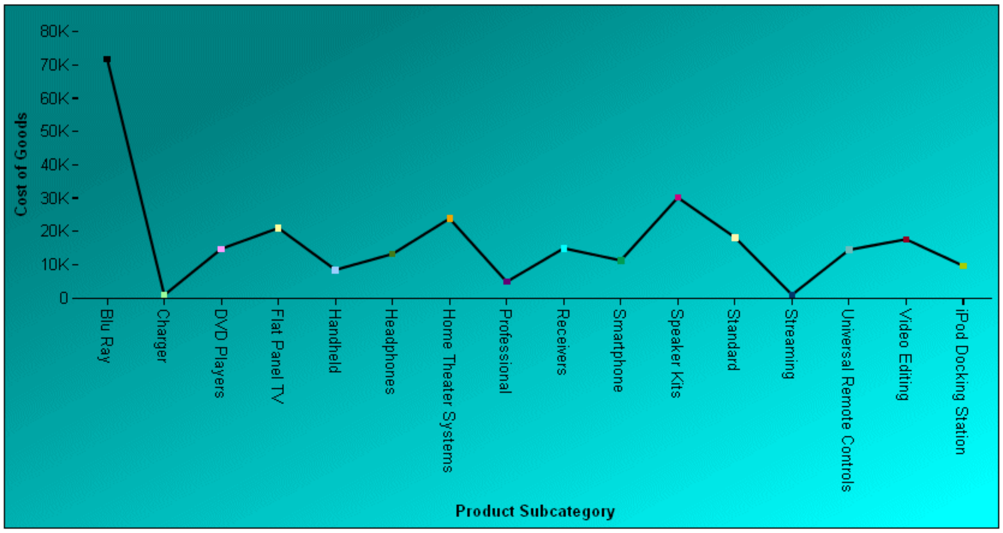
Specifies the color for the corresponding offset.

Example: Sample Linear Gradient JSON Object

The following request specifies a linear gradient for the background of the chart. The start and end coordinates describe the entire chart area, and the gradient transitions from teal to cyan. The linear gradient is specified as a JSON object:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {"color":
  {"type": "linear",
   "start": {"x": 0, "y": 0},
   "end": {"x": "100%", y: "100%"},
   "stops": [
         {"offset": "20%", "color": "teal"},
         {"offset": "95%", "color": "cyan"}
         ]
    }},
"series": [{"series": 0, "color": "black"}]
*END
ENDSTYLE
END
```

The output is:

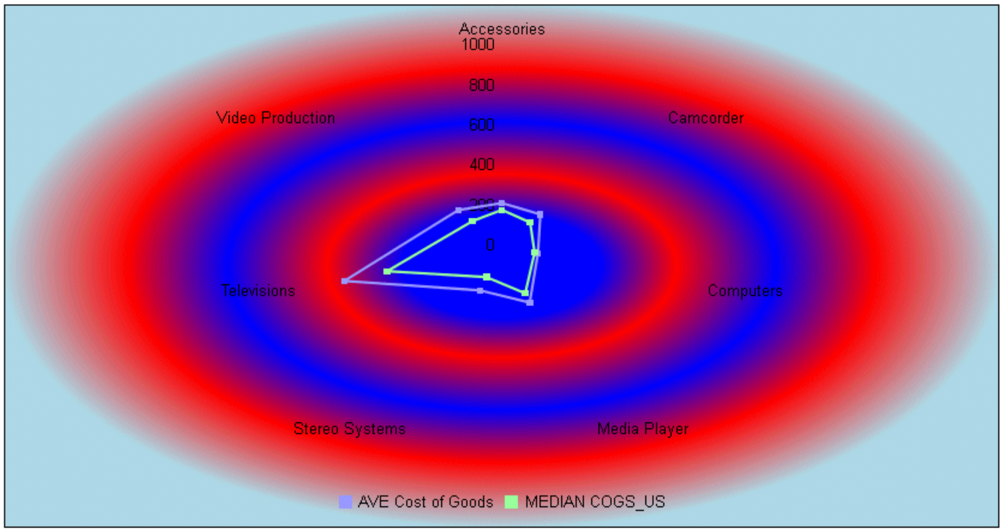


Example: Sample Radial Gradient JSON Object

The following request specifies a radial gradient for the background of the chart. The start position and radius describe the entire chart background. From inside to outside, the colors transition from blue to red to blue to red to lightblue. The gradient is specified as a JSON object:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH RADARL
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {
  "color":
   {"type": "radial",
   "start": {"x": "50%", "y": "50%"},
   "end": {"x": "50%", "y": "50%"},
   "stops": [
        {"offset": "20%", "color: "blue"},
        {"offset": "35%", "color": "red"},
        {"offset": "55%", "color": "blue"},
        {"offset": "75%", "color": "red"},
        {"offset": 1, "color": "lightblue"}
         ]
    }}
*END
ENDSTYLE
END
```

The output is:



SVG Color Patterns

In some cases, you may want to use patterns to distinguish chart elements such as risers and markers, or to fill chart frames. This can be a useful technique in black and white environments where colors are not distinguishable.

Syntax: How to Define an SVG Color Pattern

```
"color":{
  "type":"pattern",
   "color": "string",
   "backgroundColor": "string",
   "shape": "string",
   "size": number,
   "pad": number}
```

where:

`"color": "string"`

Defines the color of the pattern shape using a color name or numeric specification string.

`"backgroundColor": "string"`

Defines the background color of the pattern space using a color name or numeric specification string.

`"shape": "`*`string`*`"`

Defines the shape for the pattern character. Any SVG path string and all marker shapes are supported. For information on marker shapes, see *Defining the Size, Border, Color, Shape, and Rotation of Series Markers* on page 452.

`"size":` *`number`*

Is the size, in pixels, of the pattern shape.

`"pad":` *`number`*

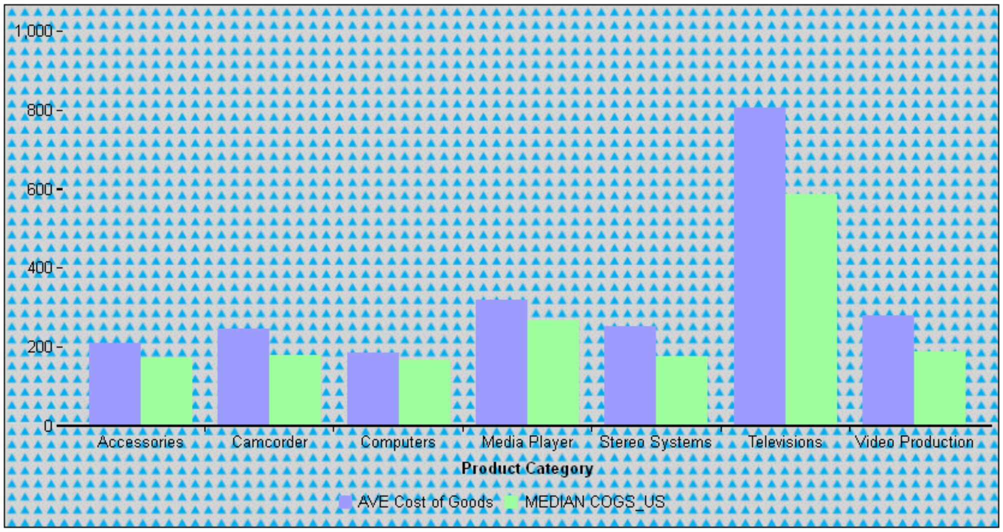
Is the size, in pixels, of the padding between the pattern shapes.

Example: **Using a Pattern to Fill the Chart Frame**

The following request fills the chart frame with triangles whose color is a version of blue and whose size is 10 pixels. The space between triangles is 8 pixels, and the background color is light gray:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {
  "color":{
   "type":"pattern",
     "color": "#00aeef",
     "backgroundColor": "lightgray",
     "shape": "triangle",
     "size": 10,
     "pad": 8}}
*END
ENDSTYLE
END
```

The output is:



Example: Using an SVG Path String to Define a Pattern

A path string describes a pattern as movement across a sheet of paper, starting at the origin. A line is described using:

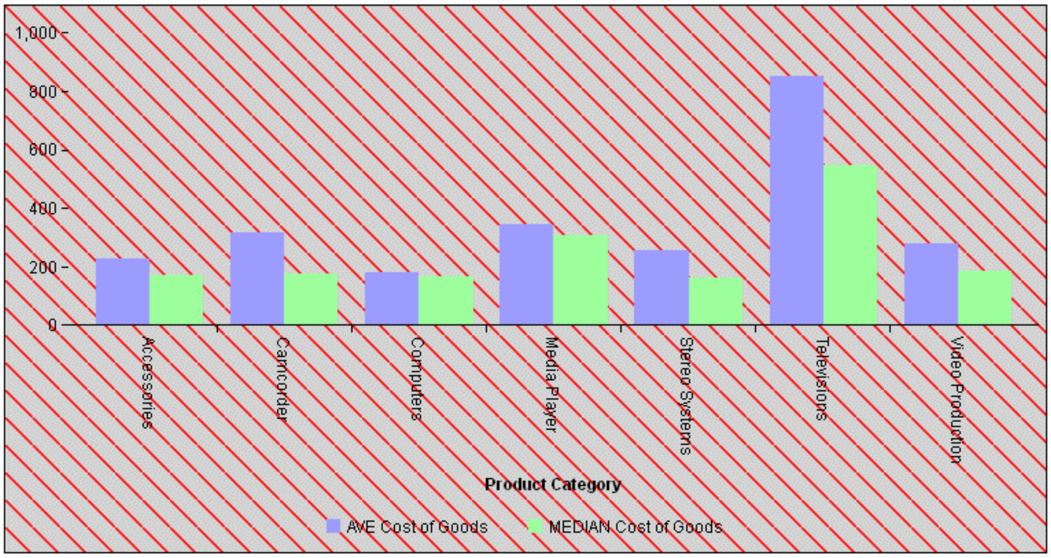
- ❏ A moveto command (M) that moves to a new coordinate defined by the pair of numbers following the M.
- ❏ A lineto command (L) moves from the current position to a new coordinate, tracing a straight line along the way.

Multiple M and L commands can be used.

The following request uses an SVG path string to define a 45-degree cross-hatch pattern:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {
  "color":{
   "type": "pattern",
   "color": "red",
   "backgroundColor": "lightgray",
   "shape": "M-10,-10L20,20",
   "size": 20}
       }
*END
ENDSTYLE
END
```

The output is shown in the following image:

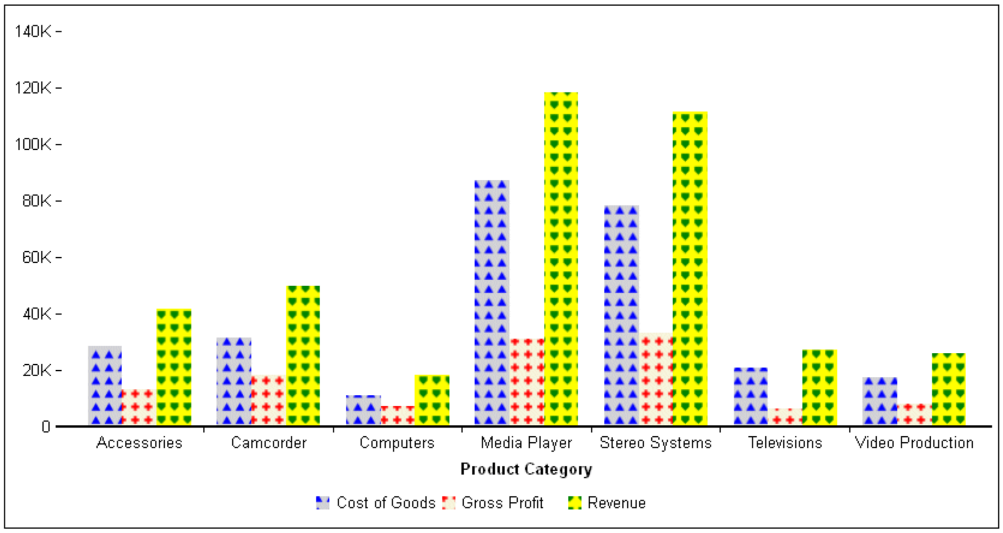


Example: Using a Pattern to Fill Series Risers

The following request defines patterns to color the risers for each series in a vertical bar chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series":[
 {"series":0, "color":{"type":"pattern","color": "blue","backgroundColor":
"lightgray","shape": "triangle","size": 10,"pad": 8}} ,
 {"series":1, "color":{"type":"pattern","color": "red","backgroundColor":
"beige","shape": "plus","size": 10,"pad": 8}} ,
 {"series":2, "color":{"type":"pattern","color": "green","backgroundColor":
"yellow","shape": "house","size": 10,"pad": 8}}
]
*END
ENDSTYLE
END
```

The output is:

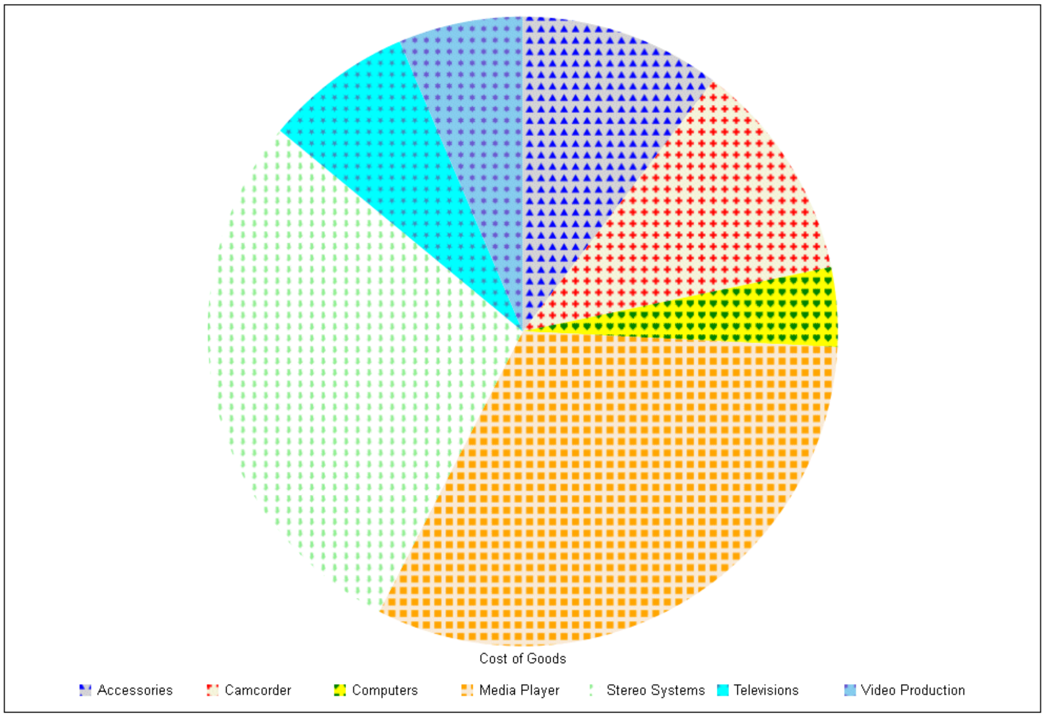


Example: Using a Pattern to Fill Pie Slices

The following request defines patterns to color the risers for each series in a pie chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH SET AUTOFIT ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
 {"series": 0, "color": {"type":"pattern","color":
"blue","backgroundColor": "lightgray","shape": "triangle","size": 10,"pad":
8}} ,
 {"series": 1, "color": {"type":"pattern","color": "red","backgroundColor":
"beige","shape": "plus","size": 10,"pad": 8}} ,
 {"series": 2, "color": {"type":"pattern","color":
"green","backgroundColor": "yellow","shape": "house","size": 10,"pad": 8}},
 {"series": 3, "color": {"type":"pattern","color":
"orange","backgroundColor": "antiquewhite","shape": "square","size":
10,"pad": 8}},
 {"series": 4, "color": {"type":"pattern","color":
"lightgreen","backgroundColor": "white","shape": "arrow","size": 10,"pad":
8}},
 {"series": 5, "color": {"type":"pattern","color":
"steelblue","backgroundColor": "cyan","shape": "fiveStar","size": 10,"pad":
8}},
 {"series": 6, "color": {"type":"pattern","color":
"slateblue","backgroundColor": "skyblue","shape": "sixStar","size":
10,"pad": 8}}
]
*END
ENDSTYLE
END
```

The output is:

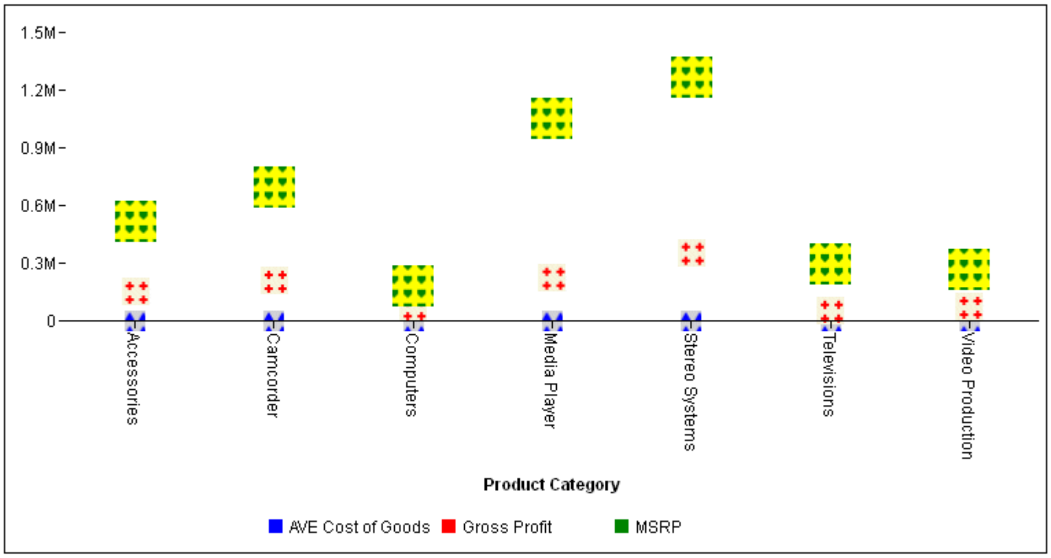


Example: **Using a Pattern to Fill Series Markers**

The following request generates a scatter chart. The markers for each series are filled with patterns and are sized large enough to fit the patterns:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US GROSS_PROFIT_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERS
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
{"series":0, "marker":{"size":15},"color":{"type":"pattern","color":
"blue","backgroundColor": "lightgray","shape": "triangle","size": 10,"pad":
8}},
{"series":1, "marker":{"size":20},"color":{"type":"pattern","color":
"red","backgroundColor": "beige","shape": "plus","size": 10,"pad": 8}},
{"series":2, "marker":{"size":30},"color":{"type":"pattern","color":
"green","backgroundColor": "yellow","shape": "house","size": 10,"pad": 8}}
]
*END
ENDSTYLE
END
```

The output is:



Font Definitions

All font properties are specified as a string of font attributes (for example, '10pt Sans-Serif'), using the format defined at:

http://www.w3.org/TR/CSS2/fonts.html#font-shorthand.

Fonts always have the properties name, size, style (for example, normal, bold, or italic), and color. Each of these has a default, depending on the fonts installed and available to your browser, and your configuration.

Note: Underline is not supported.

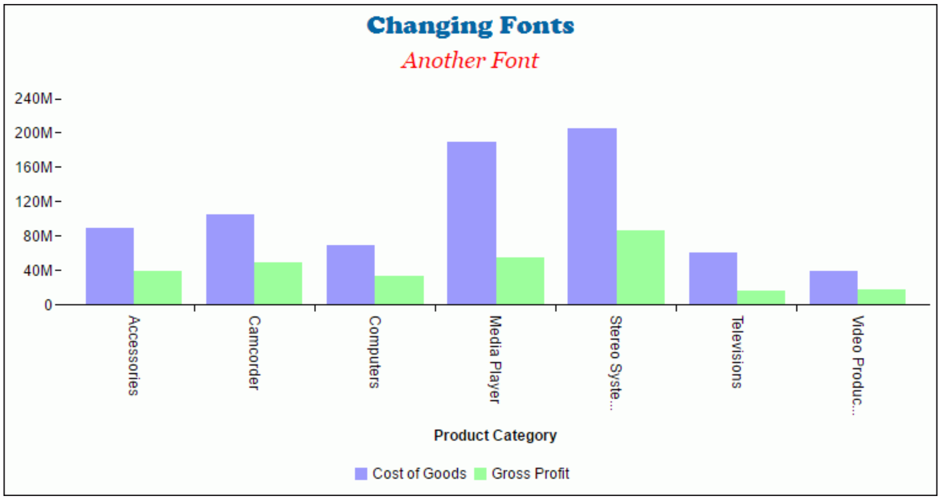
Note that any font you specify must be installed and accessible to the browser.

Example: Specifying a Font

The following request specifies fonts and font characteristics for the chart title and chart subtitle:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"title": {"visible": true, "text": "Changing Fonts",
    "font": "bold 16pt Cooper", "color": "rgb(0,101,163)"},
"subtitle": {"visible": true, "text": "Another Font",
    "font": "italic 14pt Georgia", "color": "red"}
*END
ENDSTYLE
END
```

The output is:



Using Google Fonts

You can use Google fonts in your request by pointing to the Google public font API, using the SET CSSURL command.

For a list of available Google fonts, see http://www.google.com/fonts.

Syntax: How to Access Google Fonts in a Chart Request

```
SET CSSURL='http://fonts.googleapis.com/css?family=font1|font2 ... '
```

where:

font1|font2 ...

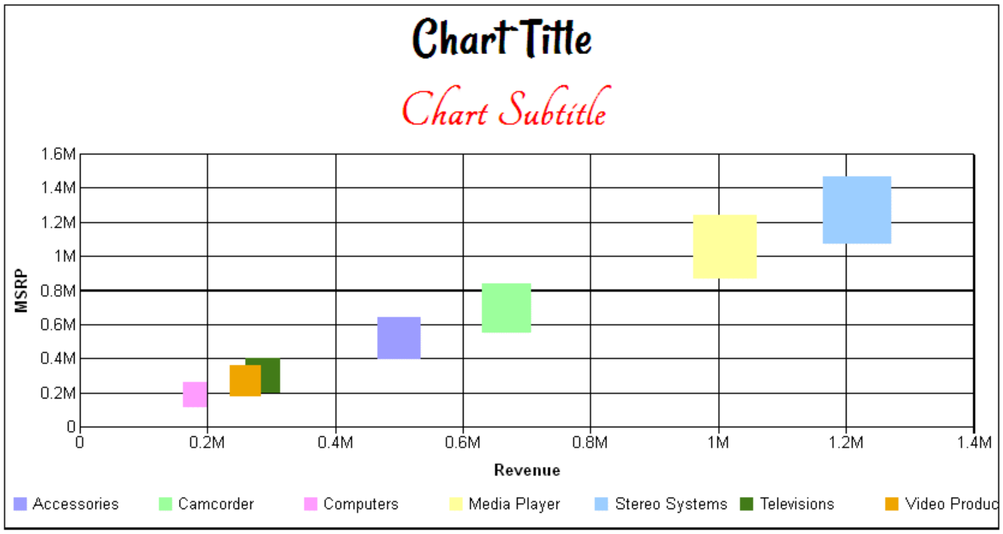
Are Google font names separated by a vertical bar (|).

Example: Using the Rancho and Tangerine Google Fonts in a Chart

The following request uses the Rancho font for the chart title and the Tangerine font for the chart subtitle:

```
SET CSSURL='http://fonts.googleapis.com/css?family=Rancho|Tangerine'
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"title": {"text": "Chart Title", "visible": true, "align": "center",
    "font": "bold 30pt Rancho", "color": "black"},
"subtitle": {"text": "Chart Subtitle", "visible": true, "align": "center",
    "font": "bold 30pt Tangerine", "color": "red"}
*END
ENDSTYLE
END
```

The output is shown on the following image:



Formatting Numbers

The numberFormat property specifies the format of numeric labels using one of the following:

- "auto" (automatic)
- JSON Object

- format "string"
- function()

Automatic Number Formatting

Use numberFormat: "auto" to let the charting engine choose the number format based on the chart data. Prefix and suffix characters can be added to the chosen format by enclosing "auto" in two sets of braces. For example, the following indicates automatic number formatting with a tilde (~) and a dollar sign ($) to the left of the number, and a plus sign (+) to the right:

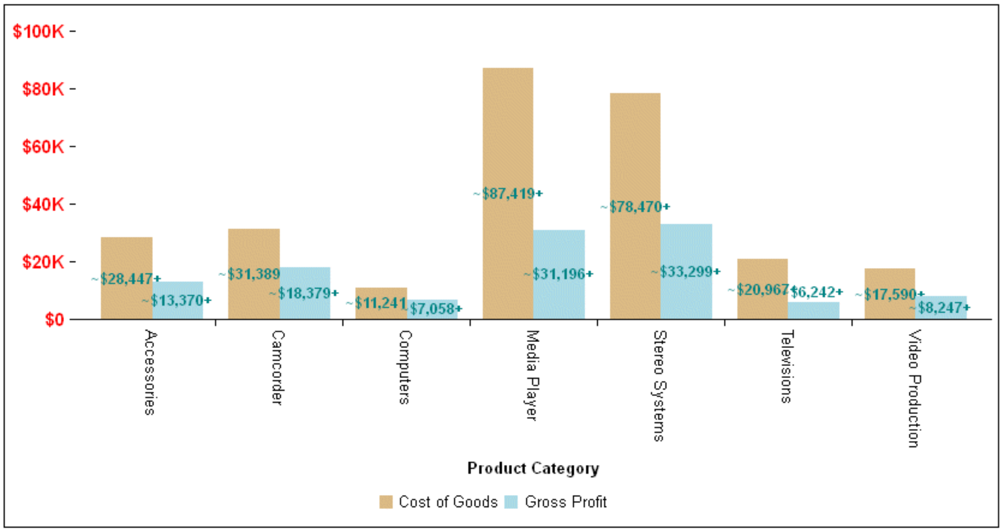
```
"numberFormat": "~${{auto}}+"
```

Example: Using Automatic Number Formatting

The following request uses automatic number formatting for the y-axis and the data text. The y-axis numbers are displayed prefixed with a dollar sign ($), and the data text labels are displayed prefixed with a tilde and dollar sign (~$) and followed by a plus sign (+):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"yaxis": {
    "labels": {"font": "bold 10pt Sans-Serif","color": "red"},
    "numberFormat": "${{auto}}"},
"dataLabels": {"visible": true, "font": "bold 8pt Sans-Serif", "color":
"teal",
    "numberFormat": "~${{auto}}+"},
"series": [
    {"series": "all", "showDataValues": true},
    {"series": 0, "color": "burlywood"},
    {"series": 1, "color": "lightblue"},
    {"series": 2, "color": "lightgreen"}
]
*END
ENDSTYLE
END
```

The output is:



JSON Object Number Formatting

The JSON object includes the following properties. You can omit properties for which you want to accept the default value:

```
"numberFormat": {
   "mode": "string",
   "thousandSep": "string",
   "decimalSep": "string",
   "decimalPlaces": number,
   "grouping": "string",
   "prefix": "string",
   "suffix": "string",

}
```

where:

`"mode": "string"`

Identifies the type of number. Valid values are:

- "numeric". This is the default value.
- "percent", to display the number multiplied by 100 and followed by a percent symbol (%).
- "currency", to display the number with a currency symbol. The currency symbol displayed depends on your number format settings.

- "scientific", to display the number in scientific notation.

`"thousandSep": "`*`string`*`"`

Specifies a character to separate values above and in multiples of one thousand, for example, a comma (,), as in 1,000,000. The default depends on your number format settings.

`"decimalSep": "`*`string`*`"`

Specifies a character to separate decimals (for example, a decimal point (.), as in 1.00). The default depends on your number format settings.

`"decimalPlaces": `*`number`*

Specifies the number of decimal places (number of digits to show to the right of the decimalSep character). The default is 2 for currency and scientific, and zero for numeric and percent.

`"grouping": "`*`string`*`"`

Specifies how to group large numbers if you do not want to display all of the digits. Valid values are:

- "K" to group by thousands. For example, 1,000 = 1K.
- "M" to group by millions. For example, 1,000,000 = 1M.
- "B" to group by billions. For example, 1,000,000,000 = 1B.
- "T" to group by trillions. For example, 1,000,000,000,000 = 1T.

`"prefix": "`*`string`*`"`

Specifies one or more characters to add to the front of the label.

`"suffix": "`*`string`*`"`

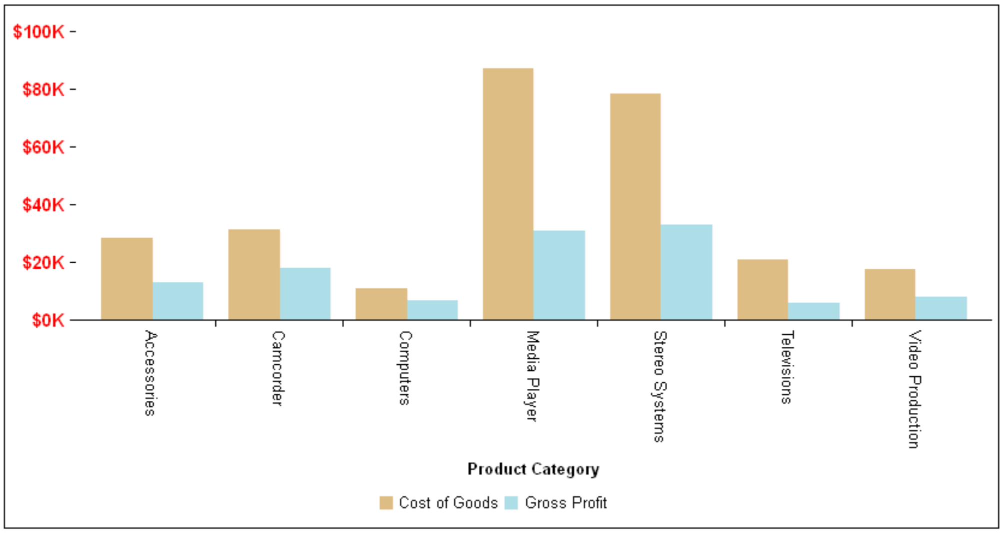
Specifies one or more characters to add to the end of the label.

Example: Using a JSON Object to Format Numbers

The following request formats the y-axis labels as currency, grouped by thousands, with no decimal places, so they display with an automatic dollar sign ($) and the letter K:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"yaxis": {
    "labels": {"font": "bold 10pt Sans-Serif", "color": "red"},
    "numberFormat": {"mode": "currency", "decimalPlaces": 0, "grouping":
"K"}},
"series": [
    {"series": 0, "color": "burlywood"},
    {"series": 1, "color": "lightblue"},
    {"series": 2, "color": "lightgreen"}
]
*END
ENDSTYLE
END
```

The output is:



Formatting Numbers Using a Format String

You can specify the format of numeric labels using a format string:

```
numberFormat: "string"
```

The string must be in the format defined by the Custom Numeric Format Strings defined at:

http://msdn.microsoft.com/en-us/library/0c899ak8(vs.71).aspx.

Reference: Summary of Format String Characters

A number formatting string can consist of from one to three sections. If it has one section, the formatting string is used to format all numbers. If it has two sections, the first is used to format positive numbers and zeros, and the second is used to format negative numbers. If it has three sections, the first is used to format positive numbers, the second is used to format negative numbers, and the third used to format zeros.

The sections are separated by semicolons (;). If the second section is empty (there are two consecutive semicolons between the first and third sections), the first section is used to format all non-negative numbers.

If a negative value is rounded and becomes zero, it is formatted as a zero value.

The following are valid within a format string:

`Zero placeholder`

Zero (0) or double zero (00)

If the value being formatted has a digit in the position where the 0 appears in the format string, then that digit is copied to the result string. Otherwise, a zero is placed in that position. The position of the leftmost 0 before the decimal point and the rightmost 0 after the decimal point determine the range of digits that are always present in the result string.

The 00 placeholder causes the value to be rounded to the nearest digit preceding the decimal. Rounding is always in the direction away from zero. For example, formatting 34.5 with "00" would result in the value 35.

`Digit placeholder`

Pound sign (#) or double pound sign (##).

If the value being formatted has a digit in the position where the # appears in the format string, that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string. Note that this specifier never displays the 0 character if it is not a significant digit, even if 0 is the only digit in the string.

The ## format string causes the value to be rounded to the nearest digit preceding the decimal, where rounding is always away from zero. For example, formatting 34.5 with "##" would result in the value 35.

Decimal separator

Depends on the number formatting settings. for American English, it is a decimal point (.).

The first decimal separator character in the format string determines the location of the decimal separator in the formatted value. Any additional decimal separator characters are ignored.

Thousands separator and scaling

Depends on the number formatting settings. for American English, it is a comma (,).

The thousands separator character serves two purposes. First, if the format string contains a thousands separator character between two digit placeholders (0 or #) and to the left of the decimal separator, if one is present, then the output will have thousand separators inserted between each group of three digits to the left of the decimal separator.

Second, if the format string contains one or more thousand separator characters immediately to the left of the decimal point, then the number will be divided by the number of those characters, multiplied by 1000 before it is formatted. For example, the following format string format the number 100,000,000 as simply 100.

```
0,,
```

Use of the thousand separator character to indicate scaling does not include thousand separators in the formatted number. Thus, to scale a number by 1 million and insert thousand separators you would use the following format string :

```
#,##0,,
```

Scientific Notation

E0, E+0, E-0, e, e+0, or e-0

If any of these strings are present in the format string and are followed immediately by at least one 0 character, then the number is formatted using scientific notation with an "E" or "e" inserted between the number and the exponent. The number of 0 characters following the scientific notation indicator determines the minimum number of digits to output for the exponent. The E+ and e+ formats indicate that a sign character (plus or minus) should always precede the exponent. The E, E-, e, or e- formats indicate that a sign character should only precede negative exponents.

`Percentage placeholder`

Depends on the number formatting settings. for American English, it is a percent symbol (%).

The presence of a percentage placeholder character in a format string causes a number to be multiplied by 100 before it is formatted. The appropriate symbol is inserted in the number itself at the location where the percentage placeholder character appears in the format string.

`Literal String or other characters`

Characters enclosed in single or double quotation marks. For example, "ABC".

Characters enclosed in single or double quotation marks are copied to the result string literally.

In addition to the format specifiers defined in this section, you can also use the following characters:

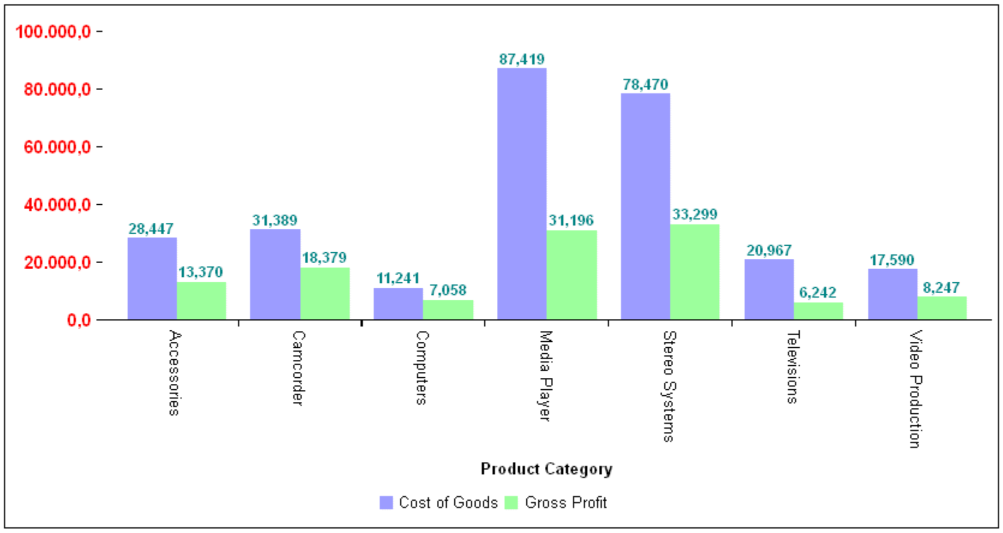
- `"{tc}"` to specify a thousands separator character, where *c* is the separator character.
- `'{dc}'` to specify a decimals separator character, where *c* is the separator character.
- back tick (`) to use a percent symbol in the format without having the number multiplied by 100 before being formatted.

Example: Formatting Labels Using a Number Format String

The following request uses a number format string for the y-axis labels and for the data labels. The series object enables data labels. The y-axis object specifies a format string that changes the thousands separator to a point (.) and the decimal separator to a comma (,) using the {t} and {d} syntax. The dataLabels object makes the labels visible and uses the default number format options:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series": "all", "showDataValues": true}],
"yaxis": {
    "labels": {"font": "bold 10pt Sans-Serif", "color": "red"},
    "numberFormat": "{t.}{d,}#,#.#"},
"dataLabels": {"visible": true, "position": "top", "font": "bold 8pt Sans-
Serif", "color": "teal"}
*END
ENDSTYLE
END
```

The output is:



Syntax: **How to Specify Conditional Number Formats**

You can make number formats conditional by defining number formats for ranges of values, using the following relational operators:

- > (greater than).
- >= (greater than or equal to).
- = (equal to).
- < (less than).
- <= (less than or equal to).

Each range definition must be enclosed within a left bracket ([) and a right bracket (]).

If you use the string format to specify the range of data to which the number format is applied, make sure the format string includes the entire range of data. Separate each range and format from the next with a semicolon (;). For example, the following defines the format:

- #,# for values less than or equal to 9 ([<=9]#,#)
- -#.# for values less than zero ([<0]-#.#)
- # for values equal to zero ([=0]#)
- #.# for values greater than 9 ([>9]#.#)

```
numberFormat:'[<=9]#,#;[<0]-#.#;[=0]#;[<0]-#.#;[>9]#.#'
```

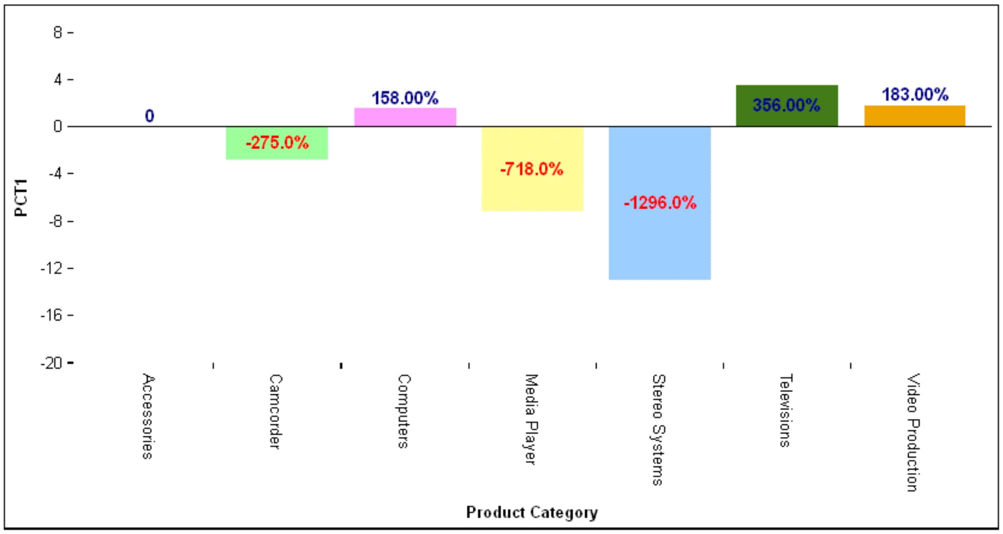
Example: **Specifying Conditional Number Formats**

The following request computes a fraction. The series object enables making the data labels visible. The dataLabels object makes the data labels visible and uses red (the default) for negative values. It also sets number formats as follows. If the number is:

- Greater than 0, the number is displayed as a percent with two decimal places ([>=0]#.## %).
- Less than zero, the number is displayed as a percent with one decimal place and prefixed by a minus sign ([<0]-#.#%).
- Equal to zero, the number is displayed as 0 ([=0]0).

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED NOPRINT
AND COMPUTE
COMPUTE PCT1 = IF DAYSDELAYED NE 763 THEN (500-DAYSDELAYED)/100 ELSE 0;
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series": "all", "showDataValues": true}],
"dataLabels": {"visible": true, "useNegativeColor": true,
    "font": "bold 10pt Sans-Serif", "color": "navy",
    "numberFormat": "[>=0]#.##%;[<0]-#.#%;[=0]0"}
*END
ENDSTYLE
END
```

The output is:



Formatting Numbers Using a Function

The function takes one argument (the number to format) and returns a string. For example:

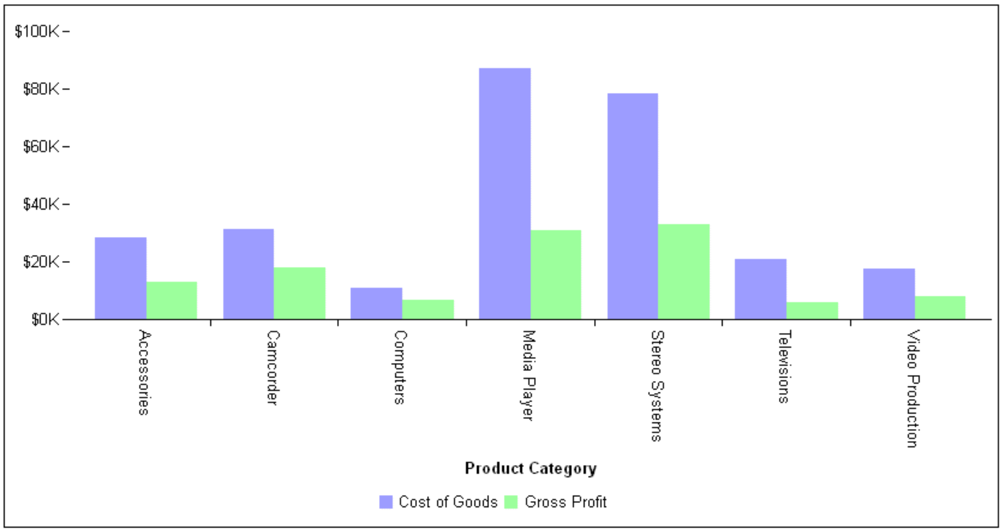
```
// Return number with a "$" in front
numberFormat = function(n){ return "$" + n; };
```

Example: Returning a Formatted Number Using a Function

The following request uses a function to divide the yaxis label by one thousand and return it prefixed with a dollar sign ($) and followed by the letter K:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"yaxis": {
    "numberFormat": function(n){ return "$" + n/1000 + "K";}
}
*END
ENDSTYLE
END
```

The output is:



Reference: Usage Notes for Number Formats

- Except with numberFormat auto, do not use a semicolon (;) as a prefix, suffix, thousands separator, or decimal separator character. A semicolon is only valid is for separating multiple conditional formats (for example, "[>=0]#,#;[<0]-#.#").
- Except with numberFormat auto, do not use a pound sign (#), period (.) or question mark (?) as a prefix or suffix character. These are special characters that cannot be interpreted as a string.
- If a percent sign (%) is used as the prefix or suffix character in a JSON object definition, the number format will use the percent mode regardless of the mode setting.
- To display all numbers in a chart using European number formatting, you can issue the SET CDN command.

Controlling the Number Formats of Tooltips and Labels

The autoNumberFormats properties control the number formatting of data labels, stacked total labels, pie total labels, and tooltips.

Syntax: How to Control the Number Formats of Tooltips and Labels

```
"autoNumberFormats": {
      "object": "numberfmt", ...
}
```

where:

object

Is one of the following objects.

- dataLabels
- tooltip
- stackTotalLabel
- totalLabel

Multiple objects can be included, separated by commas.

numberfmt

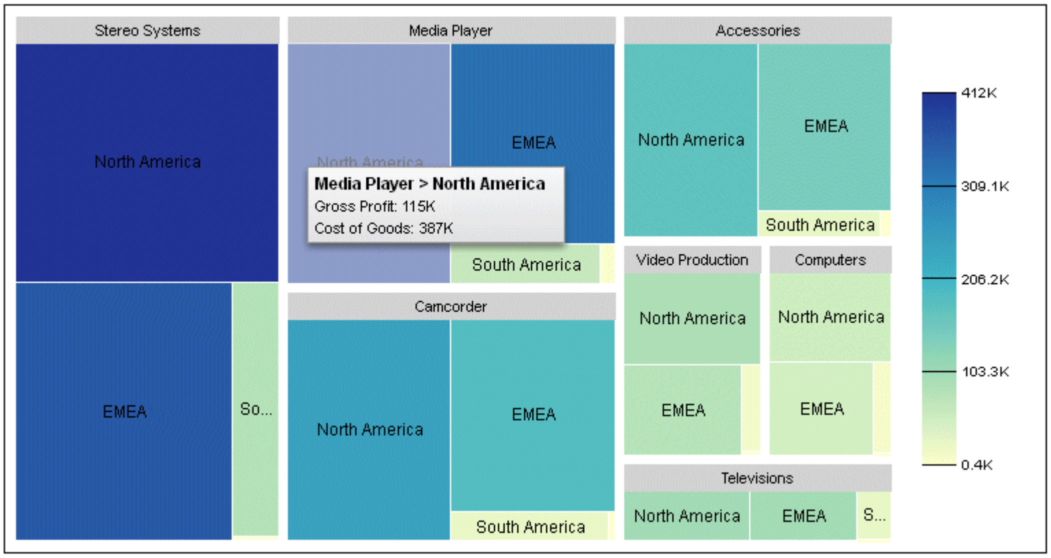
Is a valid number format. For information about number formats, see *Formatting Numbers* on page 108.

Example: Specifying a Number Format for Tooltips

The following request generates a treemap.

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
*END
ENDSTYLE
END
```

The default tooltip number format is shown in the following image.

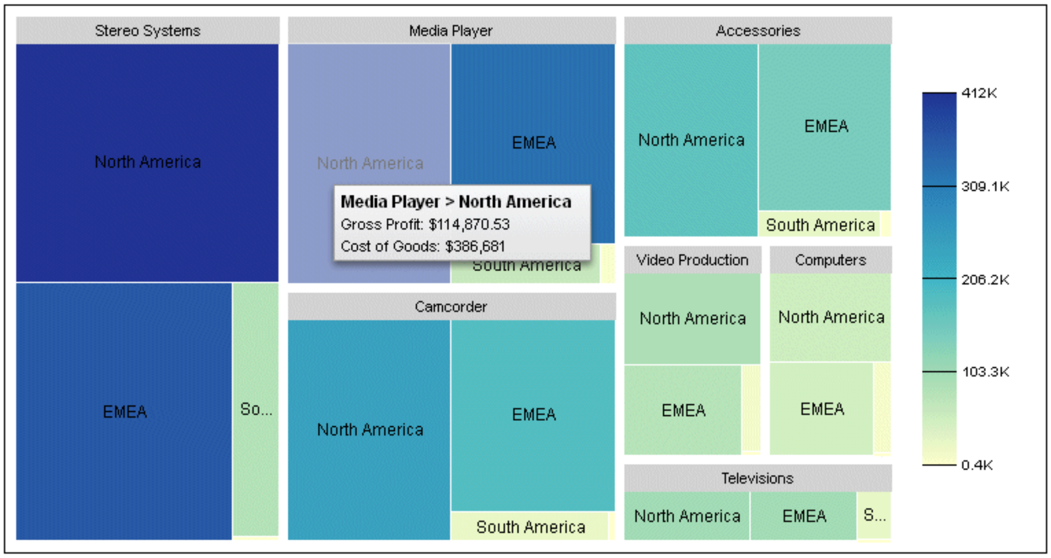


Media Player > North America
Gross Profit: 115K
Cost of Goods: 387K

The following version of the request specifies a number format for the tooltips that includes a dollar sign, a thousands separator, and a decimal separator with two decimal places.

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
"autoNumberFormats": {"tooltip": "$#,###.##"}
*END
ENDSTYLE
END
```

The new tooltip number format is shown in the following image. Note that decimal places are not displayed if they have the value zero.



HTML Codes in Strings

HTML codes can be embedded in any label or title strings. HTML codes in titles and labels are not processed by the library. They are simply passed on to the browser and will appear as they are rendered by the browser. For example:

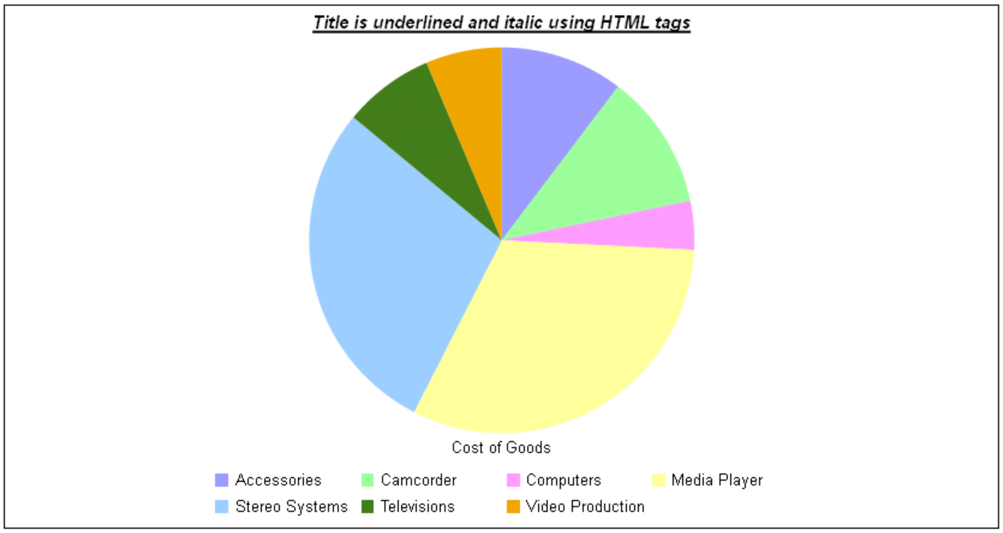
```
"title": {"text": "<u><em>Title is underlined and italic
using HTML tags<\/em><\/u>"},
```

Example: Using HTML Tags in the Chart Title Text

The following request encloses the title text within underline (<u> and </u>) and italic (<em> and </em>) HTML tags:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"title": {"visible": true,
    "text": "<u><em>Title is underlined and italic using HTML tags<\/em><\/u>"}
*END
ENDSTYLE
END
```

On the output, the chart title is underlined and italic:



Data Definitions

If your data source includes missing values, you can use the VZERO parameter to determine how to handle the missing values on the chart.

If VZERO is set to ON, missing data along the y-axis is treated as zero. If it is set to OFF, missing data along the y-axis is ignored, and values are not stored in the plot matrix. OFF is the default.

Setting Chart Property Values

All properties can be assigned a value of *undefined* or *null* (not quoted strings). When *undefined* or *null* is used as a property value, the property is ignored. The default value assigned to the property is used.

Defining a Callback Function to Display Values in Labels

You can define a callback function to return data values, series number, group number, and data object to display in labels and tooltips. Callback functions are common in JavaScript. For complete information on defining and using callback functions, you must refer to JavaScript documentation.

The function can have up to four arguments. The arguments are positional, so the first argument is always the data value, the second is always the series number, and the third is the group number.

Syntax: How to Define a Callback Function to Return Values for Use in Labels

```
formatCallback: function(d,s,g, dataset){return expression;}
```

where:

d

Is the data value argument. It is an object with properties that tell which type of data to return. The type of data returned (for example, the x-value or the y-value) is called the *data mode* and is dependent on the type of chart being generated.

In the expression to be returned by the function, if you are returning data values, you must specify which data mode to return. For example, to return the x-value, specify d.x in the expression.

The following table lists the data modes that can be used for each chart type. Note that any chart types not listed have default modes. These may include value, size, and color:

Chart Type	Data Modes
Boxplot	min, lower, median, upper, max
Polar	phi, r, size, color
Bubble	x, y, size
Scatter	x, y

Chart Type	Data Modes
Treemap	size, color
Control	value, sampleSize
Map	name, value (used when there are two arguments) or lng, lat, color (used when there are three arguments)

s

Is the argument that returns the series number.

g

Is the argument that returns the group number.

dataset

Is the set of data for the series for this point. For some chart types, it can be the entire data set, and for others it is undefined. It is mostly useful for calculating the data series sum for stacked bars and pies.

expression

Is an expression supported in a JavaScript function.

In addition to the arguments passed into the function, you can insert text and HTML tags into the expression, enclosed in single quotation marks ('). You can also use other objects, properties, operators, and functions supported in JavaScript functions.

Each component in the return value is separated from the next component by a plus sign (+). The expression must be terminated with a semicolon (;).

In addition, you can declare new variables and use any CSS style in a callback function. You can also retrieve the properties of any object passed as a parameter. For example, to retrieve the color property of series 0, when the series argument is called s, you can use the following:

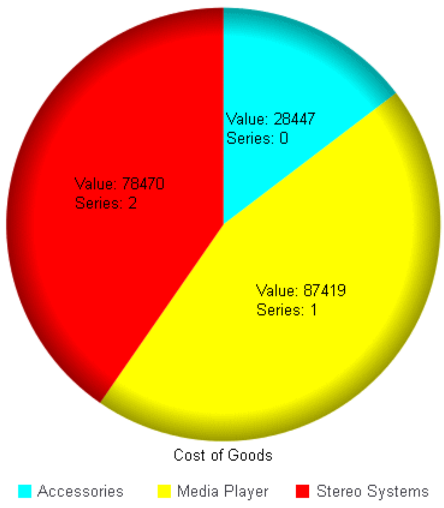
```
"series": 0, "color": "green",
              "tooltip": function(d,s,g){
                       var c = this.getSeries(s).color;
```

Example: Defining a Callback Function to Return Data Labels for a Pie Chart

The following request generates a pie chart and shows the data labels. The labels are returned by a callback function which has two arguments, a (data) and b(series). In addition to returning the data value (a.value), the function returns text, which describes what is being displayed. It also includes an HTML line-break tag (<br>) to make the label display on two lines:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Stereo Systems'
OR 'Media Player'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"series": [{series: "all", "showDataValues": true},
    {"series": 0, "color": "cyan", "showDataValues": true},
    {"series": 1, "color": "yellow", "showDataValues": true},
    {"series": 2, "color": "red", "showDataValues": true}
],
dataLabels: {
    "visible": true,
    "position": "middle",
    "formatCallback": function(a,b){return "Value: " + a.value +
        "<br>Series: " + b ;}
}
*END
ENDSTYLE
END
```

The output is:



Example: **Defining Custom Tooltips With a Callback Function**

The following request generates a stacked bar chart with two series. The tooltip for each series is returned by a callback function.

The callback function for series 0 retrieves the color and label for series 0 and returns those properties, using the series color as the text color for the tooltip. The <span object is a CSS style that formats the return values from the callback function, The backslash (\) in front of the double quotation mark (") is an escape character that causes the callback function to return the double quotation mark instead of using it as a special character. The tooltip contains breaks (<br/>) to generate multiple lines.

```
 "series": [
{"series": 0, "color": "green",
 "tooltip": function(d,s,g){
 var c = this.getSeries(s).color;
 var lbl = this.getSeries(s).label;
 return '<span style=\"color: ' + c + '\">value: ' + d + '<br />series:
' + s + '<br />group: ' + g + '<br />series name: ' + lbl +'</span>'; }
     }
```

The callback function for series 1 retrieves the color and label from series 1. It also retrieves the data values from both series and adds them together to get a total value:

```
{"series": 1, "color": "blue",
        tooltip: function(d,s,g){
        var c = this.getSeries(s).color;
        var lbl = this.getSeries(s).label;
        var total =this.data[0][g].value +
        this.data[1][g].value;
        return '<span style=\"color: ' + c + '\">value: ' + d + ' out of '
+
    total + '<br />series name: ' + lbl +'</span>';}
                    }
```

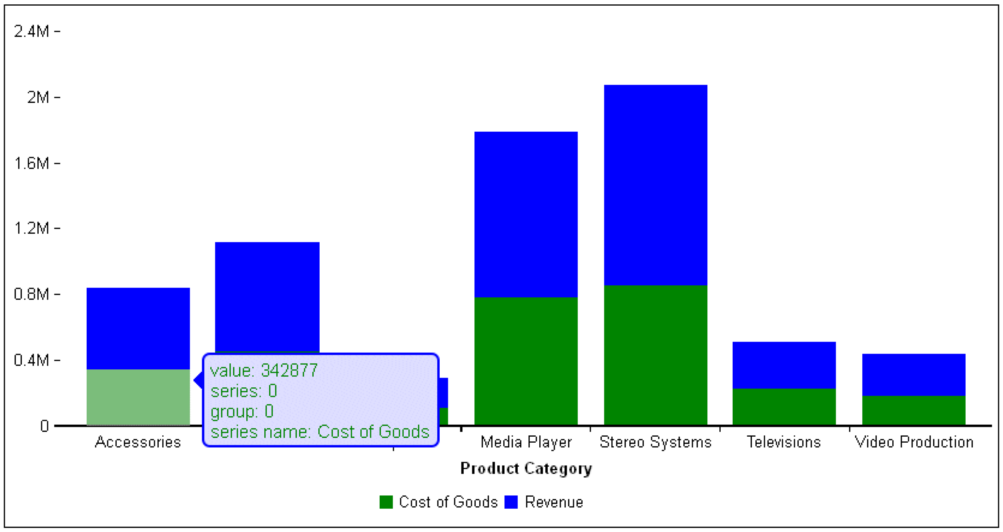
The tooltip is visualized as an HTML tooltip with a blue border and a lighter blue (#DDDDFF) fill color.

Note: The return values are enclosed in single quotation marks. Although the string may break onto separate lines on the page in this manual, it must be all on one line in the request. Therefore, in order to run the example, you must remove the breaks in the return values.

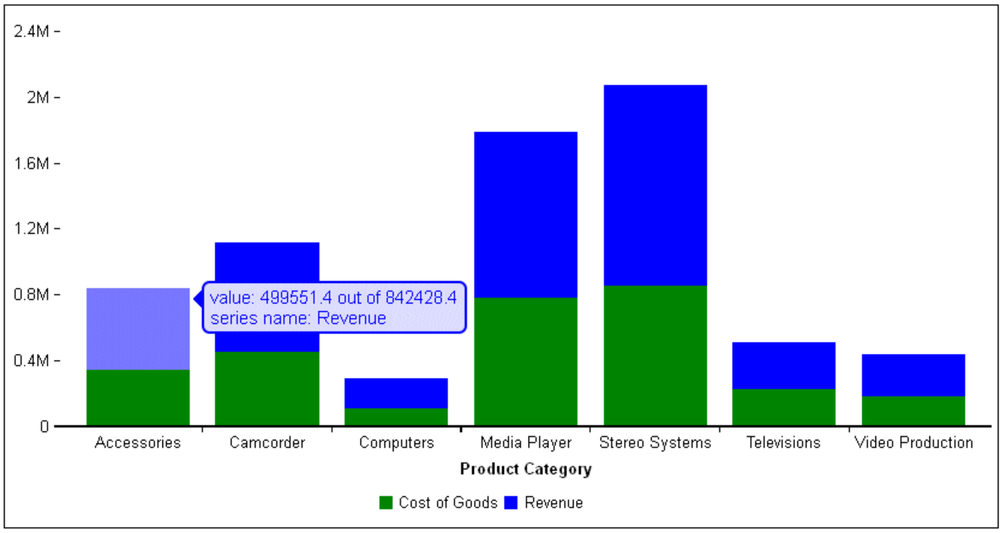
The request follows:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBRSTK1
ON GRAPH SET STYLE *
*GRAPH_JS
"xaxis":{"labels": {"rotation": 0}},
"series": [
    {"series": 0, "color": "green",
        tooltip: function(d,s,g){var c = this.getSeries(s).color;
            var lbl = this.getSeries(s).label;
            return '<span style=\"color: ' + c + '\">value: ' + d + '<br />series: ' +
            s + '<br />group: ' + g + '<br />series name: ' + lbl +'</span>'; }
    },
    {"series": 1, "color": "blue",
        tooltip: function(d,s,g){var c = this.getSeries(s).color;
            var lbl = this.getSeries(s).label;
            var total =this.data[0][g].value +
            this.data[1][g].value;
            return '<span style=\"color: ' + c + '\">value: ' + d + ' out of ' +
            total + '<br />series name: ' + lbl +'</span>';}}],
"htmlToolTip": {"snap": true, "fill": "#DDDDFF", "border": {"color": "blue"}}
ENDSTYLE
END
```

The following image shows a tooltip for series 0:



The following image shows a tooltip for series 1:



Using the Chart Template Engine to Customize Tooltips, Titles, and Data Labels

A template is a special string of characters that the chart engine will dynamically replace with chart component information when it renders the chart. A template is made up of macros. Macros are strings enclosed in a double set of curly braces. Each macro is parsed and replaced independently to build a final string. The chart engine includes many pre-defined macros. These macros can be used in data text labels, tooltips, chart titles, axis titles, and legend titles. In addition, they can be inserted in the heading of a WebFOCUS request. Although WebFOCUS cannot process these macros, it passes the heading to the chart engine for processing.

Macros can be parameterized. Parameters are a comma-separated list of strings, delimited with parentheses. The built in row_label and col_label macros are parameterized, so you can choose which row or column label to include. The first row is row zero (0). For example:

```
{{row_label(0)}}
```

Some macros apply to chart attribute syntax. For information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

When defining custom data text labels, you can use template macros to display chart component information.

Since data text labels are automatically populated with default values, you need to include the following properties to clear out the default formatting and you must place the custom dataLabels in a GRAPH_JS_FINAL block in the StyleSheet.

```
"dataLabels": {"visible":true,"formatCallback": null, "displayMode": null}
```

Reference: Template Macros

The following lists the pre-defined macros for the chart template engine:

Macro	Definition
auto_percent	Provides percentage values in pie charts, stacked bar charts, treemap chart inner-groupings, and mekko charts.
auto_tooltip_content	Automatically generated tooltip text
col_id	Column number in a matrix chart
col_label	Column label in a matrix chart

Macro	Definition
coloraxis_title	
data_page_slider_current_index	Index position of the slider
data_page_slider_current_label	Label of the current slider position
group_count	Number of groups
group_id	Group number
group_label	Group label
group_percent	Percent for each series within the group
group_sum	Series sum for current group
metadata_value	Applies the numeric format from the metadata to data labels and tooltips For more information, see *Applying the Numeric Format From the Metadata to Data Labels and Tooltips* on page 137.
misc_id	
nested_label	Group label
object_id	
row_id	Row number in a matrix chart
row_label	Row label in a matrix chart
series_count	Number of series in the request
series_id	Series number
series_label	Series label
series_percent	Percent of current series within all groups.
series_sum	Sum of current series for all groups.

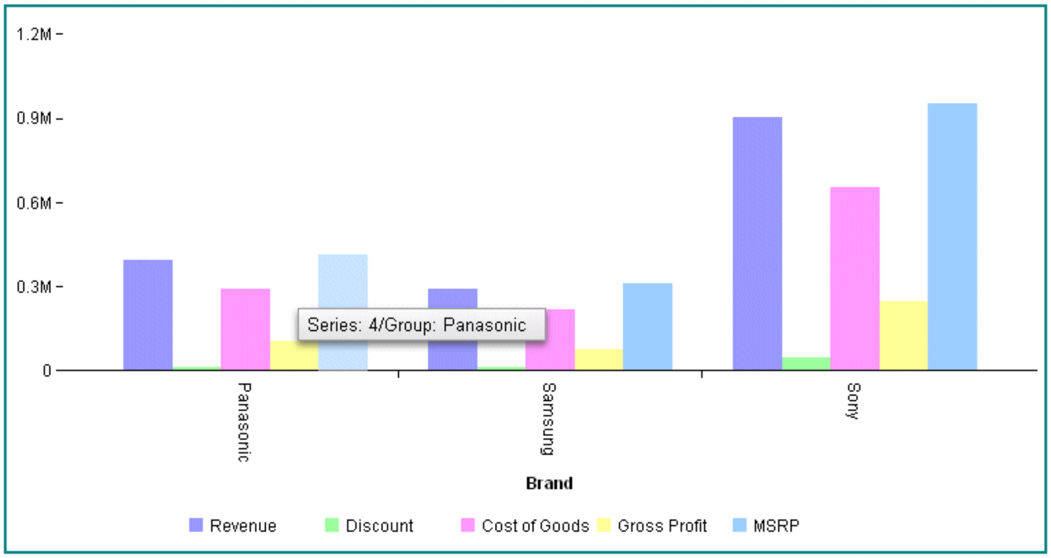
Macro	Definition
tooltip*n*	The *n*th field in the tooltip attribute category.
value	The value represented by the series when using chart attribute syntax.
xaxis_title	x-axis title
yaxis_title	y-axis title
zaxis_title	z-axis title

Example: Using the Template Engine in Tooltips

The following request generates tooltips that contain the series id and the group label:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US DISCOUNT_US COGS_US GROSS_PROFIT_US MSRP_US
BY BRAND
WHERE BRAND EQ 'Sony' OR 'Samsung' OR 'Panasonic'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"series": [{"series": "reset",
    "tooltip": "Series: {{series_id}}/Group: {{group_label}}"}]
*END
ENDSTYLE
END
```

The tooltip for series 4 in group 0 is shown in the following image:



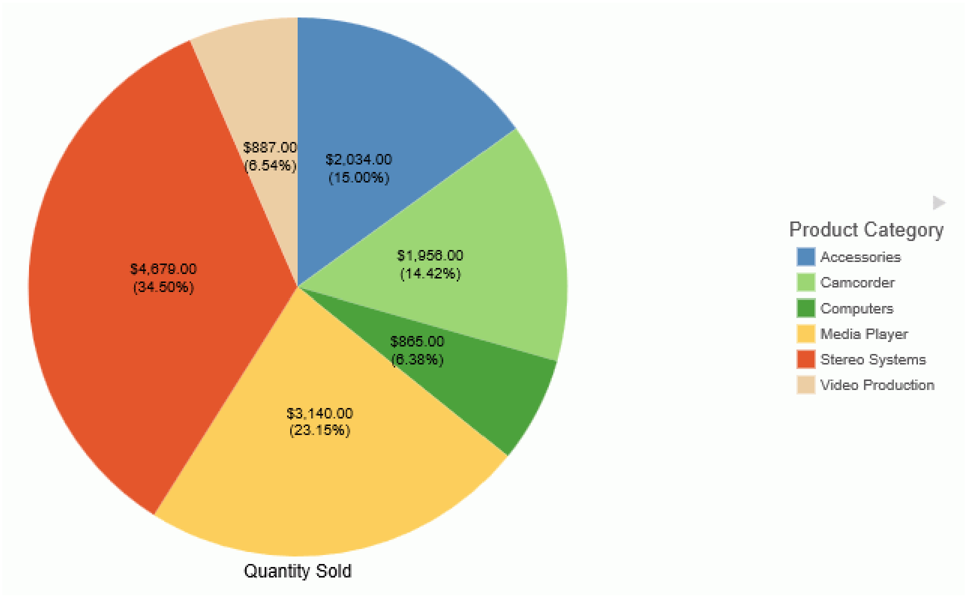
Note: If your request uses chart attribute syntax, you must put tooltip customization properties in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

Example: Using a Template Macro in a Data Text Label

The following request generates a pie chart. The series data text labels display the sum of quantity sold as a number formatted with a dollar sign and two decimal places. It uses the HTML string \n to go to a new line and displays the percentage using the auto_percent template macro.

```
GRAPH FILE WF_RETAIL_LITE
SUM QUANTITY_SOLD
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY NE 'Televisions'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=color, $
TYPE=DATA, COLUMN=N2, BUCKET=measure, $
*GRAPH_JS_FINAL
"dataLabels": {"visible":true,"formatCallback": null, "displayMode": null},
"series":[{"series": "reset",
 "dataLabels":{"content":"{{value | formatNumber('$#,###.00')}}
\n({{auto_percent}})"}}],
*END
ENDSTYLE
END
```

The output is shown in the following image.

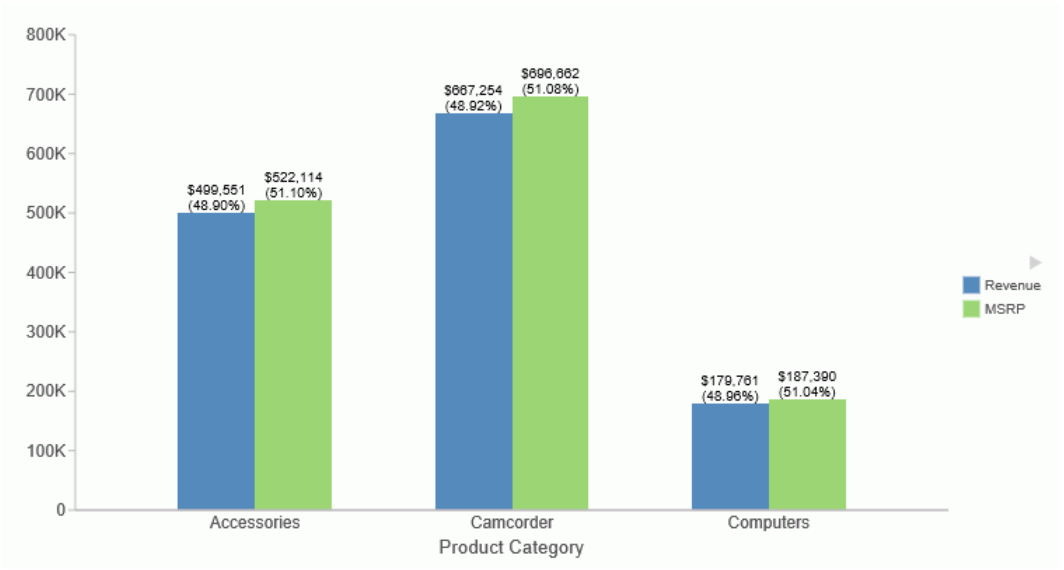


Example: Using a Template Macro in a Data Text Label for a Bar Chart

The following request generates a bar chart. The series data text labels display the revenue and MSRP as numbers formatted with a dollar sign and no decimal places. It uses the HTML string \n to go to a new line and displays the percent of each series within its group using the group_percent template macro.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Computers' OR 'Camcorder'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"dataLabels": {"visible":true,"formatCallback": null, "displayMode": null},
"series":[{"series": "reset",
 "dataLabels":{"content":"{{value | formatNumber('$#,###')}}\n({{group_percent}})"}}],
*END
ENDSTYLE
END
```

The output is shown in the following image.

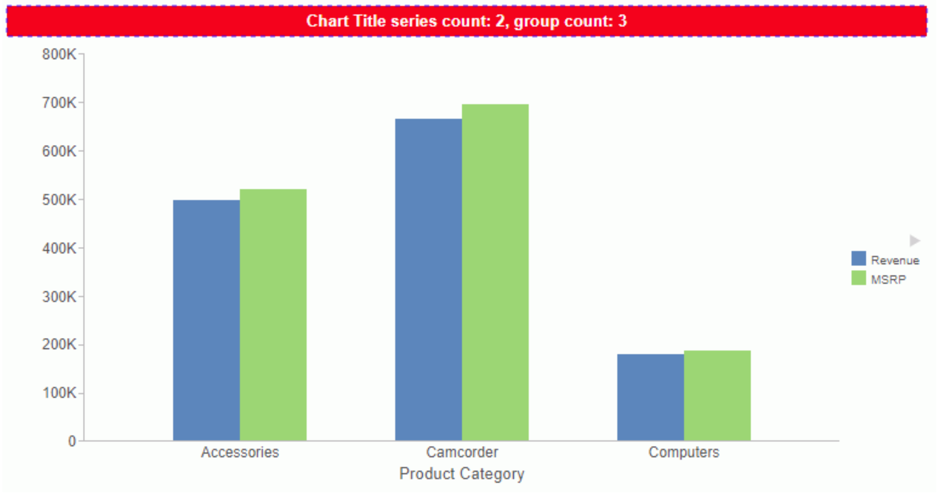


Example: Using Template Macros in a Chart Title

The following request adds the series_count and group_count macros to the chart title.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Computers' OR 'Camcorder'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"title": {
"visible": true,
"text": "Chart Title series count: {{series_count}}, group count:
{{group_count}} ",
"color": "white",
"backgroundColor": "red",
"border":
        {
        "width": 1,
        "color": "blue",
        "dash": "4 4"
        }
},

*END
ENDSTYLE
END
```

The output is shown in the following image.



Applying the Numeric Format From the Metadata to Data Labels and Tooltips

The metadata_value chart template macro applies the numeric format found in the WebFOCUS metadata to a chart object. If a format is assigned in the GRAPH FILE procedure, that format will be used, otherwise the format from the Master File will be used.

Syntax: How to Apply the Format From the Metadata to a Data Label or Tooltip

```
{{metadata_value(data_array_entry)}}
```

where:

data_array_entry

Is an entry in the data array map. Data array map entries are dependent on the type of chart, and each has a default entry. You can use the default entry without knowing its name by specifying *default* as the data array entry. Following is a list of entries by chart type.

- **Boxplot.**
  - "min"
  - "lower"
  - "median" (default)
  - "upper"

  - "max"
- **Polar.**
  - "phi"
  - "r" (default)
  - "size"
  - "color"
- **Bubble.**
  - "x"
  - "y"
  - "size" (default)
- **Scatter.**
  - "x" (default)
  - "y"
  - "pie"
- **Choropleth.**
  - "name"
  - "value" (default)
- **Bubblemap.**
  - "lng"
  - "lat"
  - "value" (default)
  - "color"
- **Matrix Marker.**
  - "size" (default)
  - "color"

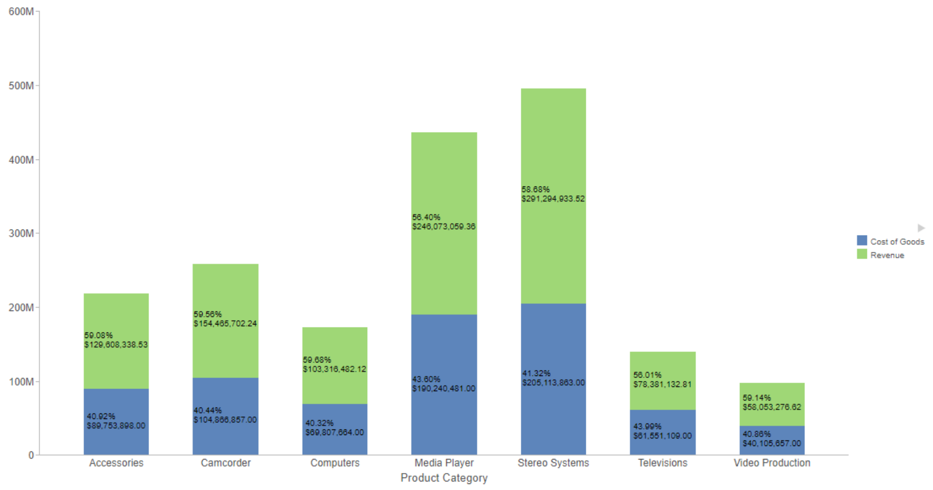
- **Treemap.**
  - "size" (default)
  - "color"
- **Heatmap.**
  - "color" (default)
- **Tagcloud.**
  - "value" (default)
  - "color"
- **Bullet.**
  - "value" (default)
  - "markers"
- **Default.** (Charts not otherwise listed that have value, size, and color properties, such as bar, line, and pie).
  - "value" (default)
  - "size"
  - "color"

Example: Displaying Data Labels Using the Format From the Master File

The following request creates a stacked bar chart in which the content for the data labels consists of two lines. The first line uses the auto_percent macro to display the percent of each bar in the stack, then adds a line break. The second line displays the series value for each bar in the stack, using the numeric display format from the Master File (D20.2M). This format displays the number with a dollar sign ($) to the left of the highest significant digit and with two decimal places.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART

ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, ORIENTATION=LANDSCAPE, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_SCRIPT

*GRAPH_JS_FINAL
"series":     [
   {"series": "all",
        "border": {
                "color": "rgba(0,0,0,0)"
            },
    "dataLabels": {
    "visible": true,
    "position": "center",
    "content": '{{auto_percent}} <br> {{metadata_value("value")}}'

},
        }
    ],
"blaProperties": {
    "seriesLayout": "stacked"
},


*END
ENDSTYLE
END
```

The output is shown in the following image.



Chapter 4

WebFOCUS Chart Attribute Syntax

Chart attribute syntax uses WebFOCUS StyleSheet attributes to assign roles to dimensions and measures in an HTML5 request. Using this syntax, you can generate chart types that are not available without it, and display multiple charts in a matrix array. This syntax also uses a simplified list of chart types and provides the ability to add sliders, pages, and additional sorts to certain types of charts.

When using chart attribute syntax, if a component is missing, partial output will display. For example, if a bar chart does not have a field assigned to the y-axis, all bars will be the same height.

You can use JSON properties and API calls in conjunction with chart attribute syntax to control additional properties of the chart output.

In this chapter:

Chart Attribute Syntax Concepts

Chart attribute syntax uses TYPE=DATA declarations in a WebFOCUS StyleSheet to assign the fields in a request to attribute categories. Some examples of attribute categories are *x-axis*, *y-axis*, *size*, and *color*.

The categories available depend on the chart type being generated and the number of sort fields and measure fields in the request. Sort fields must be assigned to categories in a specific order. This order depends on the order of sort fields in the request and on the types of categories being used. For more information about sort order and attribute categories, see *Order of Attribute Category Assignments for Sort Fields* on page 221.

Note: Only BY sort fields are supported with this syntax. ACROSS sort fields are not supported.

Understanding How Fields Are Assigned to Attribute Categories

When you use chart attribute syntax, the WebFOCUS StyleSheet contains declarations for each field in a request (TYPE=DATA declarations). These declarations assign fields to the appropriate attribute categories. The StyleSheet can also contain global (TYPE=REPORT) declarations that set the chart orientation and layout (for bar, line, and area charts).

For complete syntax, see *How to Assign Chart Data to Attribute Categories* on page 211. For information on assigning chart orientation and layout, see *Specifying Chart Orientation* on page 205 and *Specifying Chart Layout for Bar, Line, and Area Charts* on page 207.

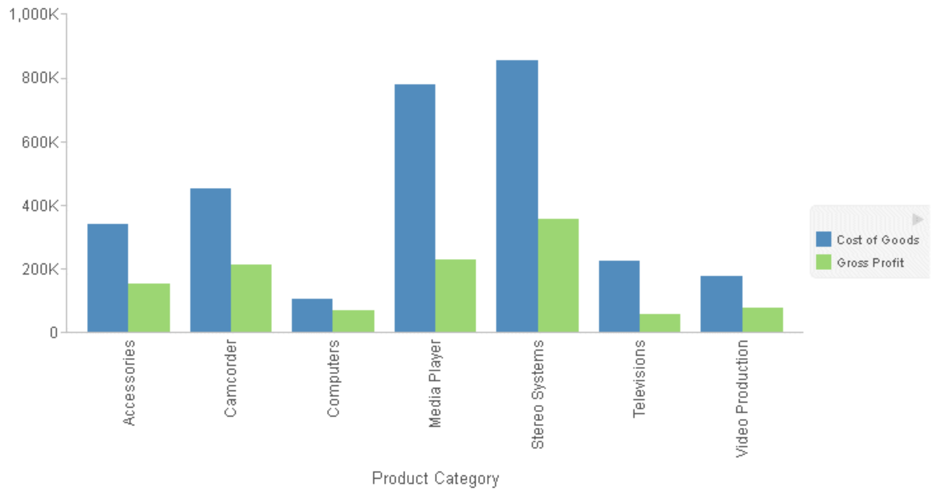
Following is a simple example of a bar chart using chart attribute syntax.

Example: Sample Bar Chart Using Chart Attribute Syntax

The following request generates a bar chart. The sort field (PRODUCT_CATEGORY) is assigned to the x-axis, and both measures (COGS_US and GROSS_PROFIT_US) are assigned to the y-axis. The chart type is BAR.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Reference: Understanding Column Sequence Numbers

Every WebFOCUS request generates an output data record in which each column is associated with a sequence number. Some fields in a request generate multiple output columns, and some are not displayed on the output. These extra columns will be ignored in this discussion.

You can use these sequence numbers to refer to fields in a WebFOCUS StyleSheet. Sort fields, starting with the highest level sort come first, then measure fields. The fields are referred to as N1, N2, N3, and so on in the StyleSheet.

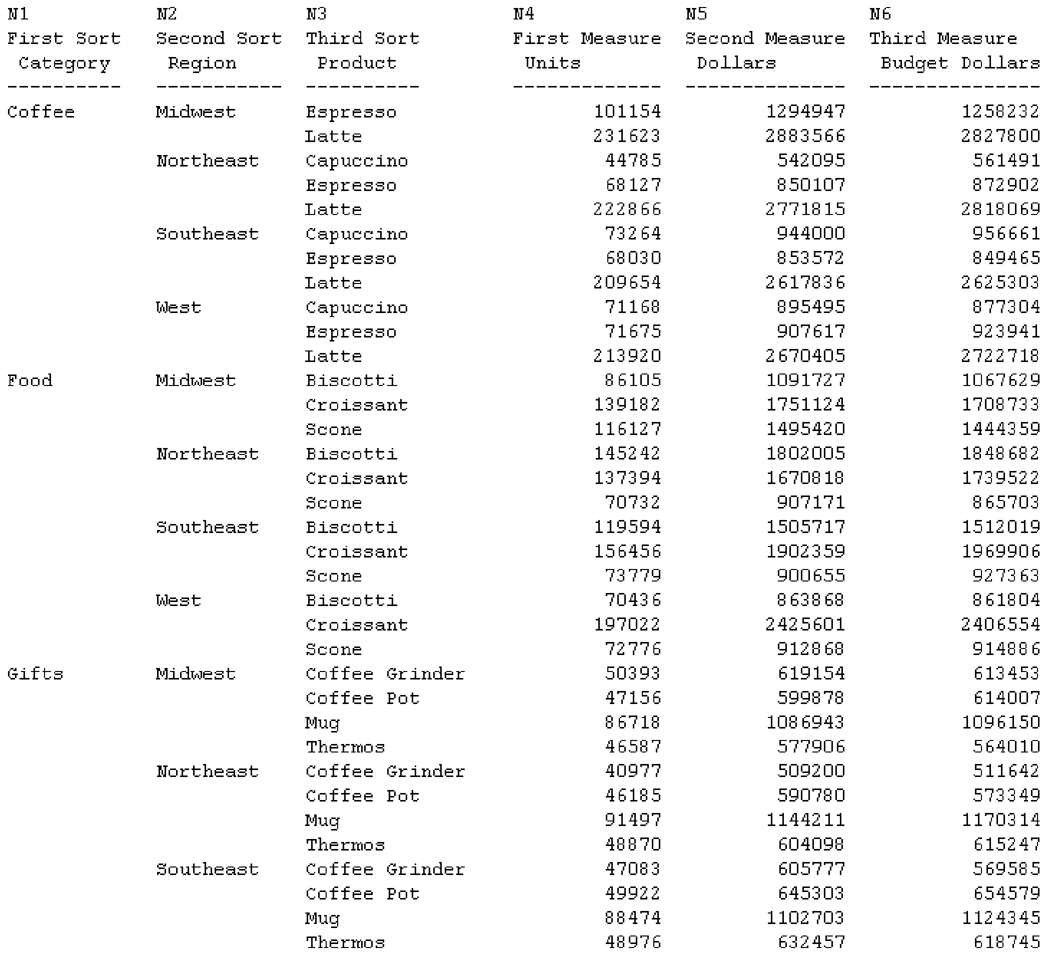
Note: If you use one of the WebFOCUS tools to generate your charts, it will assign fields to categories using column sequence numbers, not field names.

For more information about WebFOCUS StyleSheet syntax and column sequence numbers, see the *Creating Reports With WebFOCUS Language* manual.

To see how columns are sequenced, consider the following simple request that has three sort fields and three measures:

```
TABLE FILE GGSALES
HEADING
"N1           N2           N3              N4             N5              N6"
SUM UNITS AS 'First Measure, Units'
DOLLARS AS 'Second Measure, Dollars'
BUDDOLLARS AS 'Third Measure, Budget Dollars'
BY CATEGORY AS 'First Sort, Category'
BY REGION AS 'Second Sort, Region'
BY PRODUCT AS 'Third Sort, Product'
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT WP
END
```

The partial output shown in the following image illustrates the arrangement of fields on the output and the sequence number assigned to each. The first (high-order) sort field is column 1, the second sort field is 2, and so on. After all sort fields are displayed, the measure fields are displayed in their order in the request:



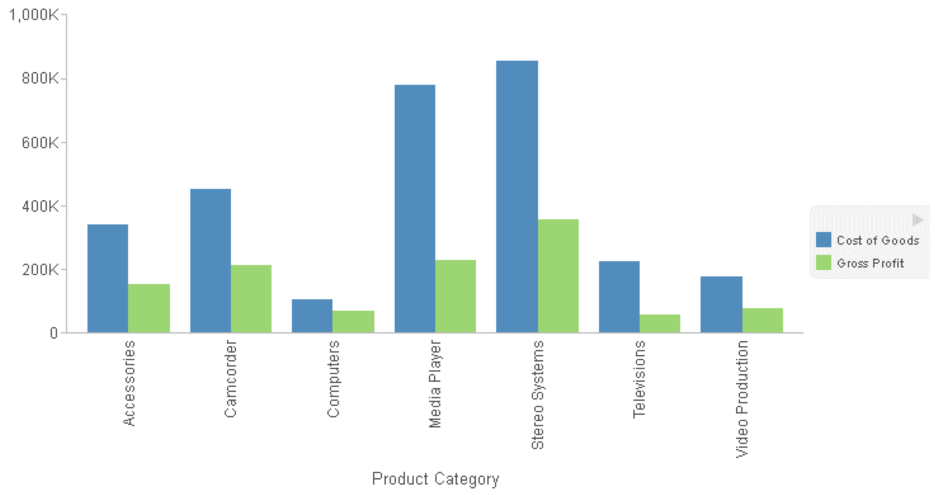
```
N1           N2           N3                  N4             N5              N6
First Sort   Second Sort  Third Sort          First Measure  Second Measure  Third Measure
 Category     Region       Product             Units          Dollars         Budget Dollars
----------   -----------  ----------          -------------  --------------  ----------------
Coffee       Midwest      Espresso                   101154         1294947          1258232
                          Latte                      231623         2883566          2827800
             Northeast    Capuccino                   44785          542095           561491
                          Espresso                    68127          850107           872902
                          Latte                      222866         2771815          2818069
             Southeast    Capuccino                   73264          944000           956661
                          Espresso                    68030          853572           849465
                          Latte                      209654         2617836          2625303
             West         Capuccino                   71168          895495           877304
                          Espresso                    71675          907617           923941
                          Latte                      213920         2670405          2722718
Food         Midwest      Biscotti                    86105         1091727          1067629
                          Croissant                  139182         1751124          1708733
                          Scone                      116127         1495420          1444359
             Northeast    Biscotti                   145242         1802005          1848682
                          Croissant                  137394         1670818          1739522
                          Scone                       70732          907171           865703
             Southeast    Biscotti                   119594         1505717          1512019
                          Croissant                  156456         1902359          1969906
                          Scone                       73779          900655           927363
             West         Biscotti                    70436          863868           861804
                          Croissant                  197022         2425601          2406554
                          Scone                       72776          912868           914886
Gifts        Midwest      Coffee Grinder              50393          619154           613453
                          Coffee Pot                  47156          599878           614007
                          Mug                         86718         1086943          1096150
                          Thermos                     46587          577906           564010
             Northeast    Coffee Grinder              40977          509200           511642
                          Coffee Pot                  46185          590780           573349
                          Mug                         91497         1144211          1170314
                          Thermos                     48870          604098           615247
             Southeast    Coffee Grinder              47083          605777           569585
                          Coffee Pot                  49922          645303           654579
                          Mug                         88474         1102703          1124345
                          Thermos                     48976          632457           618745
```

Example: Sample Bar Chart Using Column Sequence Numbers

The following request generates the same bar chart as the previous example, but using column sequence numbers. The sort field (PRODUCT_CATEGORY, field N1) is assigned to the x-axis, and each measure field is assigned to the y-axis. The chart type is BAR:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Overriding Server-Generated Properties

When you use chart attribute syntax in a request, the WebFOCUS Reporting Server generates some of the text that displays on the output. For example, the server generates row labels, column labels, axis labels, and tooltips. The server generates these labels after the chart engine has finished rendering the chart.

You can change the default row, column, and axis labels in the request using an AS phrase (for example SUM COGS_US AS 'Cost'). This causes the server to use your custom labels instead of its default labels.

You can also override server-generated properties by placing the affected properties in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For certain components, such as tooltips, the only way to override default server properties is to add a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. This tells the chart engine that the properties in that block should be rendered after server processing has been completed.

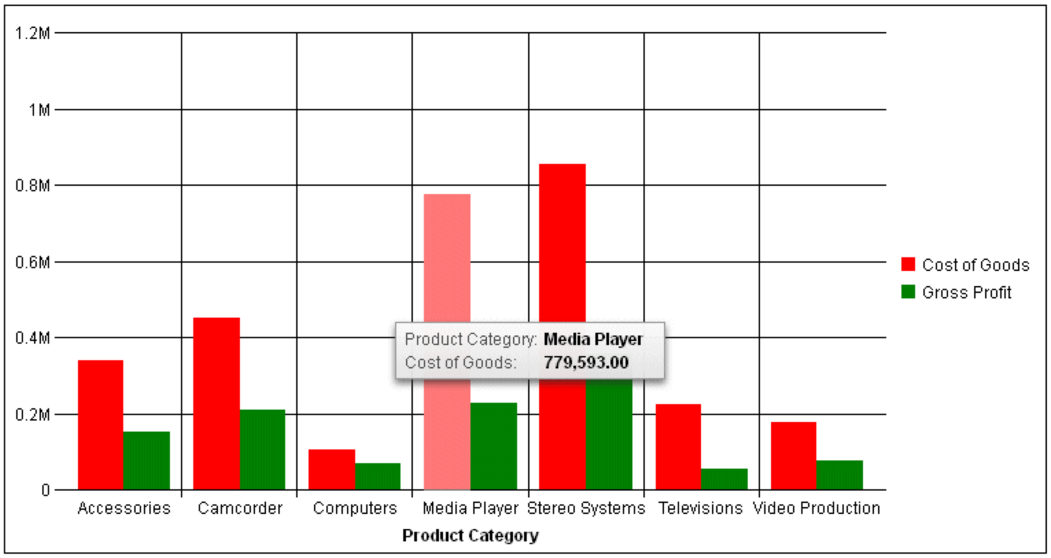
The WebFOCUS StyleSheet can have multiple *GRAPH_JS and *GRAPH_JS_FINAL blocks. The *GRAPH_JS blocks will be processed in their order in the StyleSheet, prior to server processing. The *GRAPH_JS_FINAL blocks will be processed in their order in the StyleSheet, after server processing is complete.

Example: Generating Custom Tooltips With Chart Attribute Syntax

The following request generates a bar chart and contains a customized tooltip:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US , BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"series": [
    {"series": "reset", "tooltip": "This is my custom tooltip"}
]
*END
ENDSTYLE
END
```

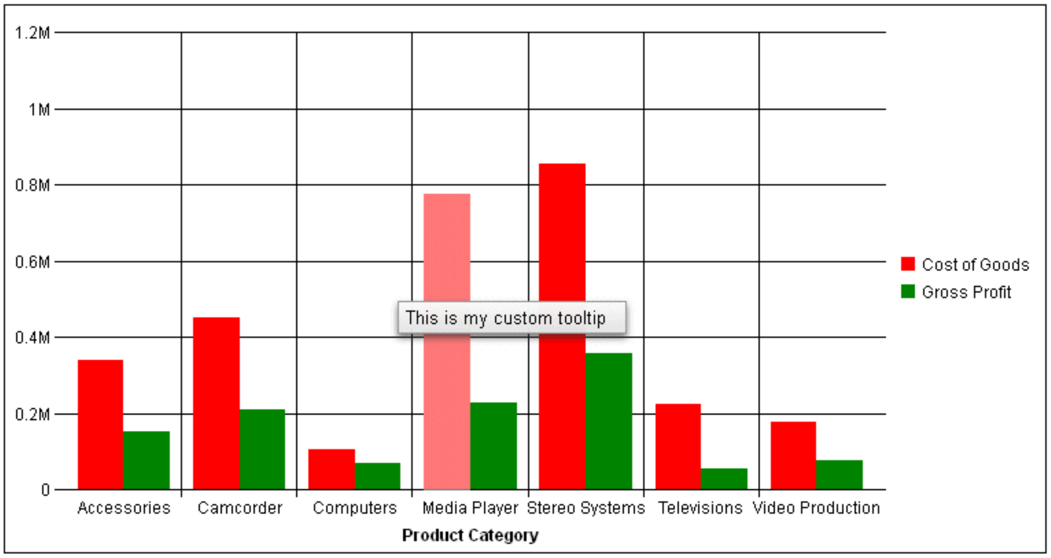
The custom tooltip text is overridden by server-generated properties as shown in the following image:



The following is the same request with a *GRAPH_JS_FINAL block instead of a *GRAPH_JS block in the WebFOCUS StyleSheet:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US , BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS_FINAL
"series": [
    {"series": "reset", "tooltip": "This is my custom tooltip"}
]
*END
ENDSTYLE
END
```

The custom tooltip displays, as shown in the following image:



You can use the chart template engine to customize tooltips with macros, as described in *Introduction to JSON Properties for HTML5 Charts* on page 83.

Displaying Multiple Charts in a Matrix Array

In addition to assigning fields in the request to specific chart components, you can create multiple charts by assigning fields to the *row* and *column* categories in order to generate a matrix of charts.

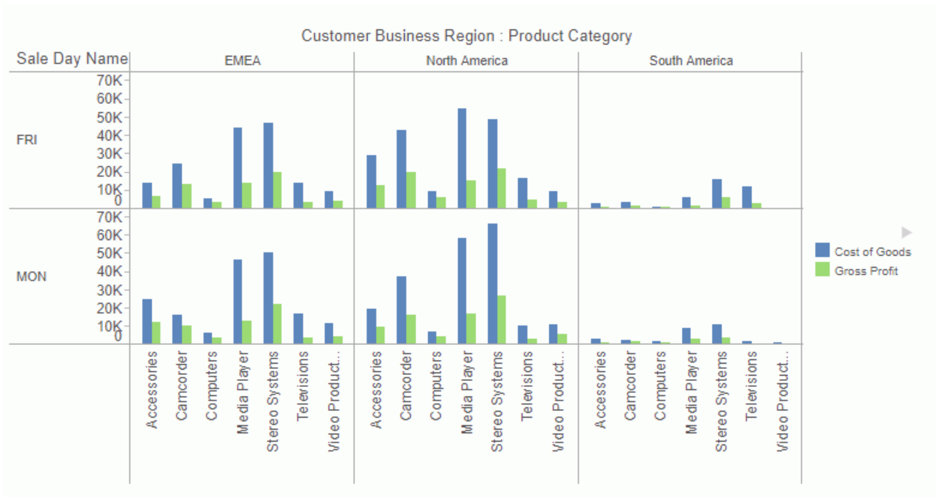
You can assign multiple fields to the row and column attribute categories, and you can specify only rows or only columns, to create finely tuned groups of charts.

Example: Creating A Matrix of Bar Charts

The following request generates a new bar chart for each combination of business region and day name:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Matrix Marker Charts

Although all matrix charts require the use of chart attribute syntax, one type of chart, called a *matrix marker* chart, was developed specifically to use with the matrix capabilities of this syntax.

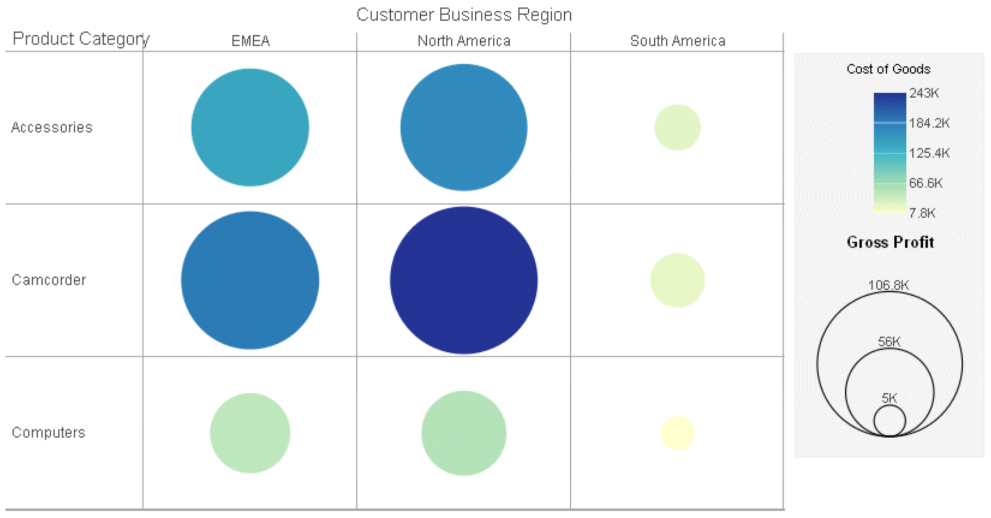
A matrix marker chart is a graphical representation of tabular data. It displays a marker in each cell of the matrix. This marker can be sized by one measure and colored by a second measure. The LOOKGRAPH value is MARKER.

Example: Creating a Matrix Marker Chart Using Chart Attribute Syntax

The following request generates a matrix marker chart. The rows represent product categories. The columns represent business regions. The marker in each cell is sized by the gross profit measure and colored by the cost of goods measure. When the field assigned to the *color* category is a measure, the legend is visualized as a gradient:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
WHERE PRODUCT_CATEGORY LE 'D'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH MARKER
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=size, $
ENDSTYLE
END
```

The output is shown in the following image:



Generating Data Grids

A data grid is a tabular representation of data, similar to a tabular report. The grid can have multiple rows, columns, and measures. Unlike the tabular reports generated with the WebFOCUS TABLE FILE command, data grids are generated in a GRAPH FILE request using PCHOLD FORMAT JSCHART. The LOOKGRAPH value is DATAGRID.

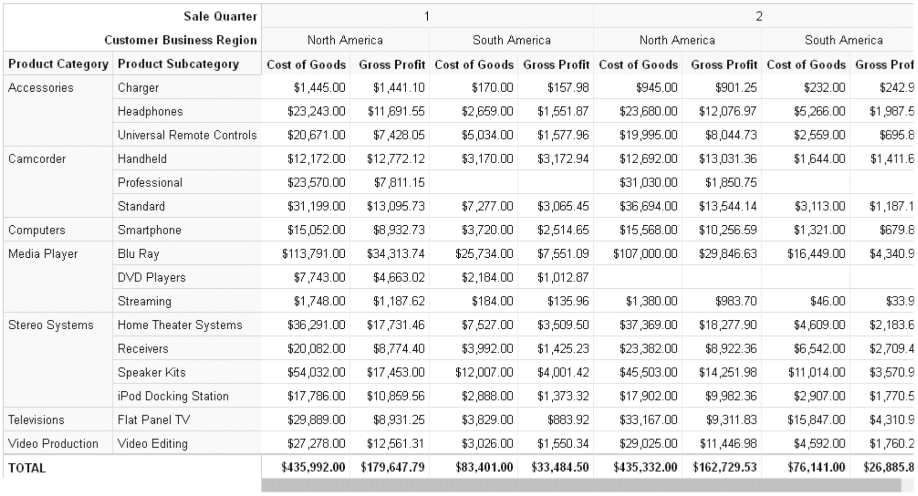
Using the data grid properties, you can format the grid and display column totals for measures in the grid. The rows and columns in a data grid are numbered starting with zero (0). For information about properties you can use to format data grids, see *Chart-Specific Properties* on page 491.

Example: Generating a Data Grid

The following request generates a data grid with column totals. The rows represent product categories and product sub-categories. The columns represent the first two quarters of the year and the North America and South America business regions. The measures are cost of goods and gross profit.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY TIME_QTR
BY BUSINESS_REGION
WHERE (BUSINESS_REGION NE 'Oceania' OR 'EMEA') AND (TIME_QTR EQ 1 OR 2)
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
type=data, column=product_category, bucket=row, $
type=data, column=product_subcateg, bucket=row, $
type=data, column=time_qtr, bucket=column, $
type=data, column=business_region, bucket=column, $
type=data, column=cogs_us, bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {
    "columnTotals": {"visible": true}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



	Sale Quarter	1				2			
	Customer Business Region	North America		South America		North America		South America	
Product Category	**Product Subcategory**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Prof**
Accessories	Charger	$1,445.00	$1,441.10	$170.00	$157.98	$945.00	$901.25	$232.00	$242.9
	Headphones	$23,243.00	$11,691.55	$2,659.00	$1,551.87	$23,680.00	$12,076.97	$5,266.00	$1,987.5
	Universal Remote Controls	$20,671.00	$7,428.05	$5,034.00	$1,577.96	$19,995.00	$8,044.73	$2,559.00	$695.8
Camcorder	Handheld	$12,172.00	$12,772.12	$3,170.00	$3,172.94	$12,692.00	$13,031.36	$1,644.00	$1,411.6
	Professional	$23,570.00	$7,811.15			$31,030.00	$1,850.75		
	Standard	$31,199.00	$13,095.73	$7,277.00	$3,065.45	$36,694.00	$13,544.14	$3,113.00	$1,187.1
Computers	Smartphone	$15,052.00	$8,932.73	$3,720.00	$2,514.65	$15,568.00	$10,256.59	$1,321.00	$679.8
Media Player	Blu Ray	$113,791.00	$34,313.74	$25,734.00	$7,551.09	$107,000.00	$29,846.63	$16,449.00	$4,340.9
	DVD Players	$7,743.00	$4,663.02	$2,184.00	$1,012.87				
	Streaming	$1,748.00	$1,187.62	$184.00	$135.96	$1,380.00	$983.70	$46.00	$33.9
Stereo Systems	Home Theater Systems	$36,291.00	$17,731.46	$7,527.00	$3,509.50	$37,369.00	$18,277.90	$4,609.00	$2,183.6
	Receivers	$20,082.00	$8,774.40	$3,992.00	$1,425.23	$23,382.00	$8,922.36	$6,542.00	$2,709.4
	Speaker Kits	$54,032.00	$17,453.00	$12,007.00	$4,001.42	$45,503.00	$14,251.98	$11,014.00	$3,570.9
	iPod Docking Station	$17,786.00	$10,859.56	$2,888.00	$1,373.32	$17,902.00	$9,982.36	$2,907.00	$1,770.5
Televisions	Flat Panel TV	$29,889.00	$8,931.25	$3,829.00	$883.92	$33,167.00	$9,311.83	$15,847.00	$4,310.9
Video Production	Video Editing	$27,278.00	$12,561.31	$3,026.00	$1,550.34	$29,025.00	$11,446.98	$4,592.00	$1,760.2
TOTAL		**$435,992.00**	**$179,647.79**	**$83,401.00**	**$33,484.50**	**$435,332.00**	**$162,729.53**	**$76,141.00**	**$26,885.8**

Adding Levels of Detail to a Bubble or Scatter Chart

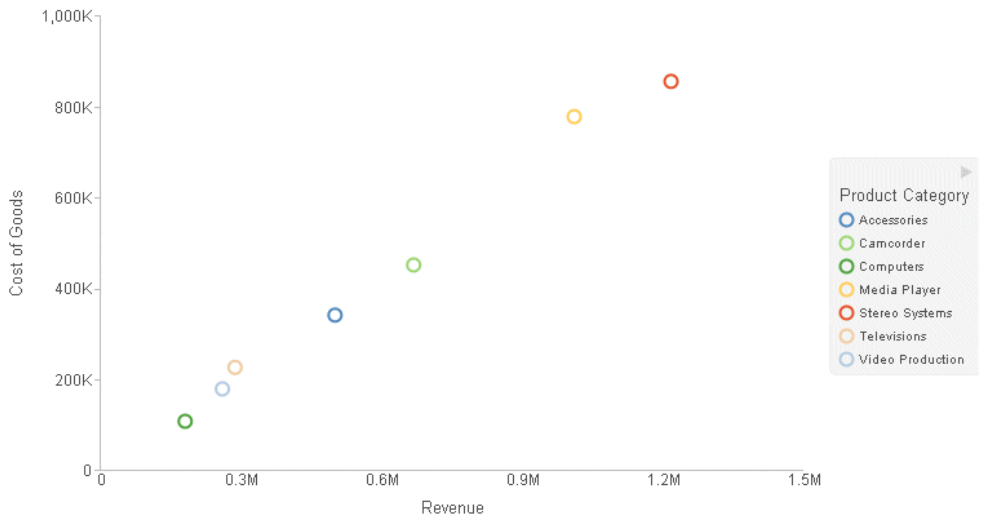
For bubble and scatter charts, you can add additional low-level sort fields to the request and assign these fields to the *detail* category. The effect of these additional sort fields is to add more data points to the chart.

Example: Adding a Detail Category to a Scatter Chart

The following request generates a scatter chart showing cost of goods compared to revenue. The markers are colored by the product category:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
ENDSTYLE
END
```

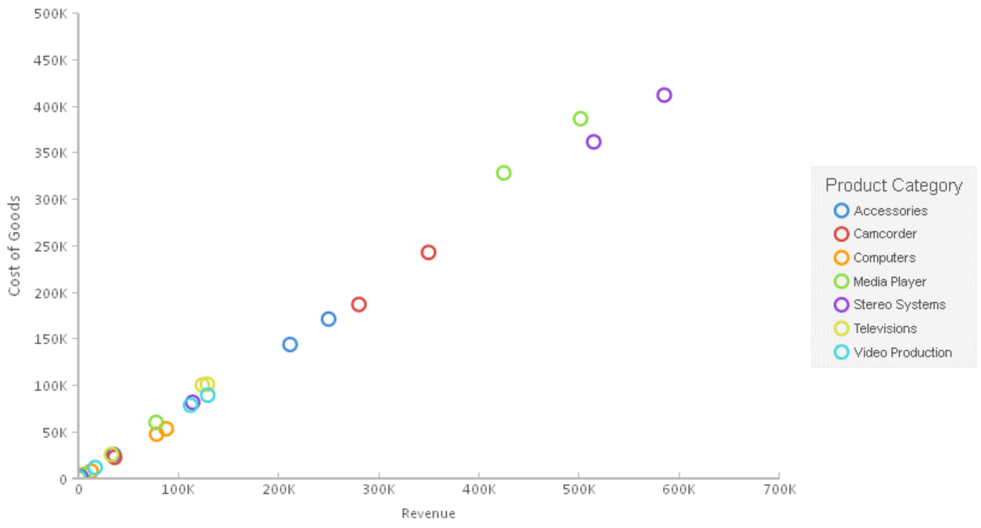
The output is shown in the following image:



Adding the business region field as a low-level sort field and assigning it to the *detail* category adds an additional level of detail to the chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY PRODUCT_CATEGORY
  BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=detail, $
ENDSTYLE
END
```

The output is shown in the following image:



Adding Color-By and Size-By Fields

For some types of charts, such as bubble charts, a size category is inherent in the chart type. The bubble size varies with the value of the measure assigned to the *size* category.

Some chart types have no inherent *size* category. For example, the height of a bar in a bar chart shows the size of the measure it represents. The measures that represent bar heights are assigned to the *y-axis* category.

However, with chart attribute syntax, you can add an additional measure and assign it to the *size* category. This size will be reflected in the *width* of the bars.

In addition, bars are normally assigned a color based on the series they represent. With chart attribute syntax, you can add an additional field, assign it to the *color* category, and have the bars colored by the value of that additional field. If a color-by field is a measure, the bar and legend colors will be visualized as a gradient. If a color-by field is a dimension, the bar and legend colors will be visualized as discrete colors.

Line charts can also have *size* and *color* categories when using chart attribute syntax. If the markers are visible, their size is varied to represent the *size* measure value. If the markers are not visible, the line width is varied to represent the *size* measure value. The color-by field can be a measure or dimension field. The color is reflected in the markers, lines, and legend. If a color-by field is a measure, the colors are visualized as a gradient. If it is a dimension, the colors are visualized as discrete colors.

Pie charts displayed in a matrix array can also be sized by a measure added to the request and assigned to the *size* category. The size of the pie chart in each cell of the matrix will represent the value of this additional measure.

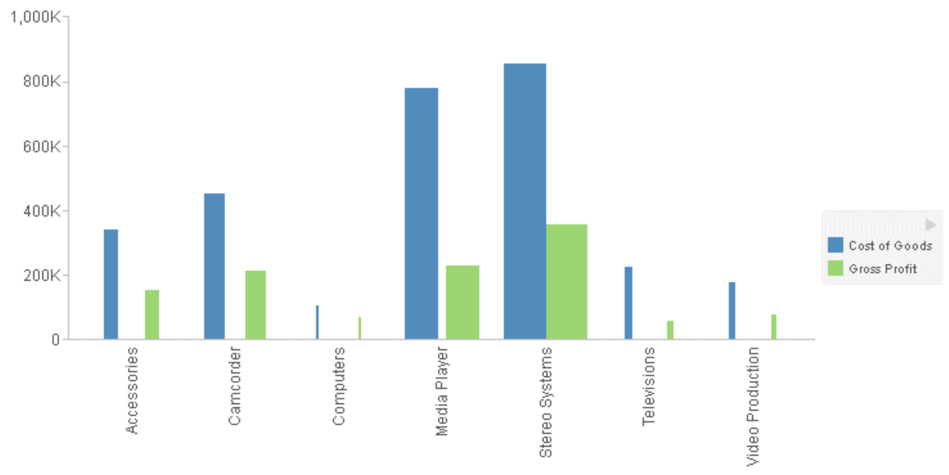
For information about each supported chart type, see *Chart Types for Chart Attribute Syntax* on page 171. For a table listing the attribute categories supported for each type of chart, see *Summary of Supported Attribute Categories for Each Chart Type* on page 222.

Example: Using a Size-By Measure in a Bar Chart

The following request generates a bar chart. The bar heights represent the COGS_US and GROSS_PROFIT_US measures, and the bar widths represent the MSRP_US measure.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US  MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=MSRP_US, BUCKET=size, $
ENDSTYLE
END
```

The output is shown in the following image:

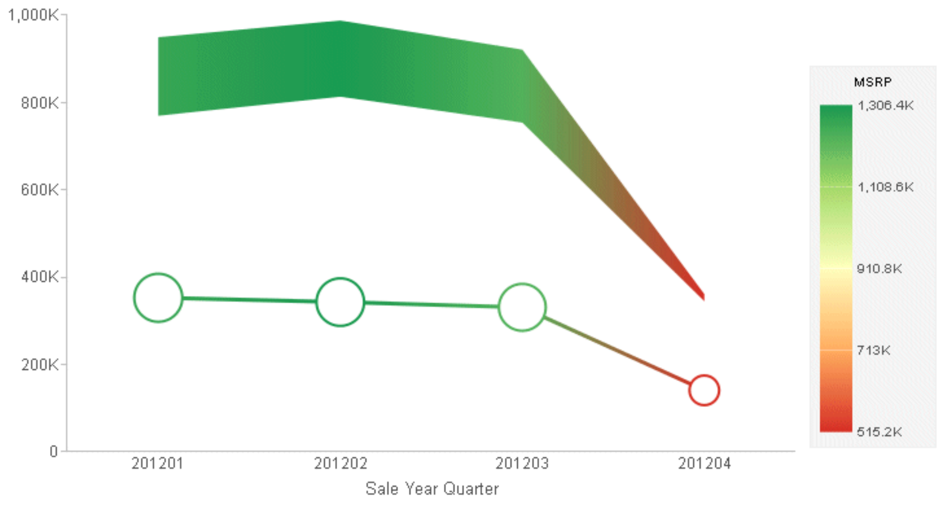


Example: Using Color-By and Size-By Measures in a Line Chart

The following request generates a line chart. The lines and markers are colored by the MSRP_US measure. The markers for series 1 are sized by the SIZE1 virtual measure. For series 0, the markers are not visible, so the line width represents the SIZE1 virtual measure:

```
DEFINE FILE WF_RETAIL_LITE
SIZE1 WITH WF_RETAIL_SALES.COGS_US =
    IF WF_RETAIL_TIME_SALES.TIME_YEARQTR LT 200902 THEN 5
    ELSE IF WF_RETAIL_TIME_SALES.TIME_YEARQTR LT 200903 THEN 50
    ELSE 200;
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US SIZE1
BY WF_RETAIL_TIME_SALES.TIME_YEARQTR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=WF_RETAIL_TIME_SALES.TIME_YEARQTR, BUCKET=x-axis, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=MSRP_US, BUCKET=color, $
TYPE=DATA, COLUMN=SIZE1, BUCKET=size, $
*GRAPH_JS
"series": [
    {"series": 0, "marker": {"visible": false}},
    {"series": 1, "marker": {"visible": true, "shape": "circle"}}]
*END
ENDSTYLE
END
```

The output is shown in the following image:

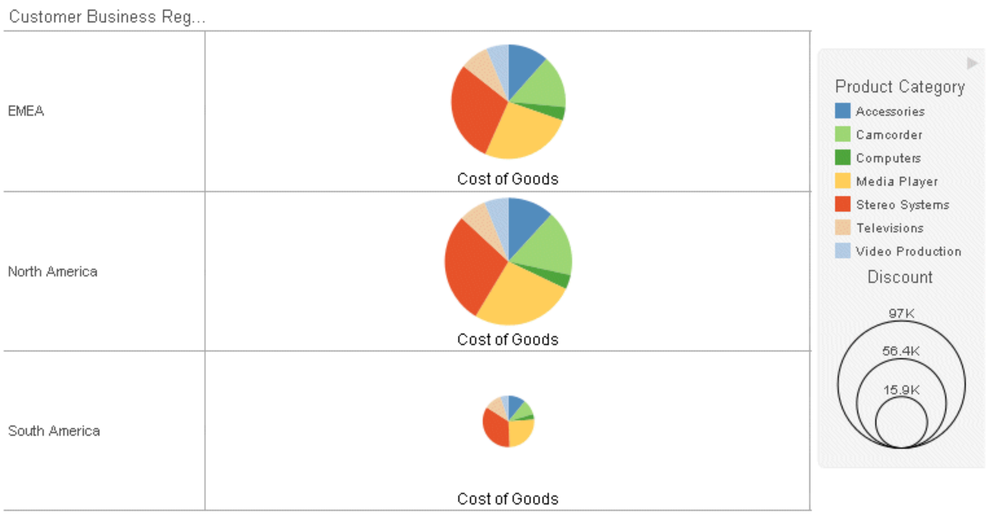


Example: Sizing Pie Charts in a Matrix

The following request generates pie charts in a matrix whose rows represent business regions. The pie chart in each row is sized by the DISCOUNT_US measure.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DISCOUNT_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=row, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure, $
TYPE=DATA, COLUMN=DISCOUNT_US, BUCKET=size, $
ENDSTYLE
END
```

The output is shown in the following image:



Animating Time Progression Using a Slider Control

A slider control on a chart creates an animation effect as the control moves along the slider bar. You can drag the control along the slider bar or click the slider play button to make the slider automatically move from value to value. A slider is limited to one sort field only and should preferably be something time or sequence related such as *YEAR*.

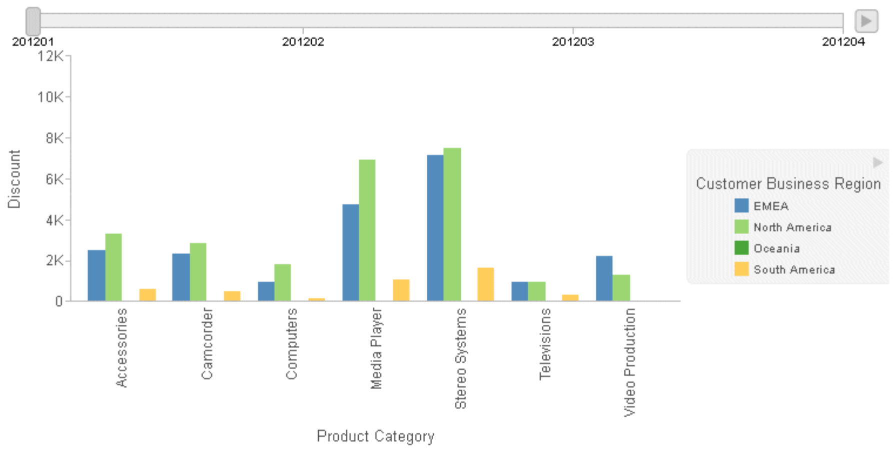
You can control the speed at which the slider moves when the play button is clicked. For information, see *Special Topics* on page 739.

Example: Adding a Slider Control to a Chart

The following request generates a bar chart with one measure for each product category. There is a slider control assigned to the TIME_YEARQTR field and a color-by field assigned to the BUSINESS_REGION dimension. These fields have been added to the request as the highest level sort fields. A new bar chart will be generated for each quarter as the slider moves from one quarter to the next:

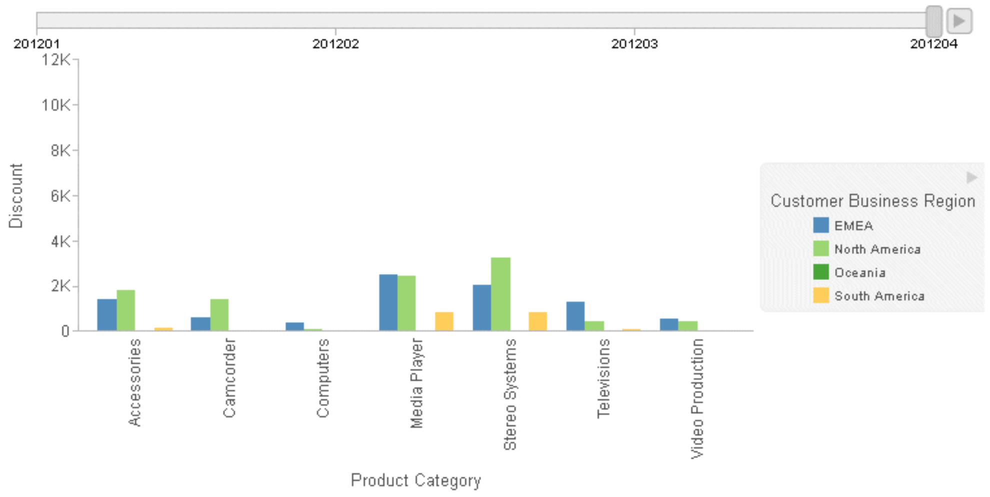
```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US
BY WF_RETAIL_TIME_SALES.TIME_YEARQTR
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=DISCOUNT_US , BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=WF_RETAIL_TIME_SALES.TIME_YEARQTR, BUCKET=slider, $
ENDSTYLE
END
```

The output is shown in the following image. Initially, the slider control is positioned at the left:



You can click the control and drag it, or you can click the right arrow to animate the progression of the slider through the sort field values from left to right.

The following image shows the control dragged to Quarter 4:



Adding Fields to the Tooltip

Fields that represent chart components are automatically included in the default tooltip. You can add additional fields to the tooltip using the *tooltip* attribute category.

Formatting of the value in the tooltip is derived from the format of the field in the metadata and request.

Items can be added to the tooltip attribute category in any order. They will be displayed in the order in which they are added to the tooltip category, below the fields that are automatically displayed.

If you place a sort field in the tooltip attribute category, the value of the sort field that is displayed in the tooltip is the value that corresponds to the riser on which the tooltip is displayed.

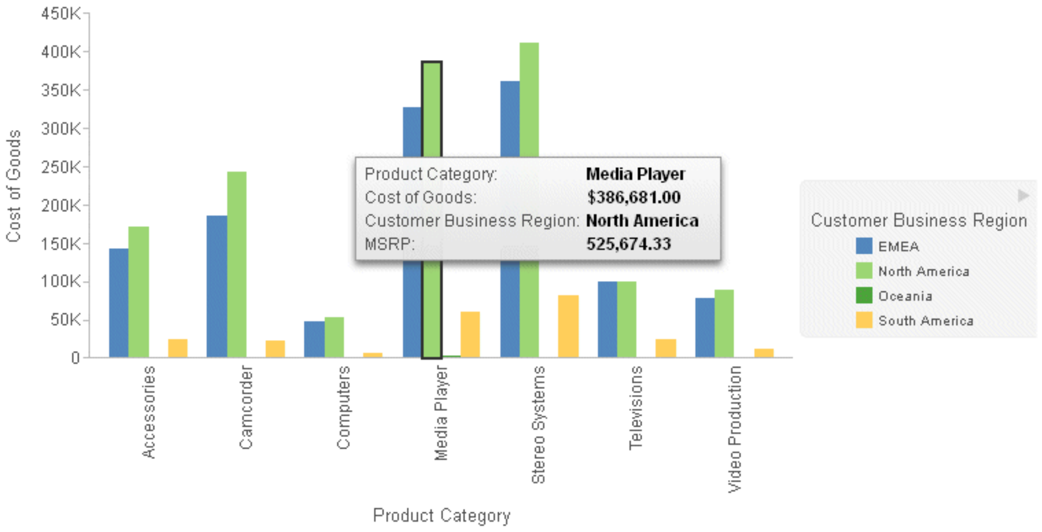
Note: Sort fields represent chart components that are automatically included in the tooltip, so you do not normally have to add them. This includes sort fields assigned to the page, row, and column attribute categories.

Example: Adding a Field to the Tooltip

The following bar chart shows cost of goods for each product category, colored by the business region. The manufacturer suggested retail price is added to the tooltip:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=MSRP_US, BUCKET=tooltip,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image. The MSRP value displays in the tooltip, but is not represented by a bar on the chart:



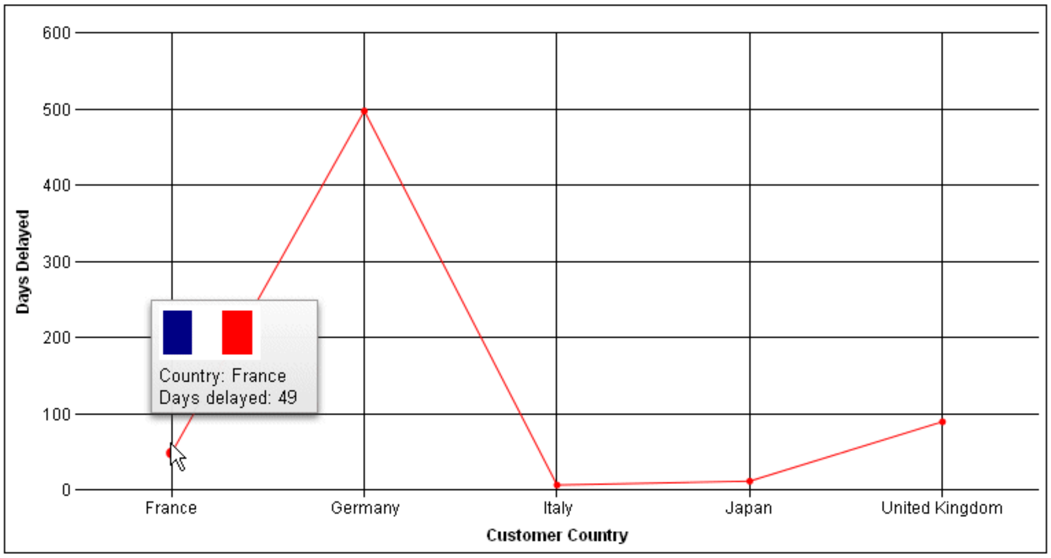
Example: Using the Chart Template Engine to Add an Image to the Tooltip

The following request adds a flag image for each country to the tooltip. If creates a DEFINE field to associate the image names with the country names in the data source. If they were the same, this step would not be needed. In the request, the DEFINE field named FLAG is added to the tooltip attribute category. This value can be inserted into the tooltip using the {{tooltip1}} macro, as it is the first field in the tooltip category. For information about the chart template engine and built-in macros, see *Introduction to JSON Properties for HTML5 Charts* on page 83.

An HTML img tag is added to the custom tooltip in the *GRAPH_JS_FINAL block of the WebFOCUS StyleSheet:

```
DEFINE FILE WF_RETAIL_LITE
FLAG/A12=
DECODE COUNTRY_NAME ( 'United Kingdom' 'uk' 'Germany' 'germany'
    'Italy' 'italy' 'France' 'france' 'Japan' 'japan' );
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED FLAG
BY COUNTRY_NAME
WHERE COUNTRY_NAME EQ 'United Kingdom' OR 'Italy' OR 'Germany' OR 'France' OR
'Japan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
type=data, column=country_name, bucket=x-axis, $
type=data, column=n2, bucket=y-axis, $
type=data, column=flag, bucket=tooltip, $
*GRAPH_JS_FINAL
"series": [{"series": 0,
    "tooltip": "<img src=\"http://ecl.informationbuilders.com/jschart/
        {{tooltip1}}.gif\"><br>Country: {{group_label}}<br>Days delayed: {{value}}"
    }]
END
```

The tooltip for France is shown in the following image:



Adding an Image With a Drilldown Link to a Heading

Chart attribute syntax supports fields with HTML in embedded headings. Embedded headings are implemented with the command ON GRAPH SET EMBEDHEADING ON.

A field with HTML can be included in an embedded heading, since for embedded headings HTML chart titles are generated.

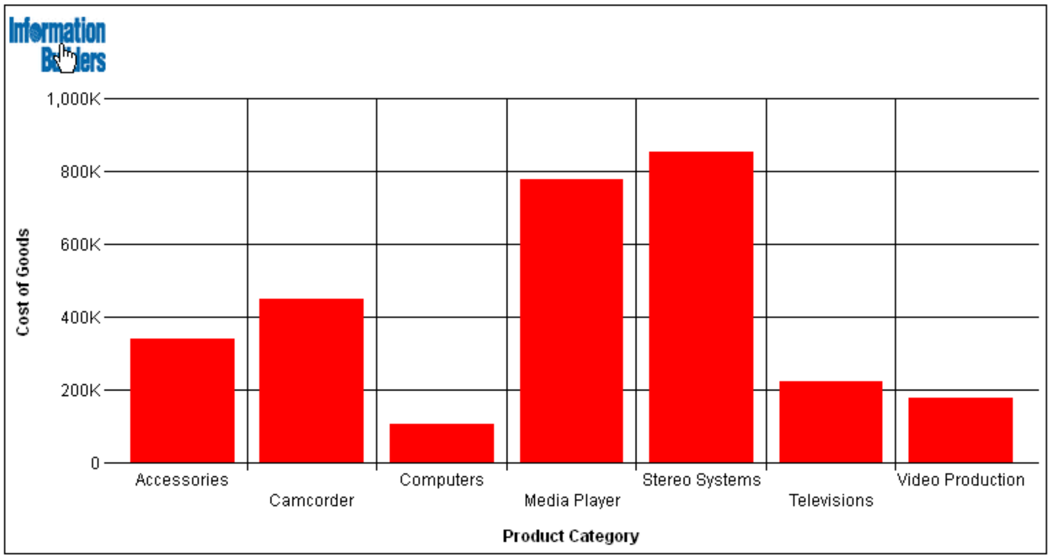
Example: Adding an Image With a Link to a Chart Title

The following request defines a field that contains an HTML image and a link from the image to the Information Builders web site. It then adds this field to the heading. The heading is embedded in the chart using the ON GRAPH SET EMBEDHEADING ON command.

Note: The page width of this document causes the HTML code to wrap onto multiple lines in the example below. In order to run the example, you must remove any line breaks within the single quotation marks.

```
DEFINE FILE WF_RETAIL_LITE
IMG_TAG/A300 =
'<a href="http://www.informationbuilders.com"><img src="http://
ecl.informationbuilders.com/jschart/ibilogo_blue.gif" width=70 height=40></a>';
END
GRAPH FILE WF_RETAIL_LITE
HEADING
" "
"<IMG_TAG"
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET STYLE *
type=data, column=product_category, bucket=x-axis, $
type=data, column=cogs_us, bucket=y-axis, $
END
```

The output is shown in the following image. The hand-shaped cursor pointing to the image indicates that the image has a link.



Binding Colors to Field Values in a Chart

Color binding enables you to associate (*bind*) BY field values to particular colors so that the same value will appear in the same color on different charts, regardless of the sequence or filter conditions. The BY field for which color binding is specified must be assigned to the *color* attribute category. For example, if the BY field is COUNTRY, that field should be assigned to the color attribute category. Then you can specify that the value *France* of the COUNTRY field should be red in every chart on every page of a report. The legend markers will also be bound to these colors.

Syntax: How to Bind a Color to a Field Value

```
type=data, marker-color=color, when=field eq value, $
```

where:

color

Specifies a color for the chart and legend marker when the condition in the WHEN phrase is satisfied. Color names, RGB color values, and hex color values are supported.

field

Is the name of the BY field that is assigned to the *color* attribute category whose values will be bound to specific colors.

value

Is the value that is bound to the color specified by the *marker-color* attribute.

Reference: Usage Notes for Color Binding

- ❑ Color binding is supported when only one field is assigned to the *color* attribute category.
- ❑ The only operator supported in the WHEN condition is EQ.
- ❑ Color binding is only supported with chart attribute syntax.
- ❑ The COLUMN attribute, which is generally required when applying conditional styling to charts, is optional for color binding, making it easier to reuse color mappings in separate procedures.
- ❑ If there are BY values for which no color is explicitly specified, their colors are whatever series color is specified by the StyleSheet (or by chart engine defaults, if there is no StyleSheet).

Example: Binding Field Values to Colors in a Bar Chart

The following request binds the field values of the PRODUCT_CATEGORY field to specific colors.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=X-AXIS, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=COLOR, $
TYPE=DATA, MARKER-COLOR=red, WHEN=PRODUCT_CATEGORY EQ 'Accessories',$
TYPE=DATA, MARKER-COLOR=orange, WHEN=PRODUCT_CATEGORY EQ 'Camcorder',$
TYPE=DATA, MARKER-COLOR=yellow, WHEN=PRODUCT_CATEGORY EQ 'Computers',$
TYPE=DATA, MARKER-COLOR=green, WHEN=PRODUCT_CATEGORY EQ 'Media Player',$
TYPE=DATA, MARKER-COLOR=blue, WHEN=PRODUCT_CATEGORY EQ 'Stereo Systems',$
TYPE=DATA, MARKER-COLOR=purple, WHEN=PRODUCT_CATEGORY EQ 'Televisions',$
TYPE=DATA, MARKER-COLOR=gray, WHEN=PRODUCT_CATEGORY EQ 'Video Production',$
ENDSTYLE
END
```

The following version of the request binds the field values of the PRODUCT_CATEGORY field to the same colors, but filters out the product categories that start with the letter C.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
WHERE PRODUCT_CATEGORY NOT LIKE 'C%'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=X-AXIS, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=COLOR, $
TYPE=DATA, MARKER-COLOR=red, WHEN=PRODUCT_CATEGORY EQ 'Accessories',$
TYPE=DATA, MARKER-COLOR=orange, WHEN=PRODUCT_CATEGORY EQ 'Camcorder',$
TYPE=DATA, MARKER-COLOR=yellow, WHEN=PRODUCT_CATEGORY EQ 'Computers',$
TYPE=DATA, MARKER-COLOR=green, WHEN=PRODUCT_CATEGORY EQ 'Media Player',$
TYPE=DATA, MARKER-COLOR=blue, WHEN=PRODUCT_CATEGORY EQ 'Stereo Systems',$
TYPE=DATA, MARKER-COLOR=purple, WHEN=PRODUCT_CATEGORY EQ 'Televisions',$
TYPE=DATA, MARKER-COLOR=gray, WHEN=PRODUCT_CATEGORY EQ 'Video Production',$
ENDSTYLE
END
```

Both charts are shown in the following image. Note that, although Media Player is represented by the fourth riser on chart 1 and the second riser on chart 2, it is green on both charts.

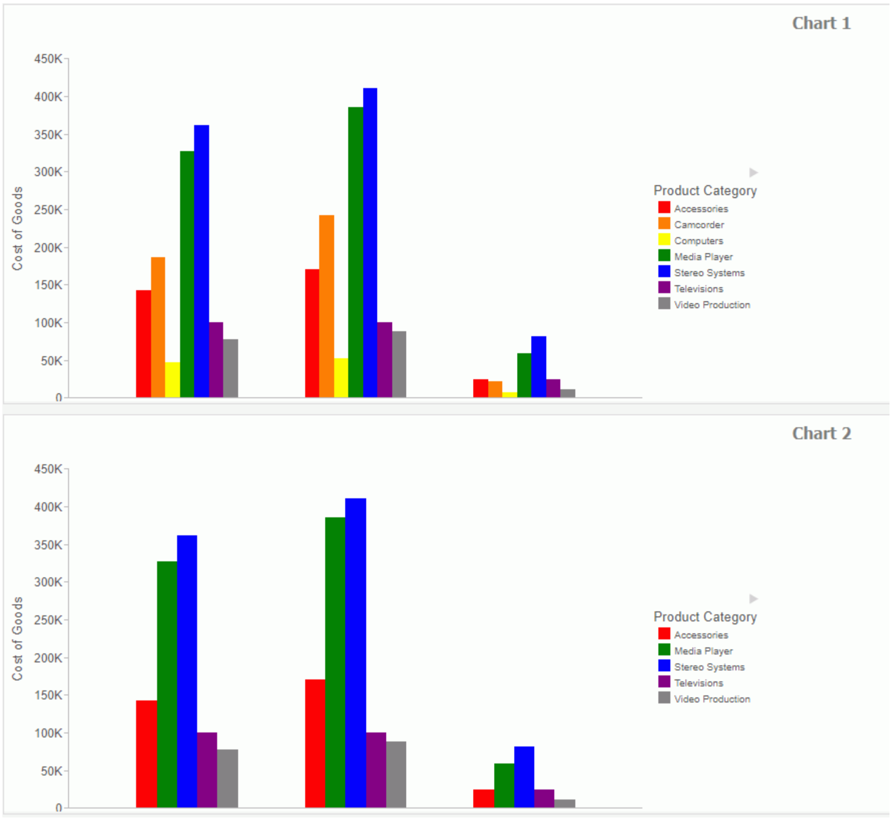


Chart Types for Chart Attribute Syntax

Without chart attribute syntax, the chart types for vertical charts are different from those for horizontal versions of those charts. For example, VBAR is used for vertical bar charts, while HBAR is used for horizontal bar charts.

With chart attribute syntax, the chart type BAR is used for any type of bar chart, and the ORIENTATION attribute in the TYPE=REPORT declaration of the WebFOCUS StyleSheet differentiates between horizontal and vertical charts. For more information, see *Specifying Chart Orientation* on page 205.

Similarly, stacked bar, line, and area charts are distinguished from side-by-side charts using the CHART-SERIES-LAYOUT attribute in the TYPE=REPORT declaration of the WebFOCUS StyleSheet. For more information, see *Specifying Chart Layout for Bar, Line, and Area Charts* on page 207.

The available chart types (LOOKGRAPH values) are:

- **AREA.** Area charts are similar to line charts in that they demonstrate the movement of numeric data. In an area chart, however, the area between the data line and the zero line (or axis) is usually colored or textured.
- **BAR.** A bar chart plots numeric data by displaying rectangular blocks against a scale. The length of a bar corresponds to a value or amount. You can develop a clear mental image of comparisons among data series by distinguishing the relative heights of the bars.
- **BOXPLOT.** A boxplot shows data broken into quartiles. In a boxplot, each series and group requires five values. For a given series and group box, the first value is the minimum (lower hat), the second defines the box bottom (first quartile), the third value is the median (second quartile), the fourth value defines the box top (third quartile), and the fifth value defines the top hat (maximum value).
- **BUBBLE.** A bubble chart is an enhanced scatter plot in which the size of each marker is proportional to the value of a third measure.
- **BUBBLEMAP.** A bubblemap is a proportional symbol map chart, in which the markers are positioned on a geographic point or area and are sized depending on the value of a measure. They can be colored by a measure or dimension field.
- **CHOROPLETH.** A choropleth is a map chart in which polygonal areas that map to geographic locations are colored depending on a measure or dimension.
- **DATAGRID.** A data grid is a tabular representation of data with rows, columns, and cells.
- **FUNNEL.** A funnel chart is basically a pie chart that shows only one group of data at a time. The series in the group are stacked in the funnel, with the first series at the top and the last series at the bottom.
- **GAUGE.** Gauge charts identify a single value along an axis that is usually displayed in a circle. They are often used in dashboards to display progress or quantity.
- **HEATMAP.** Heatmaps contain a row or column matrix of rectangles that are displayed in different colors depending on the data values.

- **LINE.** Line charts are useful for emphasizing the movement or trend of numeric data over time, since they allow a viewer to trace the evolution of a particular point by working backwards or interpolating. Highs and lows, rapid or slow movement, or a tendency toward stability are all types of trends that are well suited to a line chart.
- **MARKER.** A matrix marker chart is a graphical representation of tabular data. It displays a marker in each cell of the matrix. This marker can be sized by one measure and colored by a second measure.
- **MEKKO.** A mekko (also called marimekko) chart is a percent bar chart, except that the width of each bar riser is based on the overall value of the stack. The widths are automatically sorted high-to-low.
- **PIE.** A pie chart emphasizes where your data fits in relation to a larger whole. Each slice represents a percentage of the whole. If the hole size is zero, a pie chart is generated. If the hole size is greater than zero, a pie ring chart is generated.
- **SCATTER.** Scatter charts typically show the relationship between numeric measures. Use a scatter plot to visualize the density of individual data values around particular points or to demonstrate patterns in your data. With chart attribute syntax, dimensions as well as measures are supported.
- **STREAM.** A streamgraph is a simplified version of a stacked area chart. In a streamgraph, the x-axis is often turned off, and there are no gridlines or frames. A streamgraph does not use data text labels.
- **TAGCLOUD.** A tagcloud chart is a visual representation of group labels. The size of each label is proportional to its data value. Tagclouds are used by social media to measure the frequency of words in order to quantify sentiments.
- **TREEMAP.** A treemap chart displays hierarchical data as a set of nested rectangles.

Sample Charts Using Chart Attribute Syntax

This section gives a brief description and example of each chart type that supports chart attribute syntax.

In addition to the attributes listed for each chart type, all charts can be generated in a matrix, have fields added to the tooltip, and can be generated as separate pages. In addition, certain chart types can be generated with a slider control.

Area Charts Using Chart Attribute Syntax

Area charts are similar to line charts in that they demonstrate the movement of numeric data, except that the area between the data line and the zero line (or axis) is usually colored or textured. Area charts allow you to stack data on top of each other. Stacking allows you to highlight the relationship between data series, showing how some data series approach or shadow a second series.

The chart type is AREA. AREA attribute categories are:

- **x-axis** for the sort field, typically a time-based field such as YEAR.
- **y-axis** for the measures.
- **color** for the colors of the areas and legend (optional).

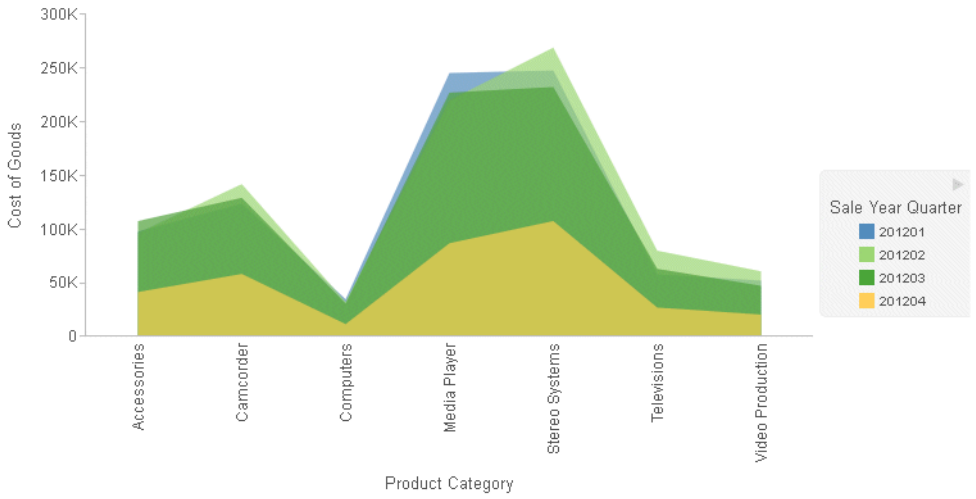
  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.

Example: Sample Area Chart Using Chart Attribute Syntax

The following request generates an area chart. The sort field (TIME_YEARQTR) is assigned to the color category, the sort field PRODUCT_CATEGORY is assigned to the x-axis category, and the measure field is assigned to the y-axis. The chart type is AREA:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY WF_RETAIL_TIME_SALES.TIME_YEARQTR
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH AREA
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=WF_RETAIL_TIME_SALES.TIME_YEARQTR, BUCKET=color, $
ENDSTYLE
END
```

The output is shown in the following image:



Bar Charts Using Chart Attribute Syntax

A bar chart plots numeric data by displaying rectangular blocks against a scale. The length of a bar corresponds to a value or amount. You can develop a clear mental image of comparisons among data series by distinguishing the relative heights of the bars. Use a bar chart to display numeric data when you want to present comparisons of data.

The chart type is BAR. BAR attribute categories are:

- **x-axis** for the sort field.
- **y-axis** for the measures.
- **color** for the colors of the bars and legend (optional).

  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.
- **size** for the width of the bars (optional).

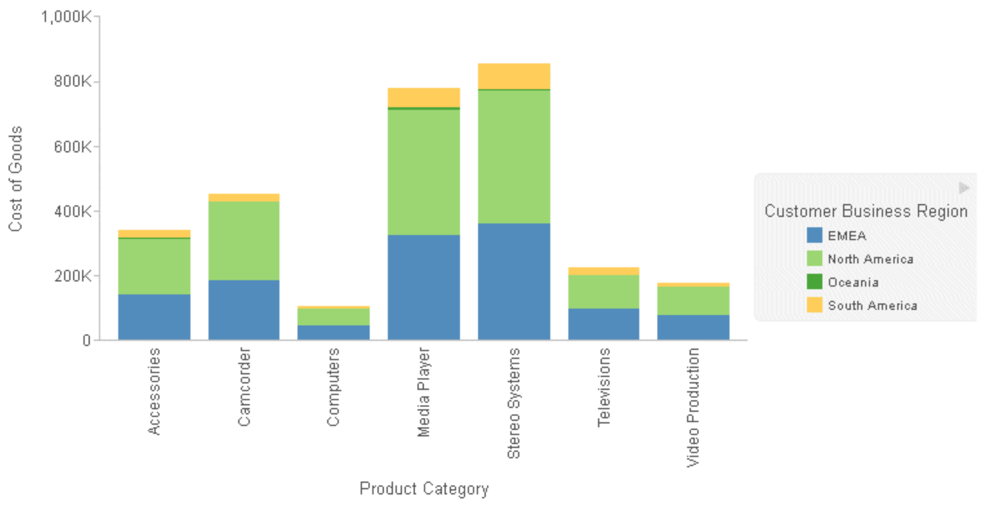
Example: Sample Stacked Bar Chart Using Chart Attribute Syntax

The following request generates a stacked bar chart. The sort field (PRODUCT_CATEGORY) is assigned to the x-axis, the sort field BUSINESS_REGION is assigned to the color category, and the measure COGS_US is assigned to the y-axis. (For information about creating stacked bar charts, see *Specifying Chart Layout for Bar, Line, and Area Charts* on page 207.)

The chart type is BAR:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=stacked,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Boxplot Charts Using Chart Attribute Syntax

A boxplot shows data broken down into quartiles. In a boxplot, each series and group requires five values. For a given series and group box, the first value is the minimum (lower hat), the second defines the box bottom (first quartile), the third value is the median (second quartile), the fourth value defines the box top (third quartile), and the fifth value defines the location of the top hat (maximum value). Box plot fill color and border are defined by the series color and border.

The chart type is BOXPLOT. BOXPLOT attribute categories are:

- **x-axis** for the sort field.
- **min** for the measure that represents the minimum value.
- **lower** for the measure that represents the box bottom.
- **median** for the measure that represents the median.
- **upper** for the measure that represents the box top.
- **max** for the measure that represents the maximum value.
- **color** for a sort field used to color the boxes and legend (optional).

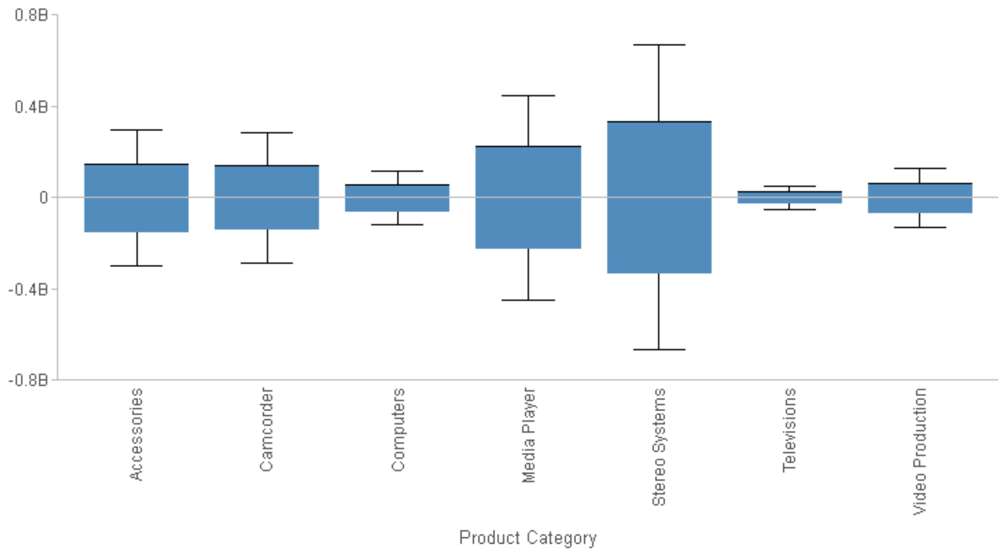
  If the field is a measure the colors will be visualized as a gradient. If the field is a dimension, the color will be visualized as discrete colors.

Example: Sample Boxplot Using Chart Attribute Syntax

The following request generates a boxplot.

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US -100000;
DIFF2 = COGS_US -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=product_category, bucket=x-axis,$
type=data, column=diff2, bucket=min,$
type=data, column=diff1, bucket=lower,$
type=data, column=c4, bucket=median,$
type=data, column=diff3, bucket=upper,$
type=data, column=diff4, bucket=max,$
*GRAPH_JS
"boxPlotProperties": {"drawHatAsBox": false}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Bubble Charts Using Chart Attribute Syntax

A bubble chart is an enhanced scatter plot in which the size of each marker is proportional to the value of a measure. Therefore, a bubble chart requires three values, (x-position, y-position, and size) to draw each bubble marker.

The chart type is BUBBLE. BUBBLE attribute categories are:

- **x-axis** for the sort field. If the sort field is a measure, it determines the bubble intercept on the x-axis
- **y-axis** for the measure that represents the bubble intercept on the y-axis.
- **size** for the measure representing bubble size.
- **color** for a field used to color the bubbles and legend (optional).

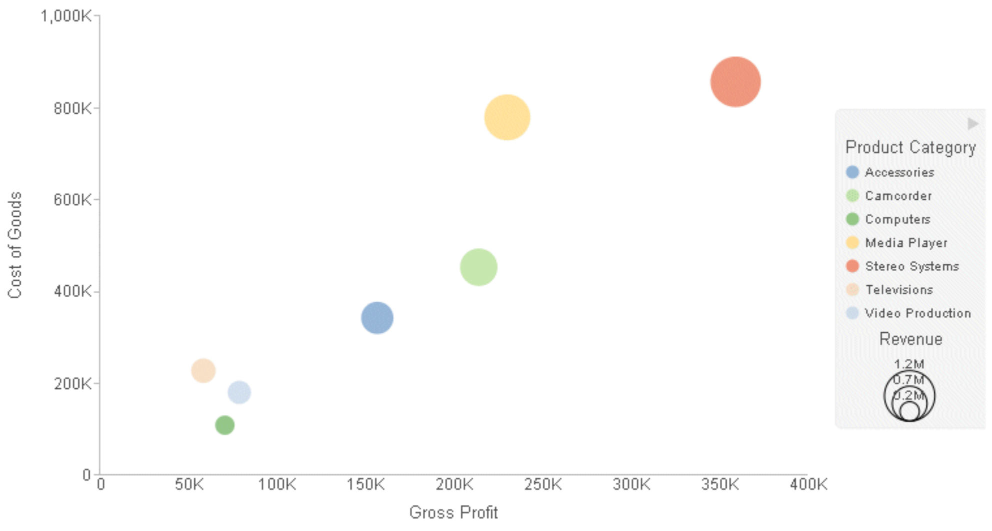
  If the field is a measure the colors will be visualized as a gradient. If the field is a dimension, the color will be visualized as discrete colors.
- **detail** for low-level sort fields used to add data points to the chart (optional).

Example: Sample Bubble Chart Using Chart Attribute Syntax

The following request generates a bubble chart. The sort field (PRODUCT_CATEGORY) is assigned to the color attribute category and generates the legend, the first measure (COGS_US) is assigned to the y-axis, the second measure (GROSS_PROFIT_US) is assigned to the x-axis, and the third measure (REVENUE_US) is assigned to the size category and generates a size indicator in the legend area. The chart type is BUBBLE:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=size, $
ENDSTYLE
END
```

The output is shown in the following image:



Bubblemap Charts Using Chart Attribute Syntax

A bubblemap is a proportional symbol map chart, in which the markers are positioned on a geographic point or area and are sized depending on the value of a measure.

The chart type is BUBBLEMAP. BUBBLEMAP attribute categories are:

- **location** for the sort field containing the geolocation at which the symbols on the map should appear. Either location or latitude and longitude can be used.
- **latitude** for the sort field or measure containing the latitude for the symbols. Either location or latitude and longitude can be used.
- **longitude** for the sort field or measure containing the longitude for the symbols. Either location or latitude and longitude can be used.
- **size** for the measure that represents the size of the symbols.
- **color** for the field used to visualize symbol and legend color (optional).

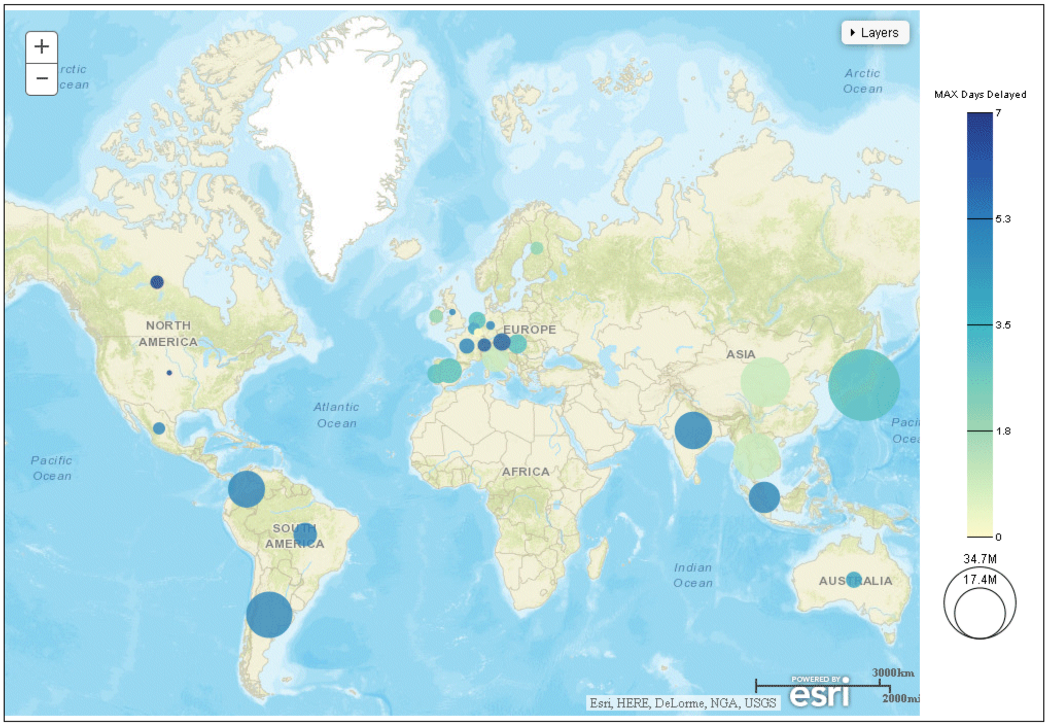
  If the field is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete.

Example: Sample Bubblemap Using a Location With Chart Attribute Syntax

The following request generates a proportional symbol chart using an Esri map as the base layer. The markers are located using the COUNTRY_NAME sort field, are colored by the MIN.CITY_POPULATION measure, and are sized by the MAX.DAYSDELAYED measure. The JSON property *bubbleMarker:maxSize* scales the markers to 10%. Other properties in the StyleSheet specify the map type, map layers, and other map properties. For more information, see *Map Support* on page 817:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION MAX.DAYSDELAYED
BY COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
TYPE=DATA, COLUMN=MAX.DAYSDELAYED, BUCKET=color, $
*GRAPH_JS
"legend": {
    "visible": true
},
 "bubbleMarker": {"maxSize": "10%" },

"extensions": {
    "com.esri.map": {
        "overlayLayers":             [
                {
                    "title": "Population",
                    "layerType": "bubble",
                    "geometrySourceType": "esri",
                    "geometryLocateField": ["Country"],
                    "geometryDataField": "name",
                    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/
arcgis/rest/services/World_Countries_(Generalized)/FeatureServer/0"
                }
            ],
        "baseLayer": {
            "basemap": "streets"
        }
    }
}
*END
ENDSTYLE
END
```

The output is shown in the following image:

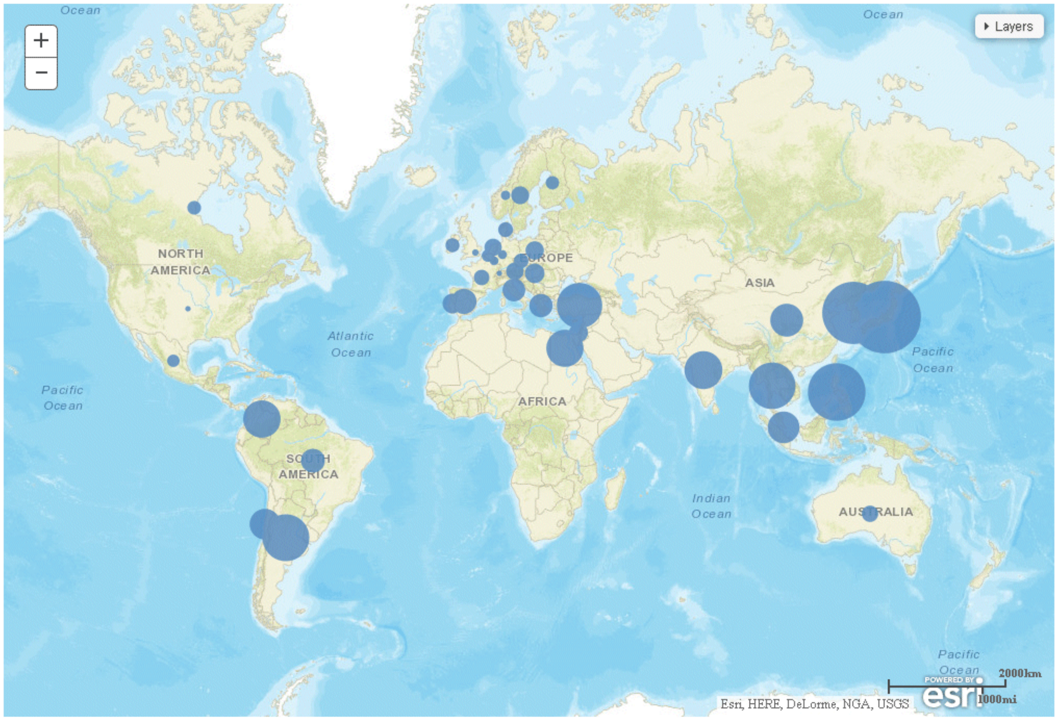


Example: Sample Bubblemap Using Latitude and Longitude With Chart Attribute Syntax

The following request generates a proportional symbol chart using an Esri map as the base layer. The markers are located using the COUNTRY_LATITUDE and COUNTRY_LONGITUDE sort fields and are sized by the MIN.CITY_POPULATION measure. The JSON property *bubbleMarker:maxSize* scales the markers to 10%. Other properties in the StyleSheet specify the map type, map layers, and other map properties. For more information, see *Map Support* on page 817:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY COUNTRY_LATITUDE
BY COUNTRY_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=COUNTRY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
  "bubbleMarker":{"maxSize":"10%"},

          "legend": {
              "visible": false
          },
          "extensions": {
              "com.esri.map": {
                  "overlayLayers":             [
                          {
                              "title": "Population",
                              "layerType": "bubble",
                              "geometryXY": {
                                  "y": "lat",
                                  "x": "lng"
                              },
                              "geometrySourceType": "seriesdata"
                          }
                      ],
                  "baseLayer": {
                      "basemap": "streets"
                  }
              }
          }

          *END
          ENDSTYLE
          END
```

The output is shown in the following image:



Choropleth Charts Using Chart Attribute Syntax

A choropleth is a map chart in which polygonal areas that map to geographic locations are colored depending on a measure or dimension.

The chart type is CHOROPLETH. CHOROPLETH attribute categories are:

- **location** for the sort field containing the geolocation at which the symbols on the map should appear.
- **color** for the polygon color.

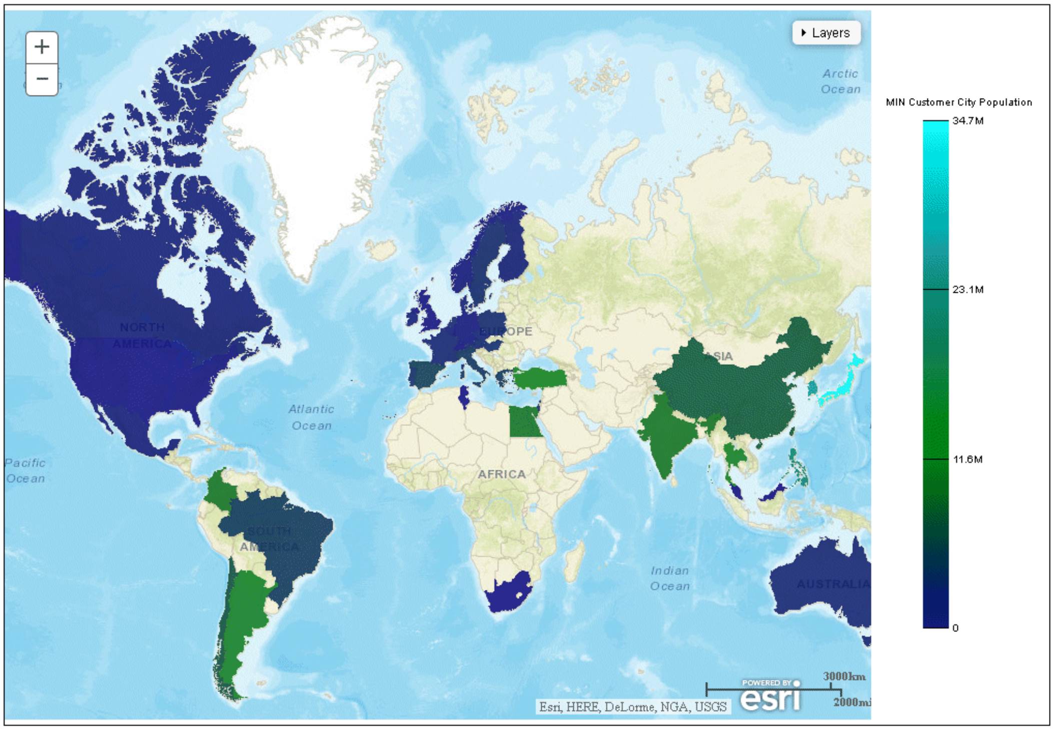
  If the field is a measure, the color will be visualized as a gradient. If it is a dimension, the colors will be discrete.

Example: Sample Choropleth Using Chart Attribute Syntax

The following request generates a choropleth chart using an Esri map as the base layer. The colored polygons represent the COUNTRY_NAME sort field and are colored by the MIN.CITY_POPULATION measure. The JSON property *colorScale* establishes the color scale for the polygons. Other properties in the StyleSheet specify the map type, map layers, and other map properties. For more information, see *Map Support* on page 817:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=color, $
*GRAPH_JS
legend: {
    "visible": true
},
 "colorScale": {"colors":["navy", "green", "teal", "cyan"]},

"extensions": {
    "com.esri.map": {
        "overlayLayers":             [
                {
                    "title": "WF_RETAIL_LITE Countries",
                    "layerType": "choropleth",
                    "geometrySourceType": "esri",
                    "geometryLocateField": ["Country"],
                    "geometryDataField": "name",
                    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/
arcgis/rest/services/World_Countries_(Generalized)/FeatureServer/0"
                }
            ],
        "baseLayer": {
            "basemap": "streets"
        }
    }
}

*END
ENDSTYLE
END
```

The output is shown in the following image:



Data Grids Using Chart Attribute Syntax

Data grids display data in a table with rows and columns. Unlike the tabular reports generated with the WebFOCUS TABLE FILE command, data grids are generated in a GRAPH FILE request using PCHOLD FORMAT JSCHART.

Using the data grid properties, you can format the grid and display column totals for measures in the grid.

The chart type is DATAGRID. The DATAGRID attribute categories are:

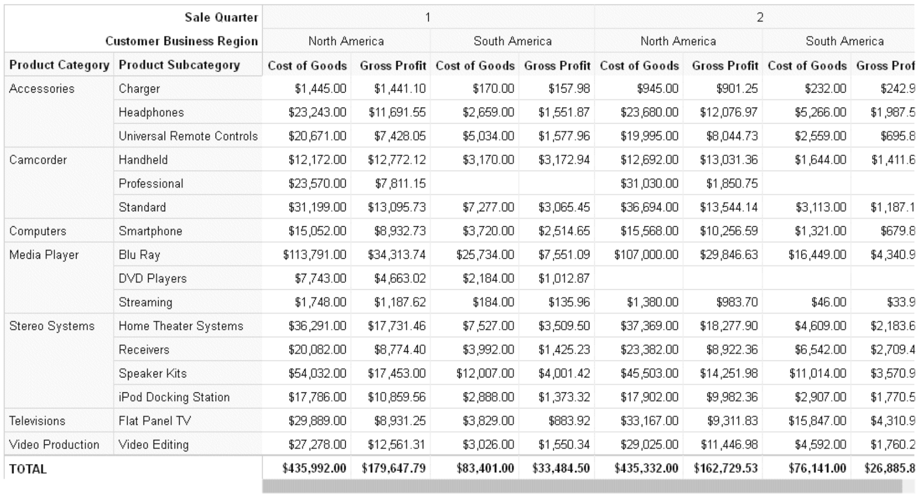
- **measure** for the measure fields whose values are displayed in the cells of the grid.
- **row** for the measure or dimension fields representing rows of the grid.
- **column** for the measure or dimension fields representing columns of the grid.

Example: Generating a Data Grid

The following request generates a data grid with column totals. The rows represent product categories and product sub-categories. The columns represent the first two quarters of the year and the North America and South America business regions. The measures are cost of goods and gross profit. By default, scroll bars are added when the entire grid does not fit in the draw area.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY TIME_QTR
BY BUSINESS_REGION
WHERE (BUSINESS_REGION NE 'Oceania' OR 'EMEA') AND (TIME_QTR EQ 1 OR 2)
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=time_qtr,          bucket=column, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,           bucket=measure, $
type=data, column=gross_profit_us,   bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"columnTotals": {"visible": true}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



	Sale Quarter	1				2			
	Customer Business Region	North America		South America		North America		South America	
Product Category	Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Prof
Accessories	Charger	$1,445.00	$1,441.10	$170.00	$157.98	$945.00	$901.25	$232.00	$242.9
	Headphones	$23,243.00	$11,691.55	$2,659.00	$1,551.87	$23,680.00	$12,076.97	$5,266.00	$1,987.5
	Universal Remote Controls	$20,671.00	$7,428.05	$5,034.00	$1,577.96	$19,995.00	$8,044.73	$2,559.00	$695.8
Camcorder	Handheld	$12,172.00	$12,772.12	$3,170.00	$3,172.94	$12,692.00	$13,031.36	$1,644.00	$1,411.6
	Professional	$23,570.00	$7,811.15			$31,030.00	$1,850.75		
	Standard	$31,199.00	$13,095.73	$7,277.00	$3,065.45	$36,694.00	$13,544.14	$3,113.00	$1,187.1
Computers	Smartphone	$15,052.00	$8,932.73	$3,720.00	$2,514.65	$15,568.00	$10,256.59	$1,321.00	$679.8
Media Player	Blu Ray	$113,791.00	$34,313.74	$25,734.00	$7,551.09	$107,000.00	$29,846.63	$16,449.00	$4,340.9
	DVD Players	$7,743.00	$4,663.02	$2,184.00	$1,012.87				
	Streaming	$1,748.00	$1,187.62	$184.00	$135.96	$1,380.00	$983.70	$46.00	$33.9
Stereo Systems	Home Theater Systems	$36,291.00	$17,731.46	$7,527.00	$3,509.50	$37,369.00	$18,277.90	$4,609.00	$2,183.6
	Receivers	$20,082.00	$8,774.40	$3,992.00	$1,425.23	$23,382.00	$8,922.36	$6,542.00	$2,709.4
	Speaker Kits	$54,032.00	$17,453.00	$12,007.00	$4,001.42	$45,503.00	$14,251.98	$11,014.00	$3,570.9
	iPod Docking Station	$17,786.00	$10,859.56	$2,888.00	$1,373.32	$17,902.00	$9,982.36	$2,907.00	$1,770.5
Televisions	Flat Panel TV	$29,889.00	$8,931.25	$3,829.00	$883.92	$33,167.00	$9,311.83	$15,847.00	$4,310.9
Video Production	Video Editing	$27,278.00	$12,561.31	$3,026.00	$1,550.34	$29,025.00	$11,446.98	$4,592.00	$1,760.2
TOTAL		**$435,992.00**	**$179,647.79**	**$83,401.00**	**$33,484.50**	**$435,332.00**	**$162,729.53**	**$76,141.00**	**$26,885.8**

Funnel Charts Using Chart Attribute Syntax

A funnel chart is basically a pie chart that shows only one group of data at a time. The series in the group are stacked in the funnel, with the first series at the top and the last series at the bottom. In the funnel chart, the display field functions as the group, and the sort fields function as the series.

The chart type is FUNNEL. FUNNEL attribute categories are:

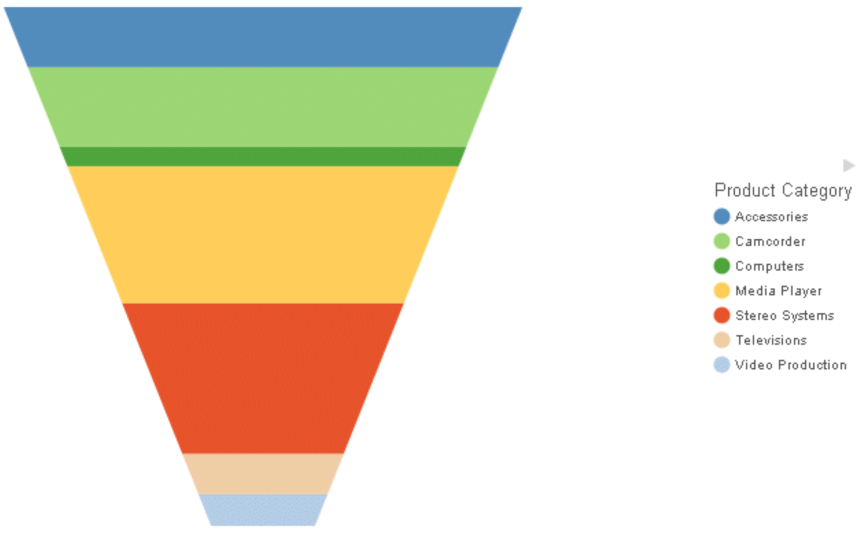
- **color** for the sort field.
- **measure** for the measure used to generate the funnel slices.

Example: Sample Funnel Chart Using Chart Attribute Syntax

The following request generates a funnel chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure, $
ENDSTYLE
END
```

The output is shown in the following image.



Gauge Charts Using Chart Attribute Syntax

Gauge charts identify a single value along an axis that is usually displayed in a circle. They are often used in dashboards to display progress or quantity.

The chart type is GAUGE. The GAUGE attribute category is:

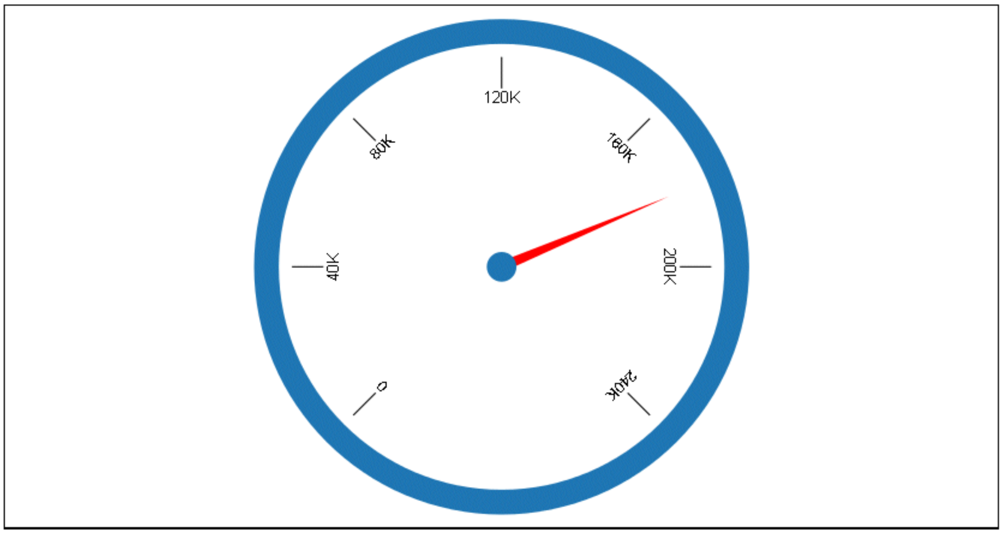
- **measure** for the field containing the value represented on the gauge.

Example: Sample Gauge Chart Using Chart Attribute Syntax

The following request generates a gauge chart. The chart type is GAUGE:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE
ON GRAPH SET STYLE *
type=data, column=revenue_us, bucket=measure,$
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "totalLabel": {"visible": false}
    }
*END
ENDSTYLE
END
```

The output is shown in the following image:



Heatmap Charts Using Chart Attribute Syntax

Heatmap charts contain a row or column matrix of rectangles that are displayed in different colors depending on the data values. They use the same kind of data as bar, line, and area charts.

The chart type is HEATMAP. HEATMAP attribute categories are:

- **y-axis** for the first sort field.
- **x-axis** for the second sort field.
- **color** for the measure.

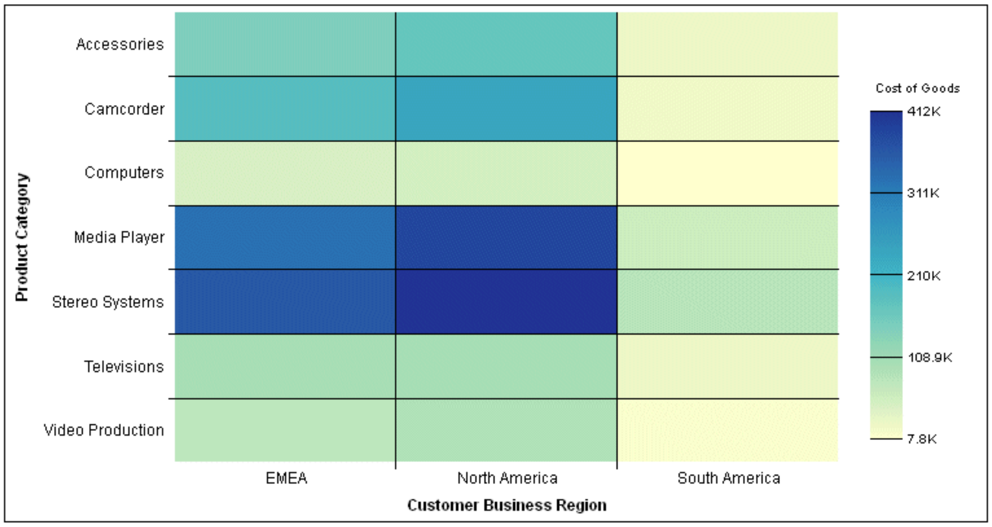
  Rectangle color will depend on the value of this measure. The color will be visualized as a gradient.

Example: Sample Heatmap Using Chart Attribute Syntax

The following request generates a heatmap where the rectangles are colored by the measure COGS_US (color category), the rows represent the first BY field (y-axis category), and the columns represent the second BY field (x-axis category). The chart type is HEATMAP:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH SET LOOKGRAPH HEATMAP
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=data, column=product_category, bucket=y-axis, $
type=data, column=business_region, bucket=x-axis, $
type=data, column=cogs_us, bucket=color, $
END
```

The output is shown in the following image:



Line Charts Using Chart Attribute Syntax

Line charts are useful for emphasizing the movement or trend of numeric data over time, since they allow a viewer to trace the evolution of a particular point by working backwards or interpolating. Highs and lows, rapid or slow movement, or a tendency toward stability are all types of trends that are well suited to a line chart.

Line charts can also be plotted with two or more scales to suggest a comparison of the same value, or set of values, in different time periods. The number of scales your chart has depends on the type of chart you select.

The chart type is LINE. LINE attribute categories are:

- **x-axis** for the sort field, typically a time-based dimension, such as YEAR.
- **y-axis** for the measures.
- **color** for the colors of the lines, markers, and legend (optional).

  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.
- **size** for the measure representing the width of the markers or lines (optional).

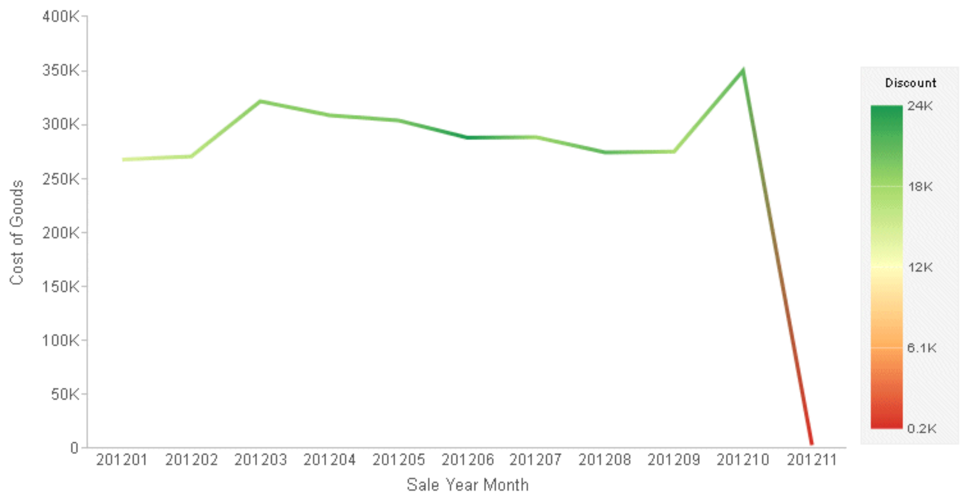
  If markers are visible, this measure is used to vary the size of the markers. If markers are not visible, this measure is used to vary the width of the lines.

Example: Sample Line Chart

The following request creates a line chart. The sort field TIME_YEARMTH is assigned to the x-axis category. The measure COGS_US is assigned to the y-axis category. The measure DISCOUNT_US is assigned to the color category and is used to color the line, markers, and legend. Since the color field is a measure, the colors are visualized as a gradient. The chart type is LINE:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DISCOUNT_US
BY WF_RETAIL_TIME_SALES.TIME_YEARMTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=WF_RETAIL_TIME_SALES.TIME_YEARMTH, BUCKET=x-axis,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis,$
TYPE=DATA, COLUMN=DISCOUNT_US, BUCKET=color,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

The output is shown in the following image:



Matrix Marker Charts Using Chart Attribute Syntax

A matrix marker chart is a matrix chart that displays a marker in each cell of the matrix.

The chart type is MARKER. MARKER attribute categories are:

- **row** for the fields representing rows of the matrix.

- **column** for the fields representing columns of the matrix.
- **size** for the measure representing the size of the marker.
- **color** for the measure representing the colors of the markers.

  The color will be visualized as a gradient.

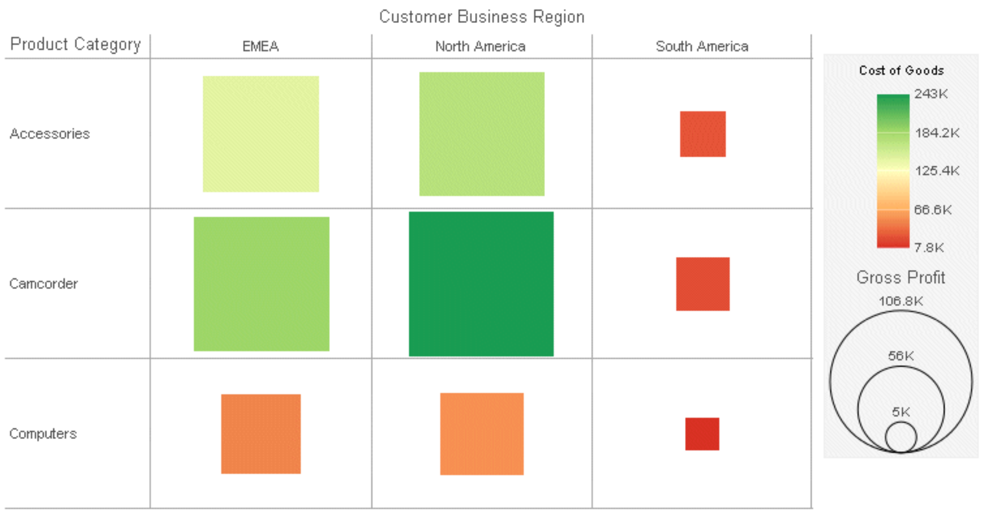
Example: Sample Matrix Marker Chart Using Chart Attribute Syntax

The following request generates a matrix marker chart. The rows represent product categories. The columns represent business regions. The marker in each cell is sized by the GROSS_PROFIT_US measure and colored by the COGS_US measure. The marker and legend colors are visualized as a gradient.

The JSON properties change the marker shape to a square. The chart type is MARKER:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
WHERE PRODUCT_CATEGORY LE 'D'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH MARKER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=size, $
*GRAPH_JS
"series": [{"series": 0, "marker": {"shape": "square"}}]
*END
ENDSTYLE
END
```

The output is shown in the following image:



Mekko Charts Using Chart Attribute Syntax

A mekko (also called marimekko) chart is a percent bar chart, except that the width of each bar riser is based on the overall value of the stack. The widths are automatically sorted in high-to-low order. The default tooltips show the value and percentage of the measure within the group.

The chart type is MEKKO. Mekko attribute categories are:

- **x-axis** for the sort field.
- **y-axis** for the measures.
- **color** for the colors of the bars and legend (optional).

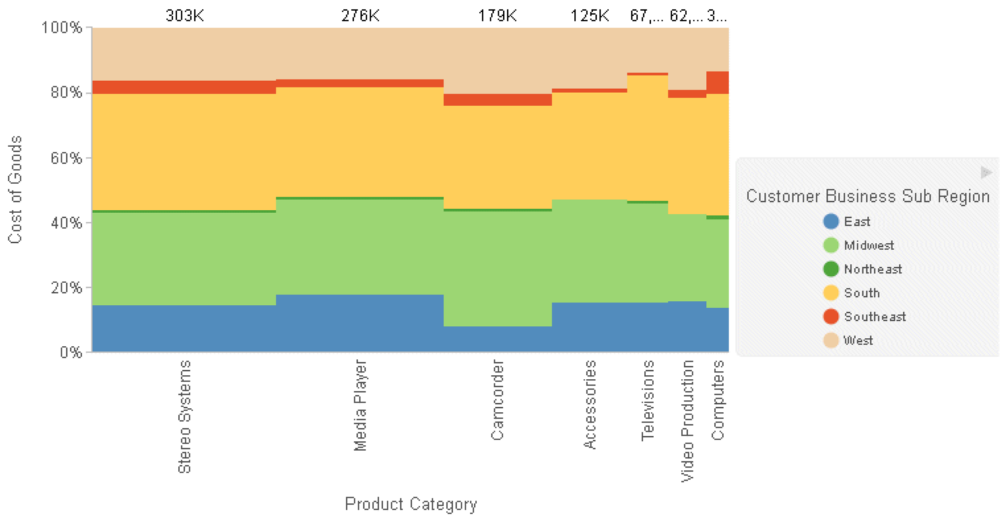
  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.

Example: Sample Mekko Chart Using Chart Attribute Syntax

The following request generates a mekko chart with one measure, one sort field for the x-axis, and one sort field for the color category.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_SUB_REGION
BY PRODUCT_CATEGORY
WHERE COUNTRY_NAME EQ 'United States'
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH MEKKO
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=BUSINESS_SUB_REGION, bucket=color,$
type=data, column=product_category, bucket=x-axis,$
ENDSTYLE
END
```

The output is shown in the following image.



Pie Charts Using Chart Attribute Syntax

A pie chart emphasizes where your data fits in relation to a larger whole. Each slice represents a percentage of the whole. Keep in mind that pie charts work best when your data falls into a limited number of groups. Too many groups divide the pie into small segments that are difficult to see. Use color or texture on individual segments to create visual contrast.

If the hole size is zero, a pie chart is generated. If the hole size is greater than zero, a pie ring chart is generated.

The chart type is PIE. PIE attribute categories are:

- **color** for the sort field.
- **measure** for the measure used to generate the pie slices.
- **size** for the measure used to size the pie charts if they are generated in a matrix (optional).

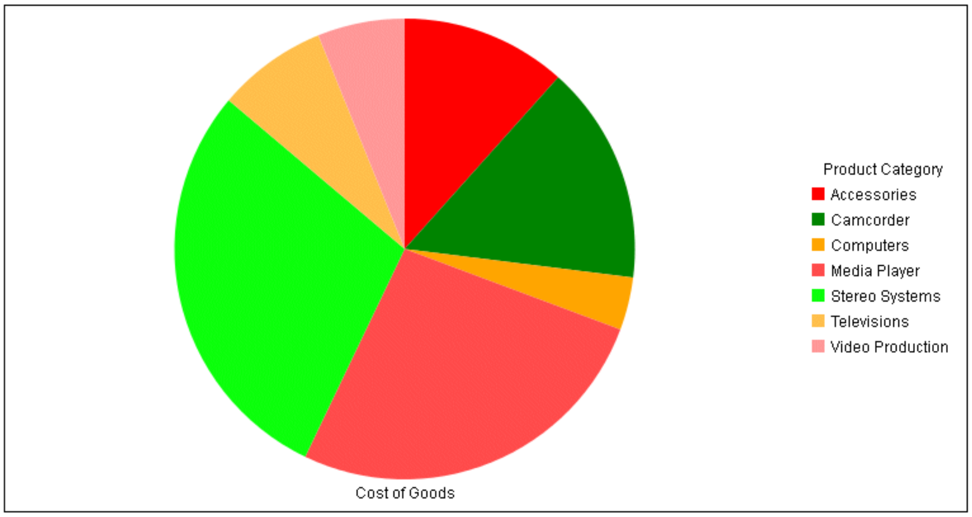
Note: A pie chart that is built by adding a measure to the measure category and a BY field to the color category results in a single pie chart where the measure name is placed at the bottom of the chart.

Example: Sample Pie Chart Using Chart Attribute Syntax

The basic components of a pie chart are a measure and a sort field. The sort field generates the colors for the slices and the legend. The chart type is PIE:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure, $
ENDSTYLE
END
```

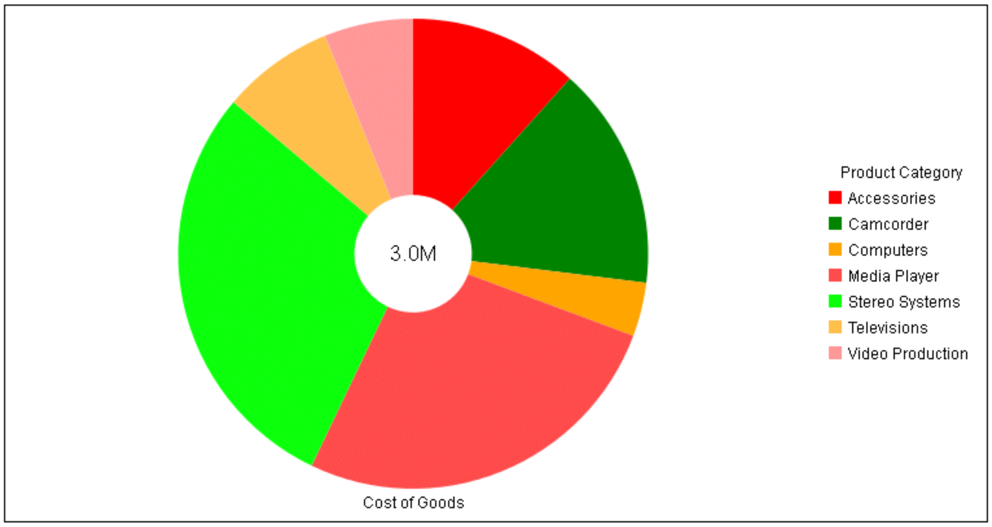
The output is shown in the following image:



To generate a pie chart with a hole, add the JSON holeSize property:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure, $
*GRAPH_JS
"pieProperties": {"holeSize": "25%"}
*END
ENDSTYLE
END
```

The output is shown in the following image. By default, the total label displays in the hole:



Scatter Charts Using Chart Attribute Syntax

Scatter charts typically show the relationship between two numeric measures. Chart attribute syntax supports dimensions as well as measures. Use a scatter plot to visualize the density of individual data values around particular points or to demonstrate patterns in your data.

The chart type is SCATTER. SCATTER attribute categories are:

- **x-axis** for the sort field (either a measure or dimension).
- **y-axis** for the measure or dimension that defines the y-axis intercepts for the markers.
- **size** for the measure that represents the size of the markers (optional).
- **color** for the field that represents the colors of the markers (optional).

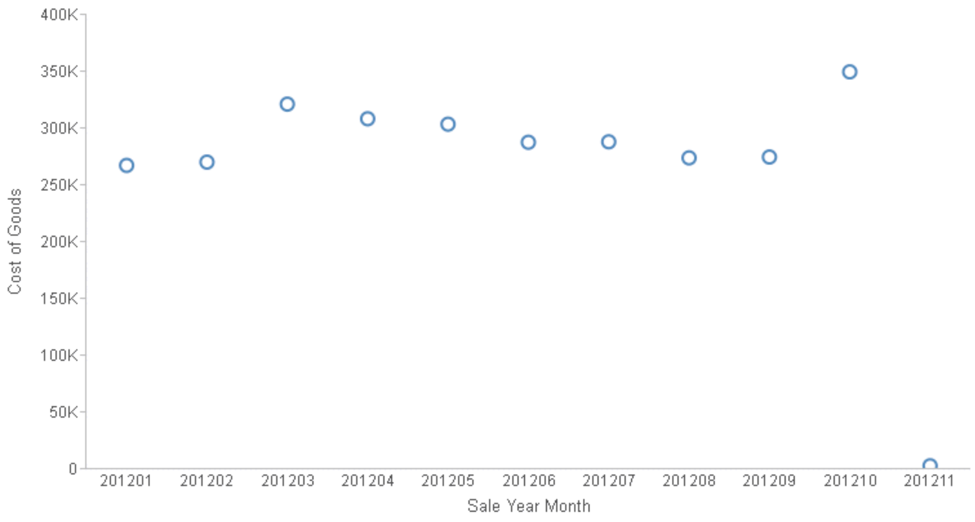
  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors.
- **detail** for low-level sort fields that can be used to generate additional data points on the chart (optional).

Example: Sample Scatter Chart Using Chart Attribute Syntax

The following request creates a scatter chart. The sort field TIME_YEARMTH is assigned to the x-axis category. The measure COGS_US is assigned to the y-axis category. The chart type is SCATTER:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY WF_RETAIL_TIME_SALES.TIME_YEARMTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=WF_RETAIL_TIME_SALES.TIME_YEARMTH, BUCKET=x-axis,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

The output is shown in the following image:



Streamgraph Charts Using Chart Attribute Syntax

A streamgraph is a simplified version of a stacked area chart. In a streamgraph, often the x-axis is turned off, and there are no gridlines or frames. A streamgraph does not use data text labels. The data required to draw a streamgraph is the same format required to draw an area chart. However, streamgraphs are normally given many series (10 or more), each with many data point points (100 or more). A typical streamgraph would include 20 series with 400 data points in each series.

The chart type is STREAM. STREAM attribute categories are:

- **x-axis** for the sort field.
- **y-axis** for the measures.
- **color** for the colors of the areas and legend (optional).

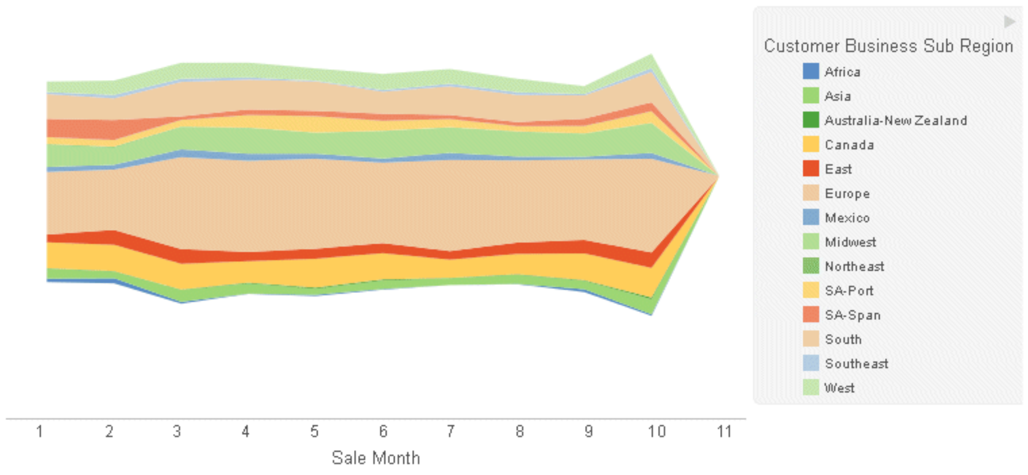
  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.

Example: Sample Streamgraph Colored by a Dimension

The following request generates a streamgraph where the areas are colored by the BUSINESS_SUB_REGION dimension.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_SUB_REGION
BY TIME_MTH
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH STREAM
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=time_mth, bucket=x-axis,$
type=data, column=business_sub_region, bucket=color,$
ENDSTYLE
END
```

The output is shown in the following image.



Tagcloud Charts Using Chart Attribute Syntax

A tagcloud chart is a visual representation of group labels. The size of each label is proportional to its data value. Tagclouds are used by social media to measure the frequency of words in order to quantify sentiments.

The chart type is TAGCLOUD. TAGCLOUD attribute categories are:

- **detail** for the sort field that contains the tags to be displayed.
- **size** for the measure that controls the size of the tags.
- **color** for the measure that controls the color of the tags. Tagclouds do not support multiple series and legends, so this category cannot be assigned to a sort field.

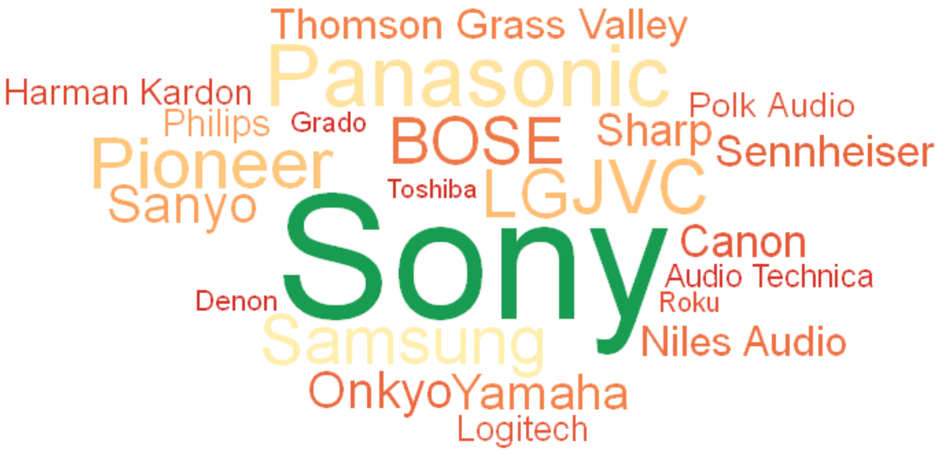
Tagclouds are not supported in a matrix.

Example: Sample Tagcloud Using Chart Attribute Syntax

The following request generates a tagcloud in which the tags are the brand names, the color of the tags is controlled by the quantity sold, and the size of the tags is controlled by the gross profit.

```
GRAPH FILE WF_RETAIL_LITE
SUM QUANTITY_SOLD  GROSS_PROFIT_US
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=brand, bucket=detail,$
type=data, column=quantity_sold, bucket=color,$
type=data, column=gross_profit_us, bucket=size,$
*GRAPH_JS
"tagcloudProperties": {"engine": "new"}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Treemap Charts Using Chart Attribute Syntax

A treemap chart displays hierarchical data as a set of nested rectangles.

An object can be nested arbitrarily deep.

The chart type is TREEMAP. TREEMAP attribute categories are:

- ❏ **detail** for the sort fields that define the hierarchy.

- **size** for a measure used to determine rectangle size (optional).
- **color** for the field that represents area color (optional).

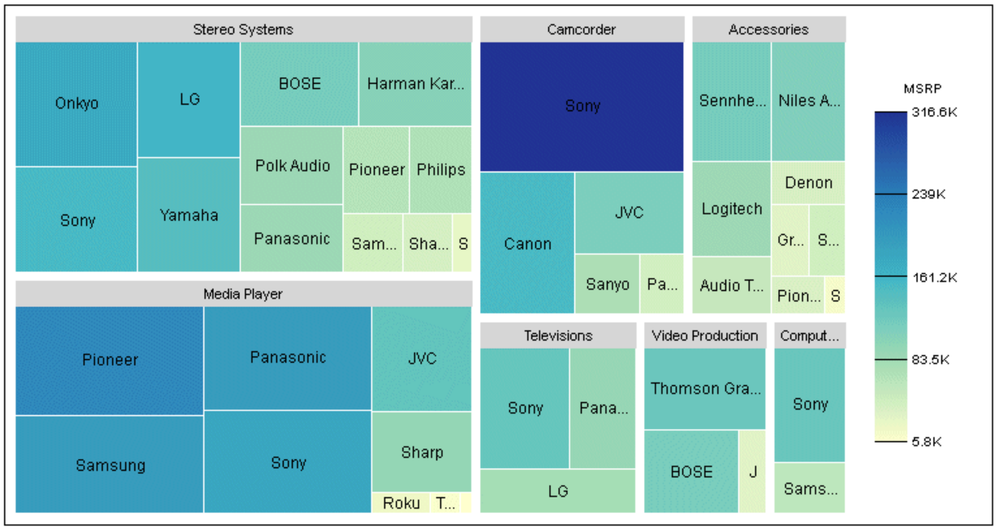
  If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors.

Example: Sample Treemap Colored by a Measure

The following example generates a treemap colored by the measure MSRP_US and sized by the measure COGS_US. Since the color-by field is a measure, the colors in the rectangles and legend are visualized as a gradient. The chart type is TREEMAP:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
BY BRAND
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=report, column=COGS_US,    bucket=size, $
type=report, column=MSRP_US,    bucket=color, $
type=report, column=PRODUCT_CATEGORY, bucket=detail, $
type=report, column=BRAND,    bucket=detail, $
END
```

The output is shown in the following image:

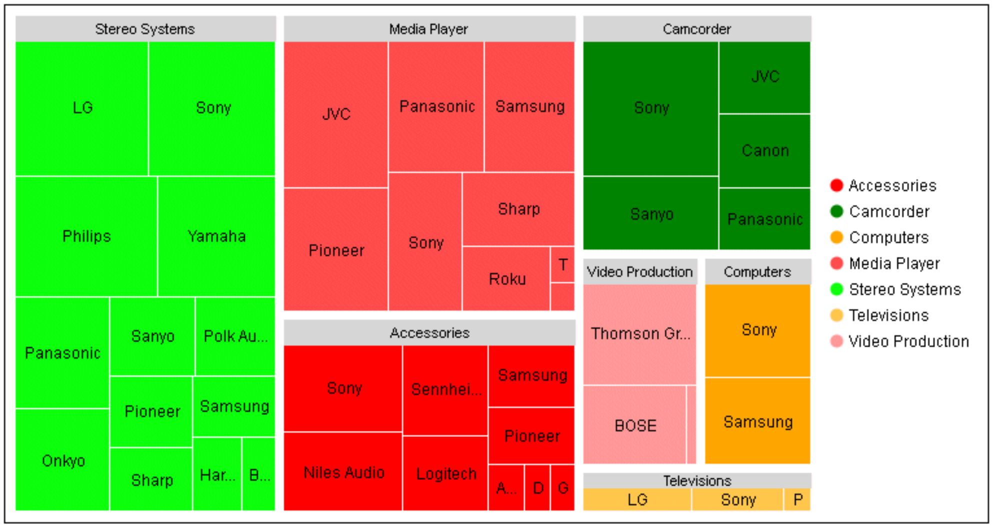


Example: Sample Treemap Colored by a Dimension

The following example generates a treemap colored by the first BY field. Since the color-by field is a dimension, discrete colors are generated for the rectangles and the legend. The second BY field (detail attribute category) accounts for the subdivisions of the colored boxes. Note that the prefix operator used in the request is also used when assigning the field to the category:

```
GRAPH FILE WF_RETAIL_LITE
SUM CNT.COGS_US
BY PRODUCT_CATEGORY
BY BRAND
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=report, column=product_category, bucket=color, $
type=report, column=brand, bucket=detail, $
type=report, column=cnt.cogs_us, bucket=size, $
END
```

The output is shown in the following image:



Specifying Chart Orientation

If you want to specify the orientation for a chart, include the CHART-ORIENTATION attribute in the TYPE=REPORT declaration of the WebFOCUS StyleSheet. The default orientation for charts is vertical.

Chart orientation is supported for bar, line, area, mekko, and boxplot charts.

Syntax: How to Specify Chart Orientation

```
TYPE=REPORT, CHART-ORIENTATION=orientation,$
```

where:

orientation

Specifies the chart orientation. Valid values are:

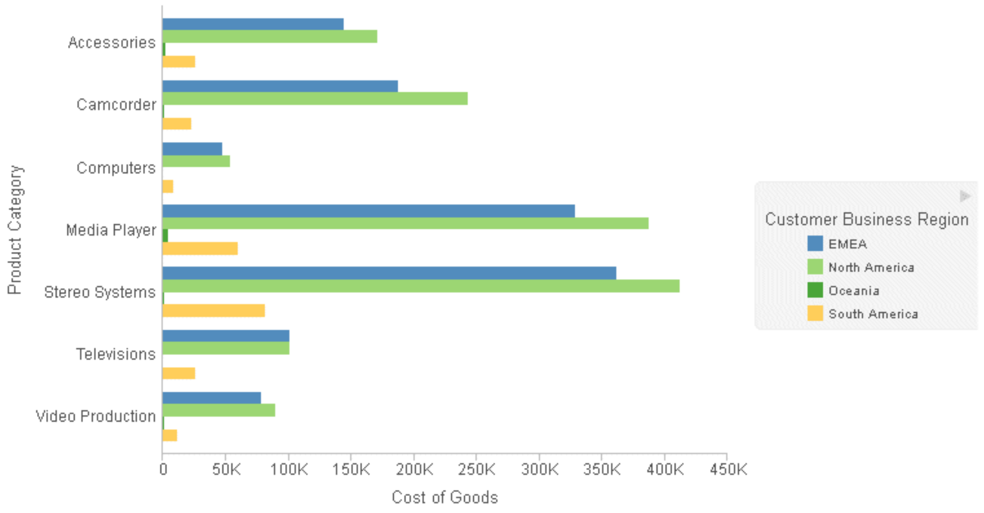
- **vertical**, which draws the x-axis on the bottom of the chart and the y-axis on the left side of the chart. This is the default value.
- **horizontal**, which draws the x-axis on the left side of the chart and the y-axis on the bottom of the chart.

Example: Specifying the Chart Orientation

The following request generates a bar chart whose orientation is horizontal. The chart type is BAR.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-ORIENTATION=horizontal,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Specifying Chart Layout for Bar, Line, and Area Charts

If you want to specify the layout (subtype) for a bar, line, or area chart, include the CHART-SERIES-LAYOUT attribute in the TYPE=REPORT declaration of the WebFOCUS StyleSheet. Layout refers to the positions of the risers in relation to each other.

- Bar charts support side-by-side, stacked, absolute, and percent layouts.
- Area charts support absolute, stacked, and percent layouts.
- Line charts support absolute, stacked, and percent layouts.

The default chart layout for bar charts is side-by-side, and the default chart layout for area charts and line charts is absolute.

Syntax: How to Specify Chart Orientation and Layout

```
TYPE=REPORT, CHART-SERIES-LAYOUT=layout,$
```

where:

layout

Defines the arrangement of risers for bar, line, and area charts. Valid values are:

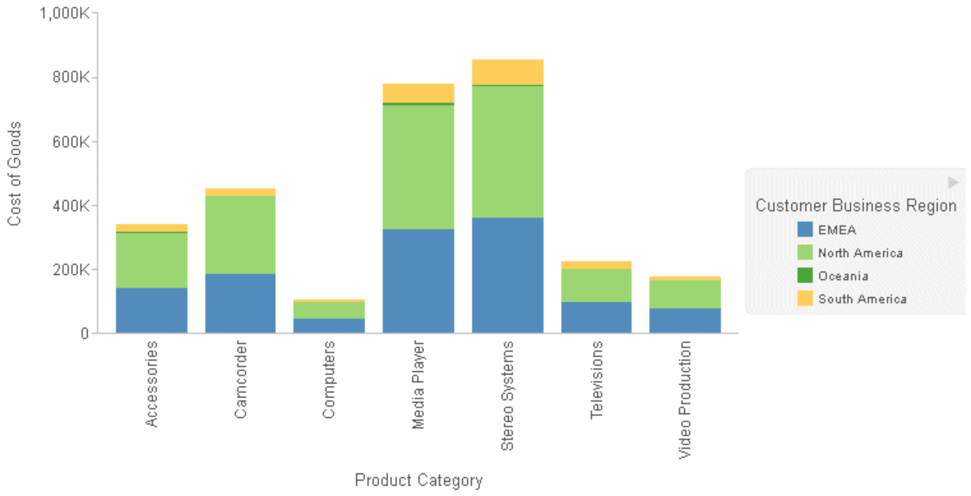
- **side-by-side** to place the risers next to each other (for vertical bar charts) or one on top of another (for horizontal bar charts). The default value is side-by-side. Side-by-side layout is supported for bar charts only.
- **stacked** (bar, line, and area) to stack the risers on top of each other, with the length of each riser representing its data value. Stacked layout is supported for bar, line, and area charts.
- **absolute** draws each riser at the same x-axis value (for a vertical chart) or y-axis value (for a horizontal chart). If the risers are different widths, they can be distinguished on the chart. This is the default chart layout. Absolute layout is supported for line and area charts.
- **percent** to stack the risers, with the length of each riser representing the percentage of the total for that riser, where the total of the stack adds up to 100%. Percent layout is supported for bar, line, and area charts.

Example: Specifying the Chart Layout

The following request generates a bar chart whose layout is stacked. The chart type is BAR.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=stacked,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Controlling the Width of an Absolute Bar Chart Inset

In an absolute bar chart, the bar centers are at the same x-axis point (for a vertical chart) or the same y-axis point (for a horizontal chart). The bar for the first series in the request is placed at the back, with the bars for each subsequent series placed on top of the previous one. The width of each bar added to the chart is scaled so that the bar in back of it remains visible on the sides. Using the absoluteInset property, you can define the width of the smallest bar as a percentage of the width of the largest bar. The other bars will be scaled accordingly

Syntax: How to Control the Width of an Absolute Bar Chart Inset

```
"blaProperties": {
   "absoluteInset": number
          }
```

where:

`absoluteInset:`*`number`*

Is a decimal number between 0 and 1 that represents the width smallest bar as a percentage of the width of the largest bar. The default value is 0.7, which represents 70%.

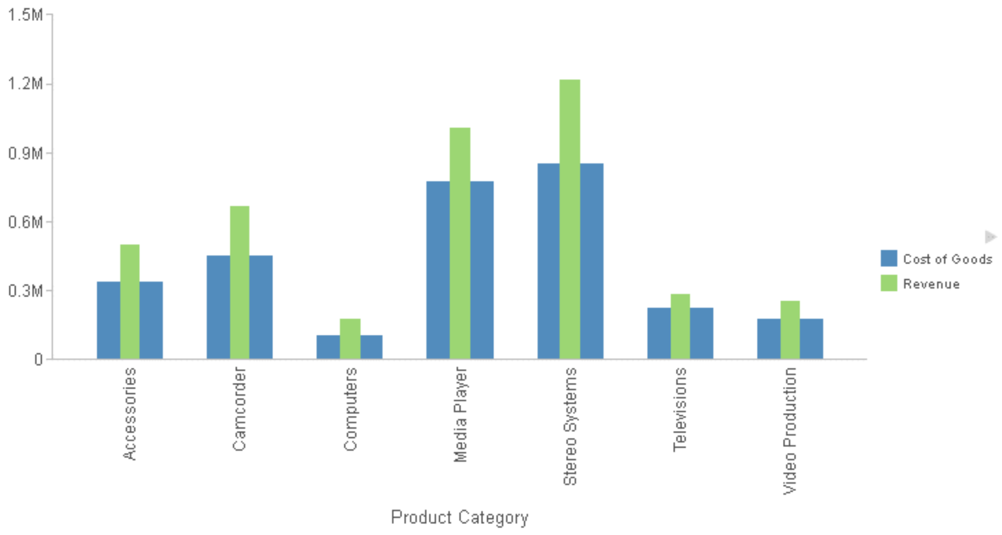
Note: You can disable the inset effect by setting the absoluteInset value to 1. This makes all of the bars the same width so that they completely overlap each other.

Example: Specifying an Inset for an Absolute Bar Chart

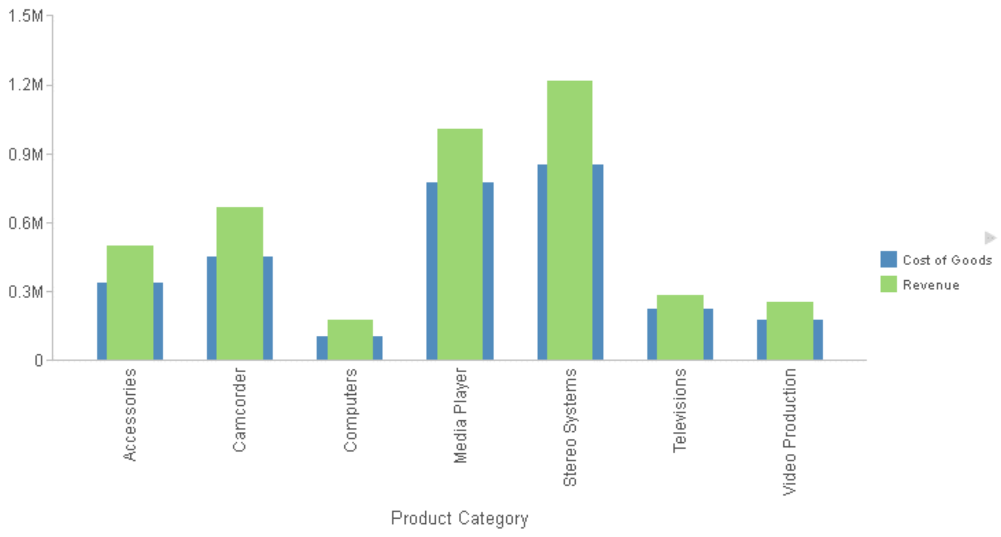
The following request makes the width of the smallest bar 30% the width of the largest bar.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=report, chart-series-layout=absolute,$
type=data, column=product_category, bucket=x-axis, $
type=data, column=cogs_us, bucket=y-axis, $
type=data, column=revenue_us, bucket=y-axis, $
*GRAPH_JS
"blaProperties": {"absoluteInset": 0.3}
*END
ENDSTYLE
END
```

The output is shown in the following image.



The following image shows the bar chart without the absoluteInset property.



Assigning Chart Data to Attribute Categories

Each field in the chart request is either a measure (value that can be aggregated) or a dimension (field that categorizes measures, used typically as sort fields in a chart request).

Some attribute categories can only be assigned to sort fields, some can only be assigned to measures, and some can be assigned to either.

The order of sort fields in the request and the type of chart being generated determine the category to which each field can be assigned.

The assignment of fields to attribute categories replaces the GRMERGE, GRMULTIGRAPH, GRLEGEND, and GRXAXIS parameters for merging charts. For information about converting requests with merge parameters to chart attribute syntax, see *Converting Requests to Chart Attribute Syntax* on page 993.

Syntax: How to Assign Chart Data to Attribute Categories

```
TYPE=DATA, COLUMN=colspec, bucket=category,$
```

where:

`COLUMN=colspec`

Is any valid column reference in a WebFOCUS StyleSheet.

For example, COLUMN=N1 represents the first column of output. This is usually the high-order BY field, as described in *Understanding Column Sequence Numbers* on page 145. Alternatively, you can specify the field name, for example, COLUMN=PRODUCT_CATEGORY. For information about WebFOCUS StyleSheet syntax, see the *Creating Reports With WebFOCUS Language* manual.

`bucket=`*`category`*

Describes the role this field is assigned in the chart.

Valid attribute category values vary by chart type. The list of all categories follows:

- **row** assigns the field values to rows in a matrix or grid.
- **column** assigns the field values to columns in a matrix or grid.
- **color** assigns a field to the legend. If it is a measure, it generates a gradient legend.
- **measure** assigns the field values as data values on the chart, not associated with an axis (as in a pie chart).
- **y-axis** assigns the field values to the vertical axis (if you are using the default value for CHART-SERIES LAYOUT).
- **x-axis** assigns the field values to the horizontal axis (if you are using the default value for CHART-SERIES LAYOUT).
- **size** assigns the field values to the size of the markers (bubble charts) or to a size-by field.
- **tooltip** adds the field values to the tooltip, All fields in the request are also part of the tooltip.
- **detail** assigns the field values to an additional low level sort field, which generates additional data points. You must add this field as the lowest sort field in the request.
- **slider** assigns the field values to a slider control. A slider creates an animation effect as you move the control along the slider bar. It is limited to one sort field only and should preferably be something time or sequence related such as *YEAR*.
- **page** assigns the field values to independent pages (separate charts). The page attribute category can be used for bursting and distributing with ReportCaster.
- **latitude** assigns the field values to the latitudes at which to display the symbols on bubblemap charts. This attribute corresponds to the y-axis attribute in a bubble chart.
- **longitude** assigns the field values to the longitudes at which to display the symbols on bubblemap charts. This attribute corresponds to the x-axis attribute in a bubble chart.

- **location** assigns the field values as the location values for the colored polygons on a choropleth or the symbols on a bubblemap.

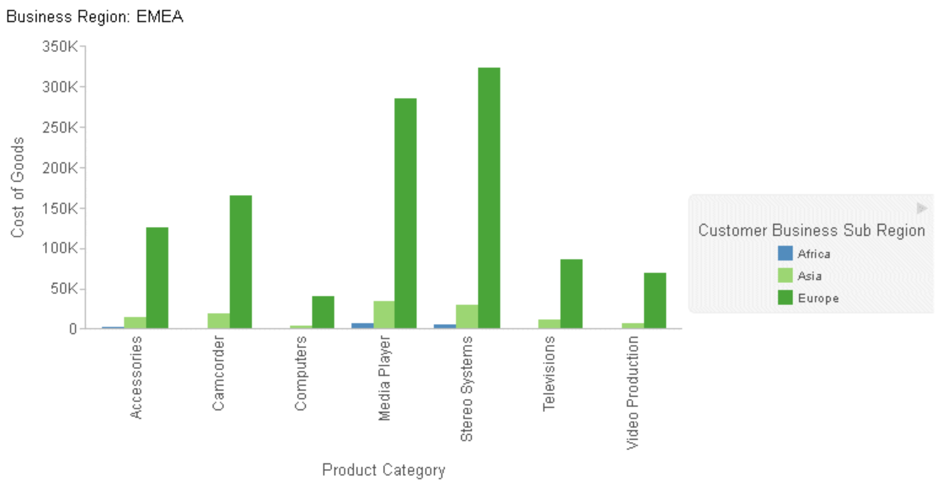
Some attribute categories can be omitted, if you do not want them on the chart output or if they do not apply to the type of chart being created. Some attribute categories can be assigned to multiple fields in the request (for example, multiple fields assigned to the y-axis in bar or line charts).

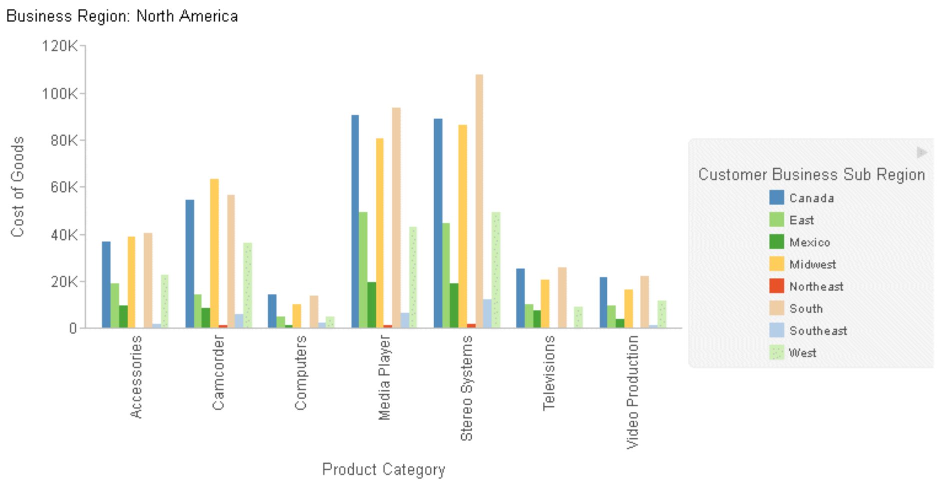
Example: Generating Chart Pages

The following request generates a separate bar chart for each business region. The BUSINESS_REGION field is added as the first sort field and is assigned to the page category. The business region value is also added to the heading, which is embedded in the chart:

```
GRAPH FILE WF_RETAIL_LITE
HEADING
"Business Region: <BUSINESS_REGION "
SUM COGS_US
BY BUSINESS_REGION
BY BUSINESS_SUB_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=X-AXIS, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=page,$
TYPE=DATA, COLUMN = BUSINESS_SUB_REGION, BUCKET=color,$
ENDSTYLE
END
```

The output is shown in the following image. A separate chart is generated for each business region:





Instead of adding a heading to indicate the business region, you can add the field to the tooltip category as well as the page category. For information and an example, see *Assigning a Field to Multiple Attribute Categories* on page 219.

Assigning Field Order Within a Category

Some categories can have multiple fields assigned to them. By default, the fields are assigned to a category in the order in which the assignments appear in the StyleSheet. For example, the first y-axis field assigned becomes the leftmost measure in a vertical bar chart. If you want to specify the order in which the fields should be assigned to the category, use an index number suffix. For example, using the following declarations, UNITS will come first regardless of its order in the request, since its suffix (.1) is explicitly set to be lower than that of BUDUNITS (.2):

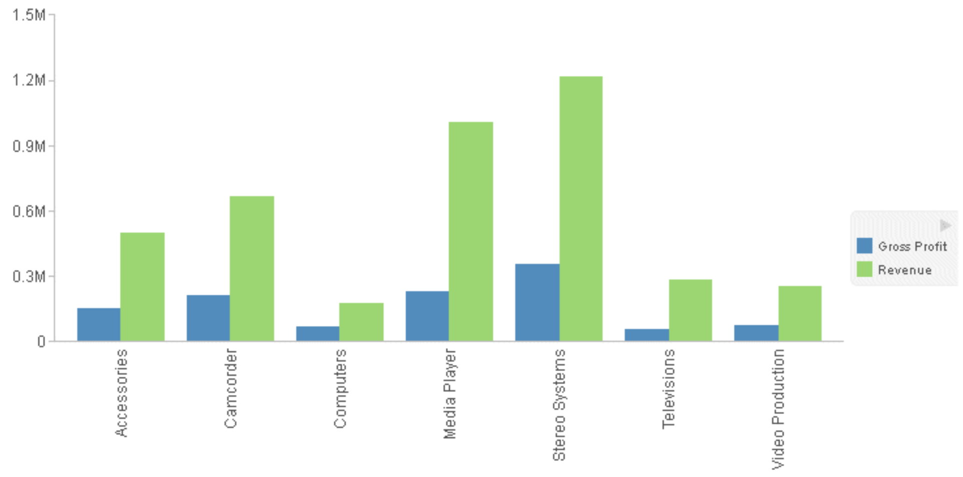
```
type=data, column=budunits, bucket=y-axis.2, $
type=data, column=units, bucket=y-axis.1, $
```

Example: Specifying Field Order

The following request generates a bar chart. the bars represent the GROSS_PROFIT_US and REVENUE_US measures. The bar representing GROSS_PROFIT_US is first because the StyleSheet declarations specify GROSS_PROFIT_US before REVENUE_US:

```
GRAPH FILE WF_RETAIL_LITE
SUM
GROSS_PROFIT_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=product_category, bucket=x-axis, $
type=data, column=gross_profit_us, bucket=y-axis, $
type=data, column=revenue_us, bucket=y-axis, $
ENDSTYLE
END
```

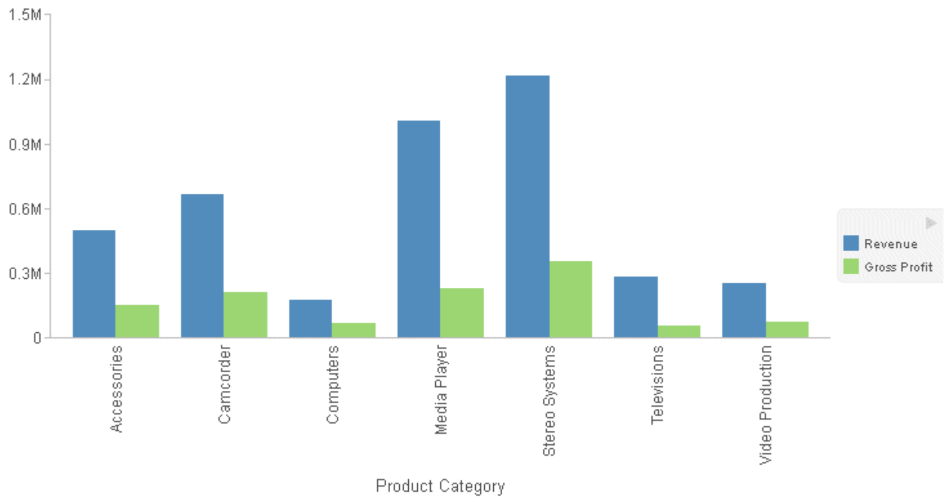
The output is shown in the following image:



The following version of the request displays the bars for REVENUE_US first by adding a .1 suffix to the attribute category for the REVENUE_US field:

```
GRAPH FILE WF_RETAIL_LITE
SUM
GROSS_PROFIT_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=product_category, bucket=x-axis, $
type=data, column=gross_profit_us, bucket=y-axis, $
type=data, column=revenue_us, bucket=y-axis.1, $
ENDSTYLE
END
```

The output is shown in the following image:



Specifying Multiple X-Axes and Y-Axes

An attribute category can have a subscript, enclosed in parentheses, if there is more than one version of it. For example, to indicate the y2-axis, use the following:

```
y-axis(2)
```

Any number of y-axes or x-axes can be specified using subscripts.

A bar, line, or area chart can have split y-axes or dual y-axes, depending on whether the BLA property splitY is true or false. For more information, see *Controlling the Display of Multiple Y-Axes in Bar, Line, or Area Charts* on page 507.

A bubble or scatter chart can have a split x-axis or a split y-axis, or both, by assigning fields to the appropriate axes using subscripts. For example:

```
type=data, column=COGS_US,      bucket=x-axis, $
type=data, column=REVENUE_US,   bucket=x-axis(2), $
type=data, column=DISCOUNT_US,  bucket=y-axis, $
type=data, column=MSRP_US,      bucket=y-axis(2), $
```

A category can have both a subscript and a suffix. For example, to indicate the first field in the y2-axis category, use the following:

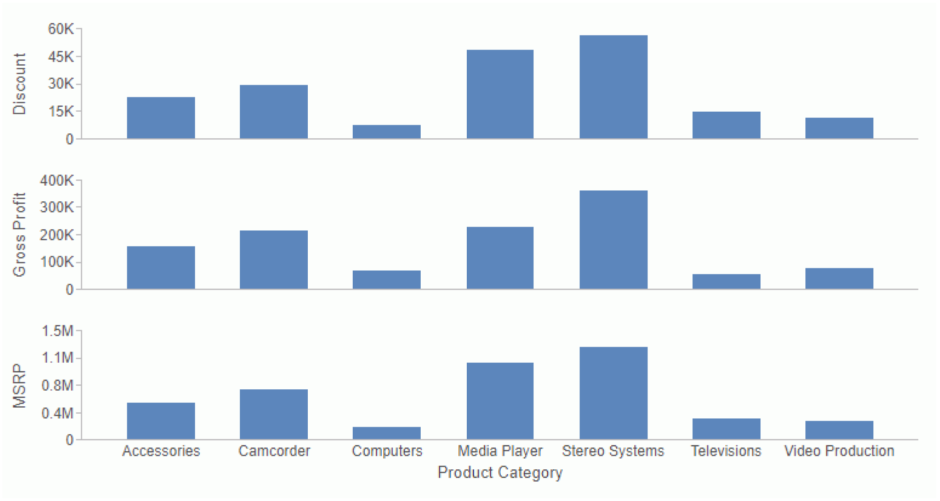
```
y-axis(2).1
```

Example: Generating Split Y-Axes for a Bar Chart

The following request assigns the measure DISCOUNT_US to the y-axis, the measure GROSS_PROFIT_US to the y2-axis, and the measure MSRP_US to the y3-axis. The axes are drawn as split y-axes (stacked on top of each other) because the blaProperties:splitY property is set to true. All of the measures are rendered with the same color because of the chart-color-measures=off StyleSheet attribute:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=product_category, bucket=x-axis,$
type=data, column=discount_us, bucket=y-axis,$
type=data, column=GROSS_PROFIT_US, bucket=y-axis(2),$
type=data, column=MSRP_US, bucket=y-axis(3),$
type=report, chart-color-measures=off,$
*GRAPH_JS
"blaProperties": {"splitY": true},
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is shown in the following image:

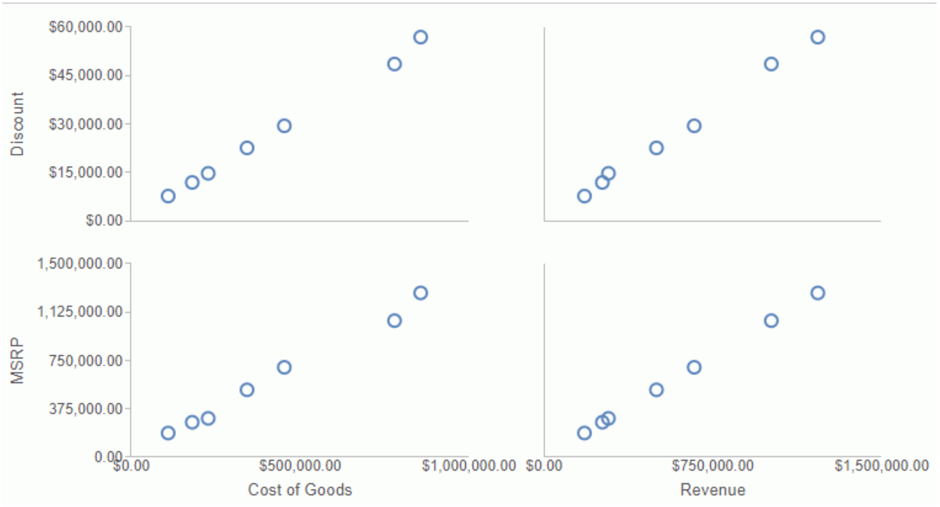


Example: Generating Split X- and Y-Axes for a Scatter Chart

The following request generates a scatter chart with split x- and y-axes.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,    $
type=data, column=PRODUCT_CATEGORY,   bucket=detail,    $
type=data, column=COGS_US,            bucket=x-axis,    $
type=data, column=REVENUE_US,         bucket=x-axis(2), $
type=data, column=DISCOUNT_US,        bucket=y-axis,    $
type=data, column=MSRP_US,            bucket=y-axis(2), $
END
```

The output is shown in the following image.



Assigning a Field to Multiple Attribute Categories

To assign a field to multiple attribute categories, enclose the list of categories in parentheses and separate them with a blank space:

```
TYPE=DATA, COLUMN=colspec, bucket=(category1 category2 ...),$
```

For example, the following declaration adds a field to both the page and tooltip categories:

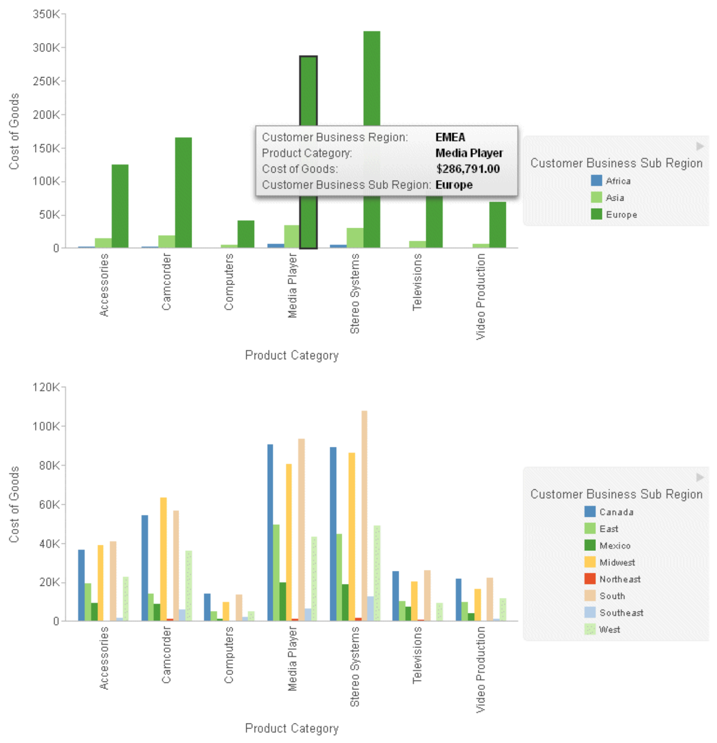
```
TYPE=DATA, COLUMN=N1, BUCKET=(page tooltip),$
```

Example: Assigning a Field to the Page and Tooltip Categories

The following request generates a separate bar chart for each business region. The BUSINESS_REGION field is added as the first sort field and is assigned to the page and tooltip categories:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY BUSINESS_SUB_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_SUB_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=(page tooltip),$
ENDSTYLE
END
```

The following image shows that the business region generates a new page and has been added to the tooltip:



Order of Attribute Category Assignments for Sort Fields

The attribute category assignments for sort fields must be in the following order:

1. page
2. slider
3. row
4. column
5. color
6. detail
7. location (for maps) or x-axis (for non-maps)

Note:

- The latitude and longitude attributes in a bubblemap correspond to the x-axis and y-axis attributes in a bubble chart.

- For a heatmap, the y-axis must be a sort dimension and be placed before the x-axis sort field in the request.
- In bubble and scatter charts, both the x-axis and y-axis can be sort dimensions. They can be specified in any order in the request.

Summary of Supported Attribute Categories for Each Chart Type

All chart types except map charts and tagclouds support multiple row and column BY field assignments. All chart types support multiple page and tooltip BY field assignments. One BY field can be assigned to the slider category.

With this syntax, ACROSS sort fields are not supported. All sort fields must be BY fields.

The following table lists the other supported attribute categories for each chart type.

Chart Type	Attribute Category and Description
AREA	**x-axis.** Sort field, typically a time-based dimension, such as YEAR. **y-axis.** Measure. Multiples supported. Two y-axes supported. **color.** Optional. The area color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. Multiples supported.
BAR	**x-axis.** Sort field. **y-axis.** Measure. Multiples supported. Multiple y-axes supported. **color.** Optional. The bar color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. Multiples supported. **size.** Optional. The bar width will depend on the value of the measure assigned to the size attribute.

Chart Type	Attribute Category and Description
BOXPLOT	**x-axis.** Sort field. **min.** Measure. Minimum value. **lower.** Measure. First quartile. **median.** Measure. Median value. **upper.** Measure. Third quartile. **max.** Measure. Maximum value. **color.** Optional. The box color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. Multiples supported.
BUBBLE	**x-axis.** Measure or sort field. **y-axis.** Measure or sort field. Multiples supported. **color.** Optional. The symbol color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. **size.** Measure. The bubble size will depend on the value of the measure assigned to the size attribute. **detail.** Sort field (optional). Adds data to the chart. Multiples supported.

Chart Type	Attribute Category and Description
BUBBLEMAP	**location.** Dimension sort field containing the geolocation at which the symbols on the map should appear. Either location or latitude and longitude can be used. **latitude.** Sort or measure field containing the latitude for the symbols. Either location or latitude and longitude can be used. **longitude.** Sort or measure field containing the longitude for the symbols. Either location or latitude and longitude can be used. **size.** Measure field used to size the symbols. **color.** Optional. The symbol color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete.
CHOROPLETH	**location.** Dimension sort field containing the geolocation at which the symbols on the map should appear. **color.** The polygon color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete.
DATAGRID	**row.** Sort dimension or measure field whose values represent the rows of the grid. Multiples supported. **column.** Sort dimension or measure field whose values represent the columns of the grid. Multiples supported. **measure.** Field whose values display in the cells of the grid. Multiples supported.
FUNNEL	**measure.** Measure field that sizes the funnel slices. **color.** Sort field.
GAUGE	**measure.** Field that determines the position of the gauge needle or the length of the band, depending on the layout.

Chart Type	Attribute Category and Description
HEATMAP	**y-axis.** Sort dimension field. Must be specified before the x-axis field in the request. **x-axis.** Sort field. **color.** Measure. Rectangle color will depend on the value of this measure. The color will be visualized as a gradient.
LINE	**x-axis.** Sort field, typically a time-based dimension, such as YEAR, **y-axis.** Measure. Multiples supported. Multiple y-axes supported. **color.** Measure (optional). The line color (and marker color, if markers display) will depend on the measure assigned to the color attribute. The color will be visualized as a gradient. Multiples supported. **size.** Measure (optional). The marker size, if markers are visible, will depend on the value of the measure assigned to the size attribute. If markers are not visible, the width of the line represents the size measure
MARKER	**size.** Measure. The size of the marker in each cell will depend on the value of the measure assigned to the size attribute. **color.** Measure. The marker color will depend on the measure assigned to the color attribute. The color will be visualized as a gradient.
MEKKO	**x-axis.** Sort field. **y-axis.** Measure. Multiples supported. **color.** Optional. The bar color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. Multiples supported.
PIE	**measure.** Determines the size of each slice. **color.** Sort field, used to color the slices. **size.** Supported for matrix charts only. Makes the pie in each cell larger or smaller depending on the measure assigned.

Chart Type	Attribute Category and Description
SCATTER	**x-axis.** Measure or dimension sort field. **y-axis.** Measure or dimension. **color.** The marker color will depend on the field assigned to the color attribute. If it is a measure, the color will be visualized as a gradient. If it is a sort field, the colors will be discrete. If no field is assigned to this category, all of the markers will be the same color. **size.** Measure (optional). The marker size will depend on the value of the measure assigned to the size attribute. **detail.** Sort field (optional). Adds data to the chart. Multiples supported.
STREAM	**x-axis.** Sort field. **y-axis.** Measure. Multiples supported. **color.** Colors the areas and legend (optional). If the field is a measure, the colors will be visualized as a gradient. If it is a dimension, the colors will be visualized as discrete colors. Multiples are supported.
TAGCLOUD	**detail.** Sort dimension that contains the tags to be displayed. **size.** Measure that controls the size of the tags. **color.** Measure that controls the color of the tags. Tagclouds do not support multiple series and legends, so this category cannot be assigned to a sort field.
TREEMAP	**color.** Measure or sort field. Rectangle color will depend on the value of this measure or dimension. The color will be visualized as a gradient for a measure or discrete colors for a dimension. **size.** Measure. Rectangle size will be determined by the value of this measure. **detail.** Dimension sort field. Multiples supported.

Chapter 5

Chart-Wide Properties

This chapter describes how to control the general appearance of a chart.

In this chapter:

Chart-Wide Properties Overview

These properties control the general appearance of a chart.

This code segment shows the default settings:

```
"catchErrors": true,
"fill": {
"color": "white"
},


"border": {
    "width": 1,
    "color": "black",
    "dash": ""
    },
"chartFrame": {
    "fill": {
        "color": "transparent"
    },
    "border": {
        "width": 0,
        "color": "transparent",
        "dash": ""
    },
    "shadow": false
},


"dataSubset": {
    "startGroup": "undefined",
    "stopGroup": "undefined"
},


"swapData": false,


"swapDataAndLabels": false,


"riserCycleEndLightness": 0.8,


"chartsPerRow": undefined,


"depth": undefined,
"riserDepthGap": 0.2,
"riserShadow": false,
"riserBevel": undefined,


"groupLabels": "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split(''),
```

Formatting the Chart Border

The border property defines the border of the drawing area.

The chart border encloses the drawing area, which contains the chart and its frame, the axis labels, the legend, and the chart title, subtitle, and footnote,

Syntax: How to Format the Chart Border

```
"border": {
   "width": number,
   "color": "color",
   "dash": "string"
}
```

where:

`"width": number`

Is a number of pixels that defines the width of the border around the drawing area. The default value is zero (no border).

`"color": "color"`

Can be:

A string, enclosed in single quotation marks, that defines a color by name or numeric specification, or that defines a gradient. The default value is 'transparent'.

A JSON gradient definition.

For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"dash": "string"`

Is a string that defines the dash style of the border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

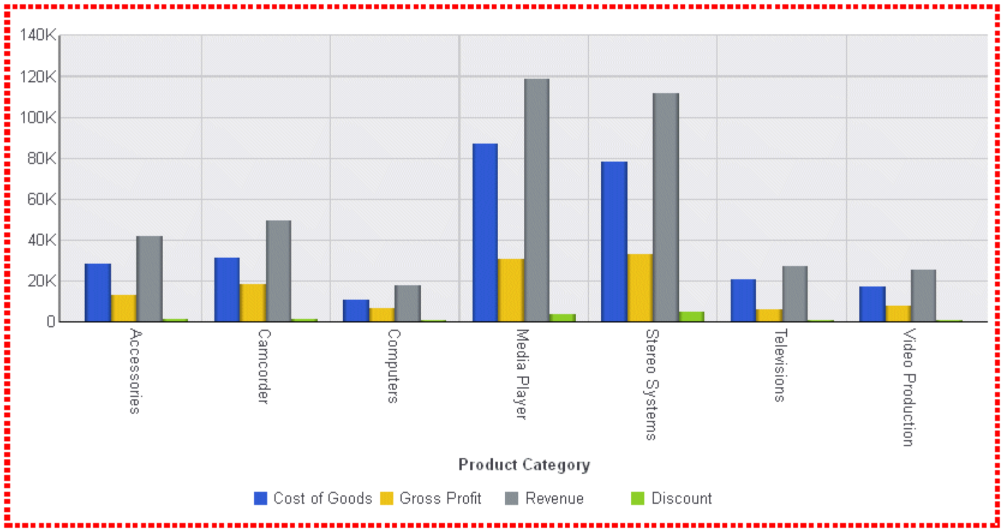
Multiple dashes can be defined within the dash string (for example, "2 4 4 2"), in which case, the dash types will alternate along the border.

Example: Formatting the Chart Border

The following request generates a border that is red, 4 pixels wide, and has a dash in which the length of each dash is 4 pixels and the gap between dashes is 2 pixels:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {
    "width": 4,
    "color": "red",
    "dash": "4 2"
    }
*END
ENDSTYLE
END
```

The output is:



Defining How to Handle JavaScript Errors

The catchErrors property defines how JavaScript errors are handled. Typical errors can include invalid property settings that do not allow the engine to parse the JavaScript. When catchErrors is false, you can use the JavaScript console (for example, in Chrome, press Ctrl + Shift + J. In Firefox, press F12, then the Console tab) to determine the exact location of the error.

Syntax: How to Specify How to Handle JavaScript Errors

```
"catchErrors": boolean
```

where:

`"catchErrors":` *`boolean`*

Can be:

- true, which displays errors as HTML text where the chart would otherwise be drawn. This is the default value. Subsequent JavaScript code in the calling application continues to run.
- false, which passes errors to the calling application. If the calling application does not catch the error, all subsequent JavaScript execution terminates.

Formatting the Chart Frame

The chartFrame property defines the fill color and border of the chart frame.

The chart frame encloses the visual area of the chart. It does not enclose the axis labels, the legend, or the chart title, subtitle, and footnote,

Note: Frames are not supported for pie charts. Use background fill instead.

Syntax: How to Format the Chart Frame

```
"chartFrame": {
   "fill": {
      "color": "color"
           },
   "border": {
      "width": number,
      "color": "string",
      "dash": "string"
            },
   "shadow": boolean
              }
```

where:

`"fill": {"color": "color" }`

Is the fill color. Can be:

A color string, enclosed in single quotation marks, specifying a name or numeric specification string, or a gradient defined by a string. The default value is 'transparent'.

A JSON object that defines a gradient.

For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"border":`

Defines the properties of the frame border.

`"width": number`

Is the width of the frame border in pixels. The default value is zero (no border).

`"color": "string"`

Is the color of the border around the frame. Can be:

A color string, enclosed in single quotation marks, specifying a name or numeric specification string, or a gradient defined by a string. The default value is 'transparent'.

A JSON object that defines a gradient.

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Multiple dashes can be defined within the string (for example, "2 4 4 2"), in which case, the dashes alternate around the border.

`"shadow": `*`boolean`*

Enables or disables the shadow. Can be:

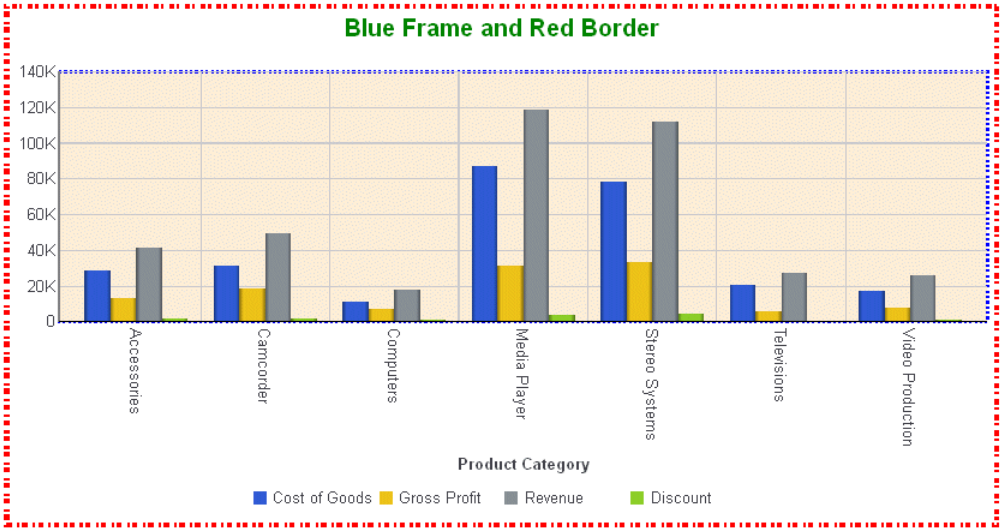
- ❑ true, which enables a default shadow on the chart frame.
- ❑ <u>false</u>, which disables a default shadow. The default value is false.

Example: Formatting the Chart Frame

The following request generates a chart frame whose fill color is antique white and whose border is a blue dashed line. It also generates a border that is a red dashed line with two different dash sizes that alternate around the border:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"title": {"visible": true,
    "text": "Blue Frame and Red Border",
    "color": "green",
    "font": "bold 14pt Sans-Serif"
    },
"chartFrame": {"fill": {"color": "antiquewhite"},
    "border": {
        "width": 2,
        "color": "blue",
        "dash": "2 2"
        }},
"border": {
    "width": 4,
    "color": "red",
    "dash": "2 4 4 2"}
*END
ENDSTYLE
END
```

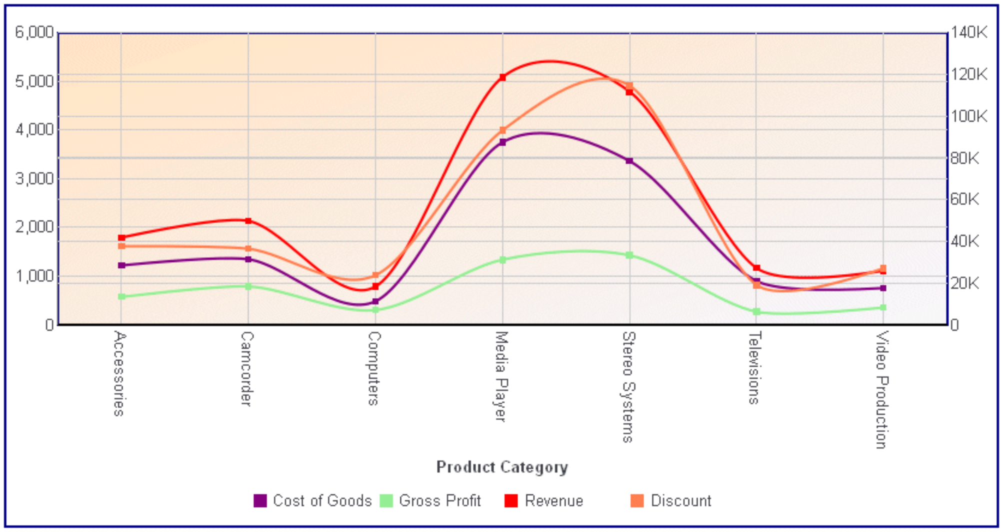
The output is



The following example applies a linear gradient fill to the chart frame that transitions from bisque to ghost white:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"blaProperties": {"lineConnection": "curved"},
"border": {"width": 2, "color": "navy"},
"chartFrame": {"fill": {"color":
    {"type": "linear",
        "start": {"x": "0%", "y": "0%"},
        "end": {"x": "100%", "y": "100%"},
        "stops": [[0, "bisque"], [1, "ghostwhite"]]}},
    "border": {"width": 2, "color": "navy"}},
"series": [
    {"series": 0, "color": "purple"},
    {"series": 1, "color": "lightgreen"},
    {"series": 2, "color": "red"},
    {"series": 3, "color": "coral"},
    {"series": 4, "color": "tan"}
    ]
*END
ENDSTYLE
END
```

The output is:



Controlling the Number of Charts in a Horizontal Row

For pie, gauge, and funnel charts that can have multiple charts in a single draw area, the chartsPerRow property defines how many charts to draw in a horizontal row.

Syntax: How to Control the Number of Charts Per Row

```
"chartsPerRow": number
```

where:

`"chartsPerRow": number`

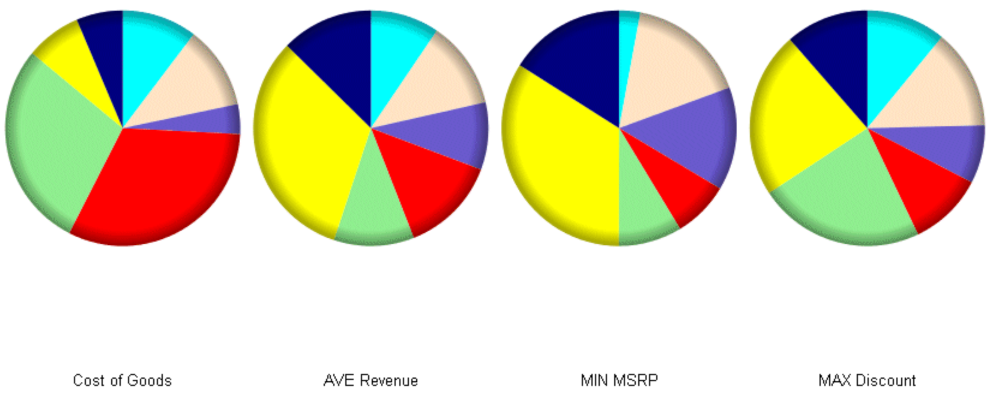
Is the number of charts to draw in a horizontal row. The default value is undefined.

Example: Controlling the Number of Charts Per Row

The following request generates four pie charts. The chartsPerRow property places them all on one horizontal row:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US AVE.REVENUE_US MIN.MSRP_US MAX.DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"chartsPerRow": 4,
"legend": {"visible": false},
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "bisque"},
    {"series": 2, "color": "slateblue"},
    {"series": 3, "color": "red"},
    {"series": 4, "color": "lightgreen"},
    {"series": 5, "color": "yellow"},
    {"series": 6, "color": "navy"},
    {"series": 7, "color": "lavender"},
    {"series": 8, "color": "limegreen"},
    {"series": 9, "color": "red"}
    ]
*END
ENDSTYLE
END
```

The output is:



Defining the Range of Data to Draw in the Chart

The dataSubset properties define the range of data to draw in the chart. They can be used with the interaction properties (for example, mouseDrag: pan) to define the range of data that is visible following a mouseDrag interaction.

Syntax: How to Define the Range of Data to Draw in the Chart

```
"dataSubset": {
   "startGroup": number,
   "stopGroup": number,
}
```

where:

`"startGroup": number`

Is a zero-based starting group number to show. The default value is undefined.

`"stopGroup": number`

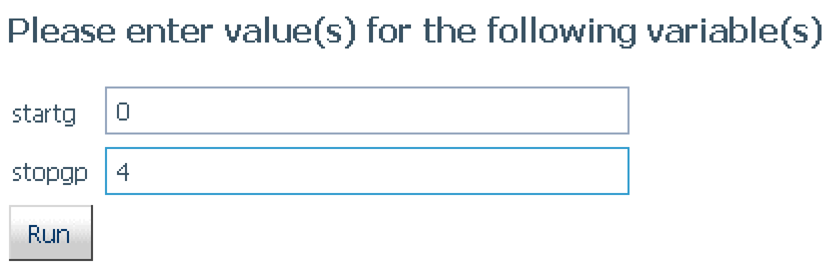
Is a zero-based number identifying the first group that should not be shown. The default value is undefined.

Example: Defining the Range of Data to Show in the Chart

The following request uses Dialogue Manager variables to enable the user to specify the data subset to draw:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"title": {"visible": true,
    "text": "Start Group: &startg, Stop Group: &stopgp",
    "color": "blue"},
"dataSubset": {
    "startGroup": "&startg",
    "stopGroup": "&stopgp"
    }
 *END
ENDSTYLE
END
```

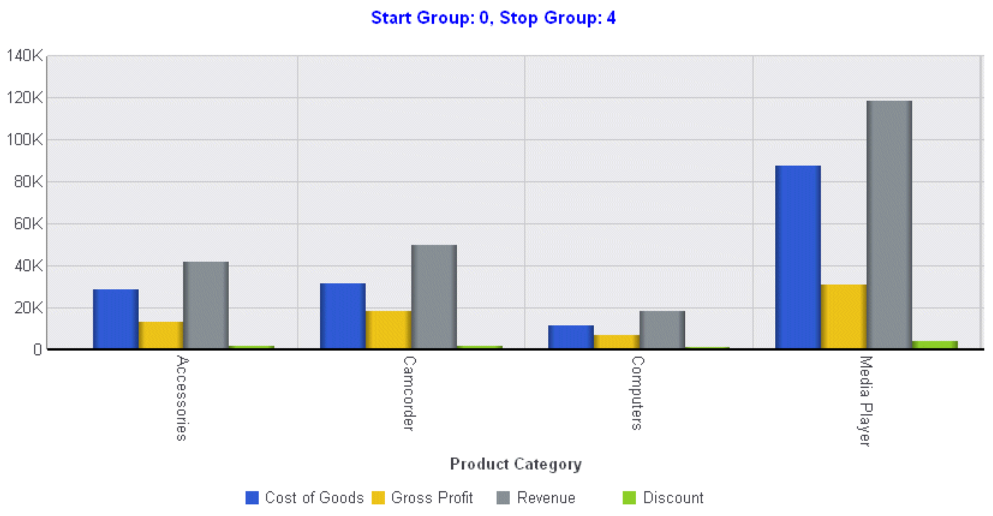
When the request is run, the user enters the start group number and the stop group number, in this case 0 and 4:


Please enter value(s) for the following variable(s)

startg	0
stopgp	4

Note: You can use the WebFOCUS amper auto-prompting facility to create a launch page that prompts users for the amper variables necessary to execute a procedure. With this facility, you can generate dynamic or static drop-down lists of allowed values and can validate the values entered by the user. For more information, see the *Developing Reporting Applications* manual.

When the user clicks *Run*, the following chart is generated that shows groups 0 through 3:



Applying Depth to Charts

The depth property applies a 2.5D depth effect to bar charts, line charts, area charts, pie charts, and any 'bar-like' chart.

Syntax: How to Apply Depth to Charts

```
"depth": number
```

where:

`"depth": number`

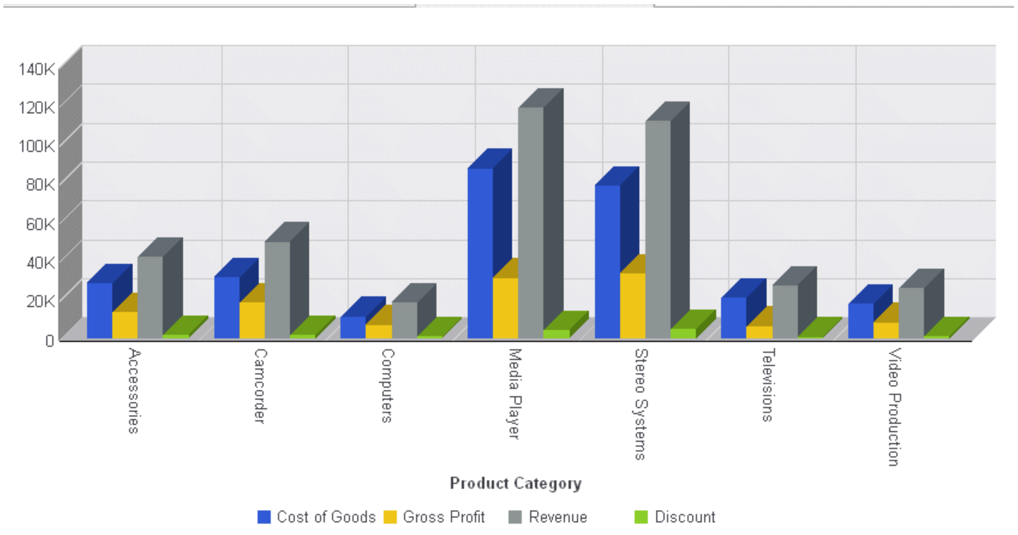
Is a number from 0 to 100 that specifies the amount of depth as a percent of the available space to apply to risers and the chart frame. Use zero or undefined for no 2.5D depth effect. The default value is *undefined*.

Example: Applying Depth to a Chart

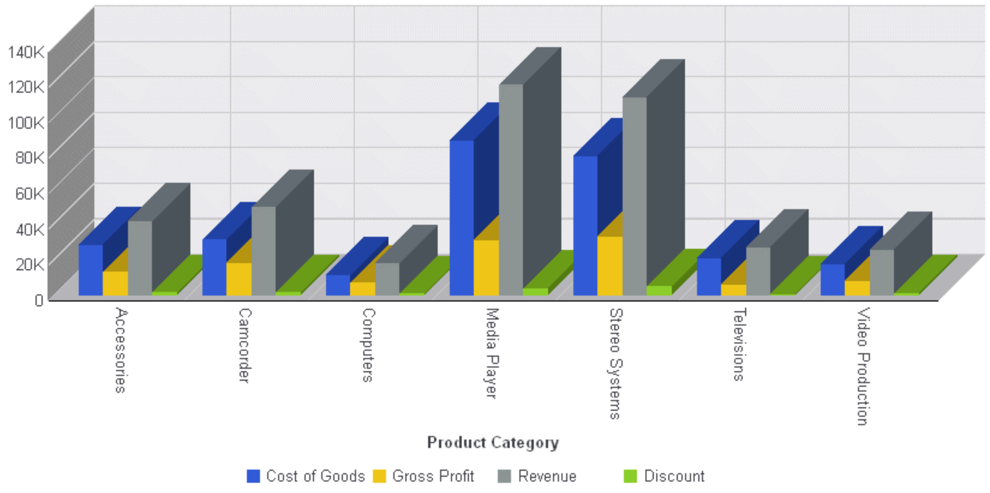
The following request generates a vertical bar chart with a depth property of 25:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
  "depth": 25,
*END
ENDSTYLE
END
```

On the output, the chart has been drawn with 2.5D affect:



Changing the depth value to 50 draws the chart with a deeper 2.5D affect:



Applying a Color or Gradient to the Draw Area

The fill property defines a color or gradient to be applied to the draw area.

Syntax: How to Apply a Color or Gradient to the Draw Area

```
"fill": {
        "color": "color"}
```

where:

`"color": "color"`

Defines a fill color. Can be:

A string that defines a color by name or numeric specification string, or a gradient string. The default value is "white".

A JSON gradient definition object.

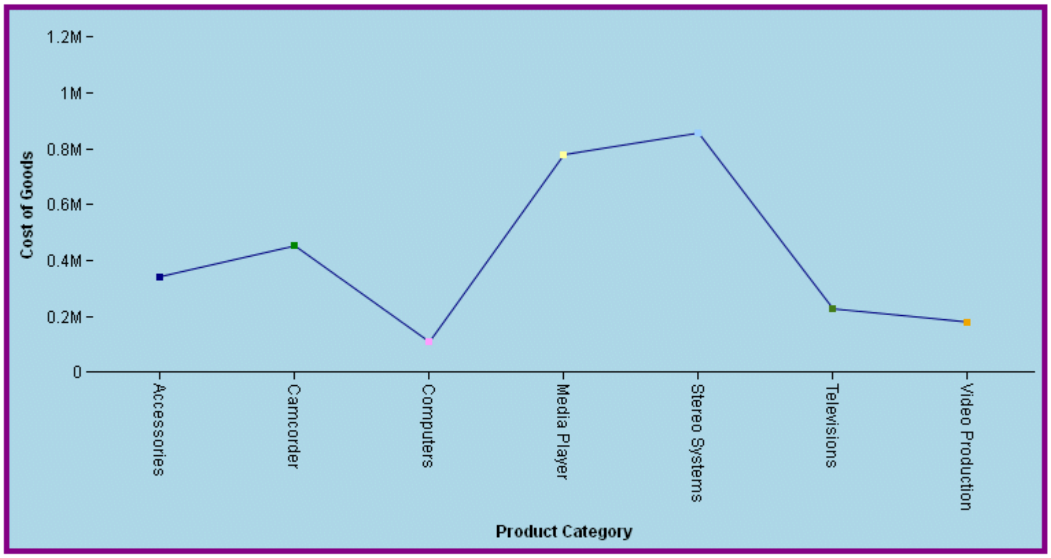
For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

Example: Specifying a Fill Color or Gradient for the Draw Area

The following request generates a vertical line chart with a light blue fill color:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {"color": "LightBlue"},
"border": {"width": 4, "color": "purple"},
"series": [
    {"series": 0, "color": "navy"},
    {"series": 1, "color": "green"}
    ]
*END
ENDSTYLE
END
```

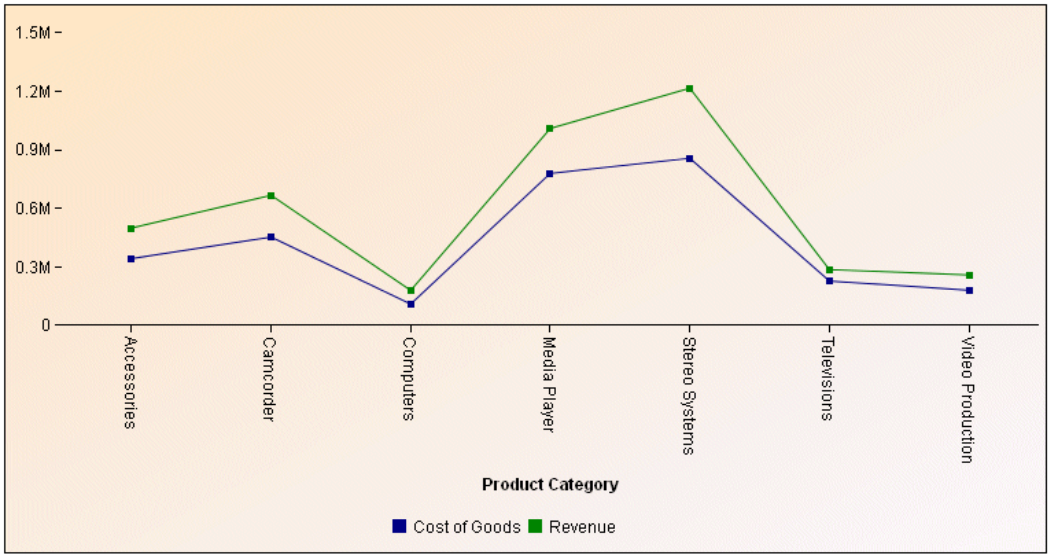
The output is:



The following request defines a JSON object that applies a linear gradient to the draw area:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"fill": {"color": {"type": "linear",
    "start": {"x": "0%", "y": "0%"},
    "end": {"x": "100%", "y": "100%"},
    "stops": [[0, "bisque"], [1, "ghostwhite"]]}
    },
"series": [
    {"series": 0, "color": "navy"},
    {"series": 1, "color": "green"}
    ]
*END
ENDSTYLE
END
```

The gradient in the draw area transitions from bisque to ghost white:



Defining Group Labels

For all chart types except histogram and tagcloud, the groupLabels property defines the group labels that appear on the ordinal axis. For a tagcloud chart, this property defines the labels that appear in the chart, according to the data values. For all chart types except tagcloud, group labels can be defined as a string or an array of strings. For a tagcloud chart, group labels must be defined as an array of strings.

Syntax: How to Define Group Labels

```
"groupLabels": "string"
```

or

```
"groupLabels": ["string", ..., "string"]
```

where:

```
"groupLabels": "string"
```

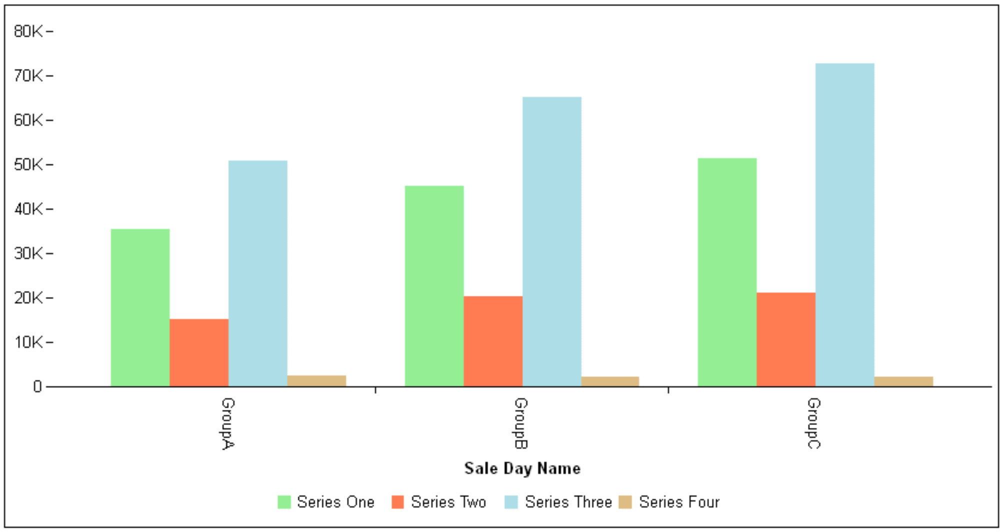
Is a string or an array of strings. A string will be split into an array along any space character. The default value is "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split('').

Example: Defining Group Labels as a String

The following request specifies the group labels as the string 'GroupA GroupB GroupC'. The blank spaces indicate where one group label ends and the next starts:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"groupLabels": "GroupA GroupB GroupC",
"series": [
    {"series": 0, "label": "Series One", "color": "lightgreen"},
    {"series": 1, "label": "Series Two", "color": "coral"},
    {"series": 2, "label": "Series Three", "color": "lightblue"},
    {"series": 3, "label": "Series Four", "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:

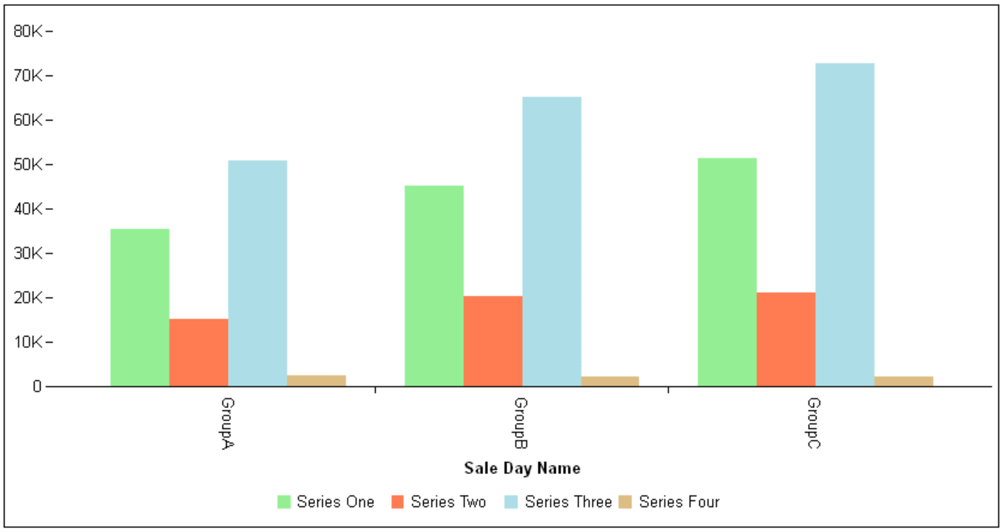


Example: **Defining Group Labels as an Array**

The following request specifies the group labels as the array ['GroupA', 'GroupB', 'GroupC']:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"groupLabels": ["GroupA", "GroupB", "GroupC"],
"series": [
    {"series": 0, "label": "Series One", "color": "lightgreen"},
    {"series": 1, "label": "Series Two", "color": "coral"},
    {"series": 2, "label": "Series Three", "color": "lightblue"},
    {"series": 3, "label": "Series Four", "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Applying a Bevel to Risers, Markers, and Slices

The riserBevel property applies a bevel to risers in a bar chart, markers in bubble and scatter charts, and slices in a pie chart.

Syntax: How to Apply a Bevel to Risers, Markers, and Slices

```
"riserBevel": "string"
```

where:

`"riserBevel": "string"`

Is a string that defines the bevel type. The default value is undefined.

- **For bar charts**, the values can be:

  `"bevel"`

  Applies a slight angle at the edges of the risers, giving them a beveled appearance.

  `"cylinder"`

  Draws the risers as cylinders with gradient color.

`"darken"`

Applies a gradient to the risers that darkens toward the bottom for vertical risers, and darkens toward the left for horizontal risers.

`"darkenInverted"`

Applies a gradient to the risers that darkens toward the top for vertical risers, and darkens toward the right for horizontal risers.

`"lighten"`

Applies a gradient to the risers that lightens toward the bottom for vertical risers, and lightens toward the left for horizontal risers.

`"lightenInverted"`

Applies a gradient to the risers that lightens toward the top for vertical risers, and lightens toward the left for horizontal risers.

- **For bubble and scatter charts**, the values can be:

`"darken"`

Applies a circular gradient to the markers that transitions from the original marker color to a darker color.

`"lighten"`

Applies a circular gradient to the markers that transitions from the original marker color to a lighter color.

- **For pie charts**, the values can be:

`"bevel"`

Applies an angle at the outer edge of the pie.

`"cylinder"`

Makes the pie look more cylindrical using a darkening gradient effect.

`"donut"`

Draws the pie with a donut effect if it has a hole.

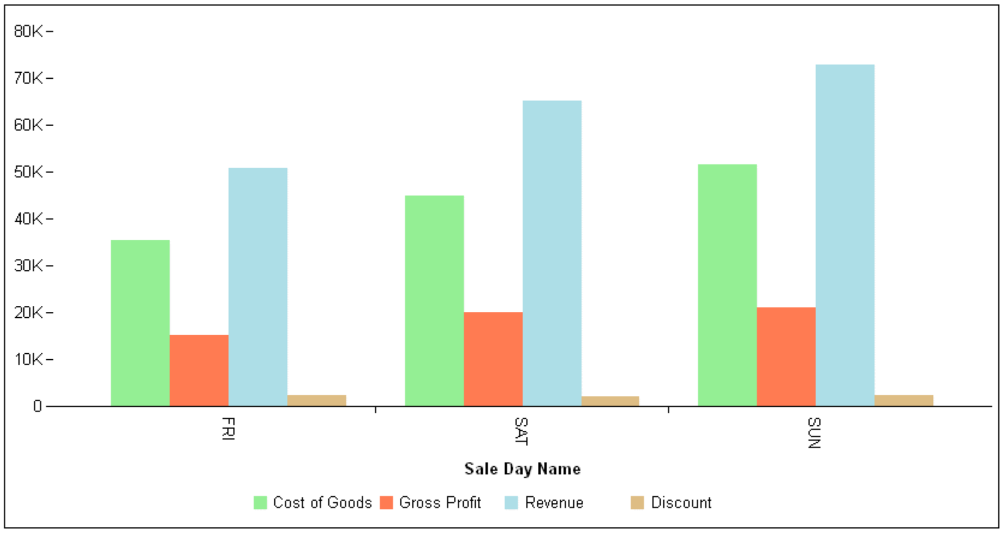
Bevel types *darken* and *lighten* will also be applied to legend markers (if visible).

Example: Generating a Bar Chart With Bevel Effects

The following request generates a bar chart with no bevel effect:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

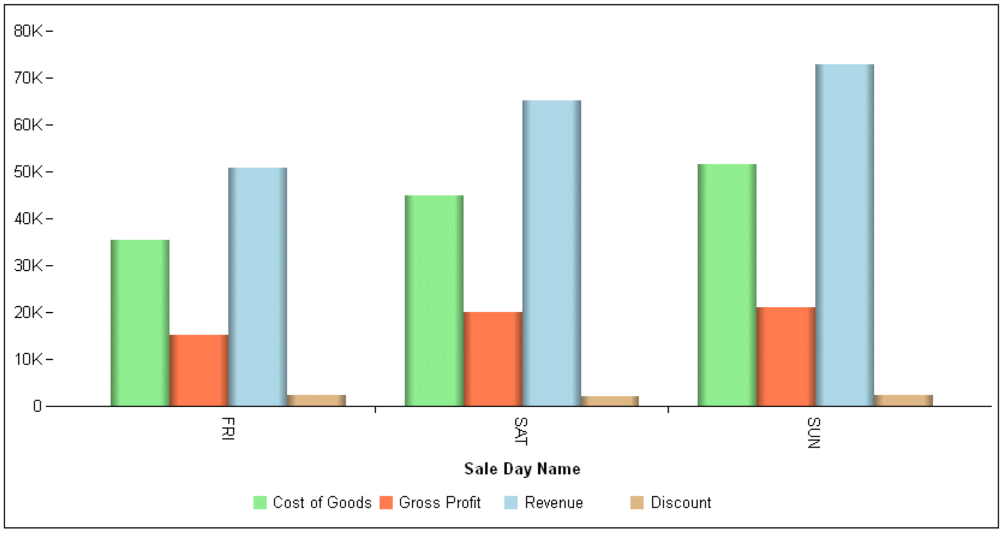
The output is:



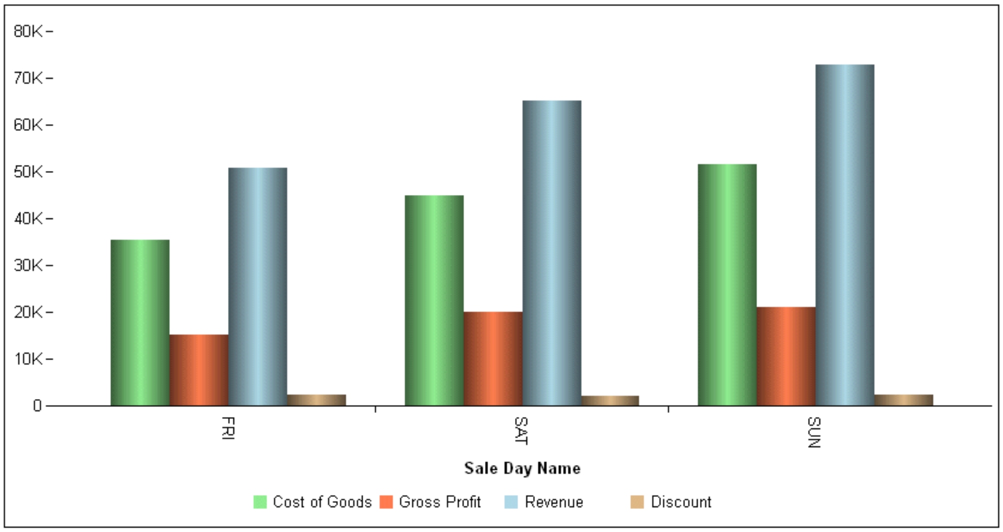
The following version of the request applies the "riserBevel": "bevel" effect:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"riserBevel": "bevel",
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

On the output, the risers are not totally flat. They have a slight angle at their edges:



Changing the property to "riserBevel": "cylinder" generates the following chart, in which the risers are drawn as cylinders with gradient color:

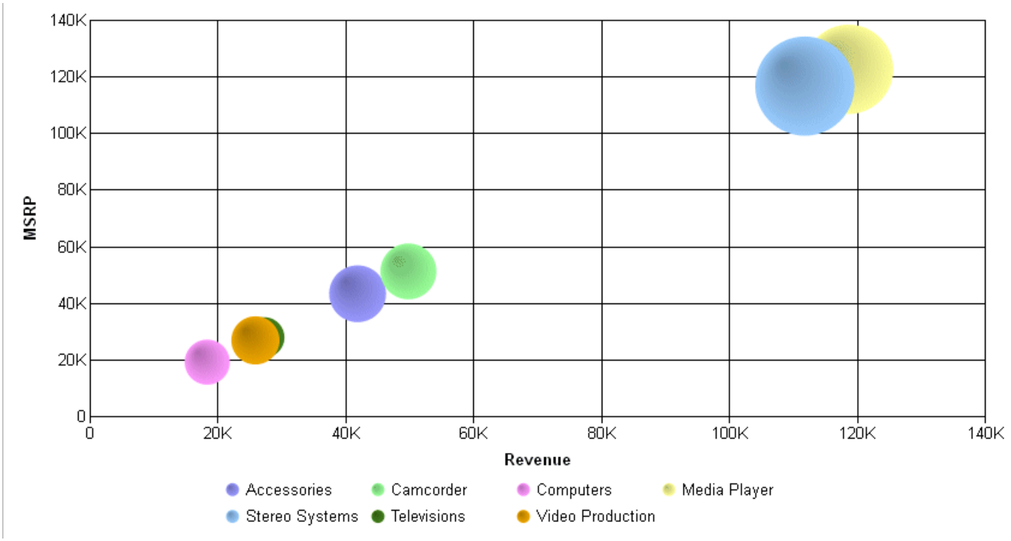


Example: Applying a Bevel Effect to a Bubble Chart

The following request applies the bevel effect "darken" to the bubble markers and legend markers:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 0},
"series": [{"series": "all",    "marker": {"shape": "circle"}}],
"riserBevel": "darken"
*END
ENDSTYLE
END
```

The output is:



Applying a Lightening or Darkening Effect to Successive Risers

If there are more series than series colors defined, the charting engine cycles through the defined colors. The riserCycleEndLightness property defines the lightness or darkness of the final riser color cycle.

```
"riserCycleEndLightness": number
```

where:

`riserCycleEndLightness:` *`number`*

Is a number between 0 and 1 that controls the brightness of colors assigned to undefined series risers. Values less 0.5 darken each series cycle. Values greater than 0.5 lighten each series cycle. The default value is 0.8.

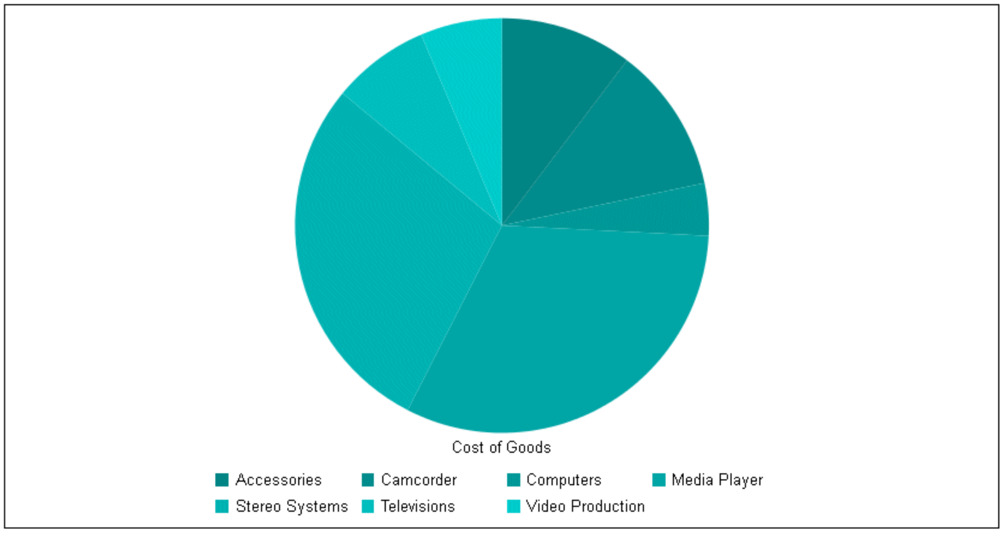
Note: Since WebFOCUS provides default values for the series colors, you must remove these defaults by including {"series": "reset", "color": undefined} in the series array. Then define one color to be the starting color for the cycle.

Example: Applying A Darkening Effect To Successive Pie Slices

The following request starts with the color teal for series 0 and applies a darkening effect to each successive pie slice:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"riserCycleEndLightness": 0.4,
"series": [
    {"series": "reset", "color": undefined},
    {"series": 0, "color": "teal"}
    ]
*END
ENDSTYLE
END
```

The output is:



Controlling Space Between Risers in 3D Charts

In 3D charts and charts where 2.5D depth is applied with the depth property, the riserDepthGap property adds space between risers that draw behind other risers.

Syntax: How to Control Space Between Risers in 3D Charts

```
"riserDepthGap": number
```

where:

`"riserDepthGap": number`

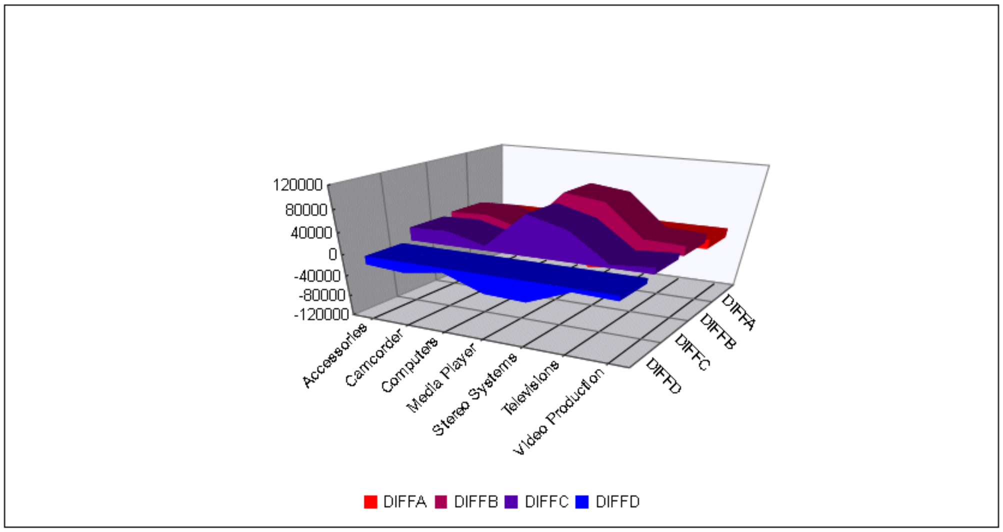
Is a number between 0 and 1 that defines the margin between the risers as a factor of the depth. The default value is 0.2.

Example: Controlling the Space Between Risers in 3D Charts

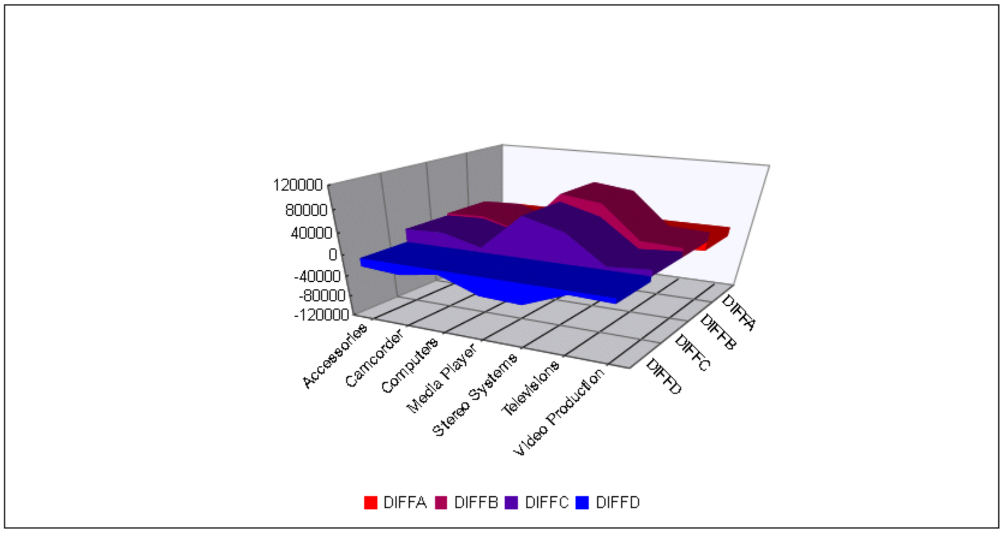
The following request generates a 3D area chart with the default space between the risers:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - (COGS_US * DISCOUNT_US)/100;
DIFFD = COGS_US - MSRP_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DAREAS
ON GRAPH SET STYLE *
*GRAPH_JS
"riserDepthGap": 0.2,
"chartFrame": {
    "border": {"width": 1, "color": "grey"},
    "fill": {"color": "ghostwhite"}},
"colorMode": {
    "mode": "byInterpolation",
    "colorList": ["red", "blue"]
    }
*END
ENDSTYLE
END
```

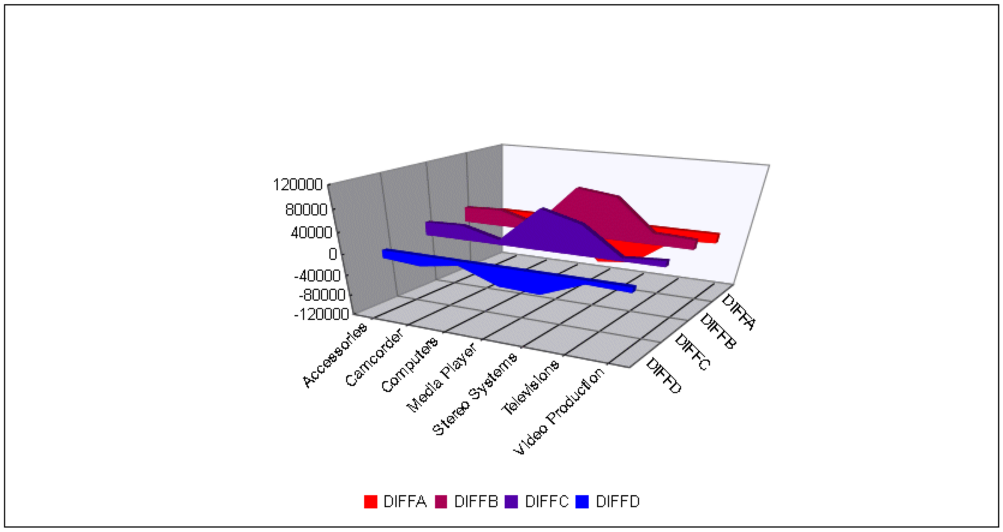
The output is:



Changing the riserDepthGap to zero (0) generates the following chart:



The value 0.9 for riserDepthGap generates the following chart:



Applying a Shadow to Risers and Markers

The riserShadow property applies a shadow to the chart risers and markers.

Syntax: **How to Apply a Shadow to Risers and Markers**

To specify the shadow as a Boolean value, use the following property:

```
"riserShadow": boolean
```

where:

`"riserShadow": boolean`

Can be:

true, to enable a default shadow.

false, to disable a default shadow. This is the default value.

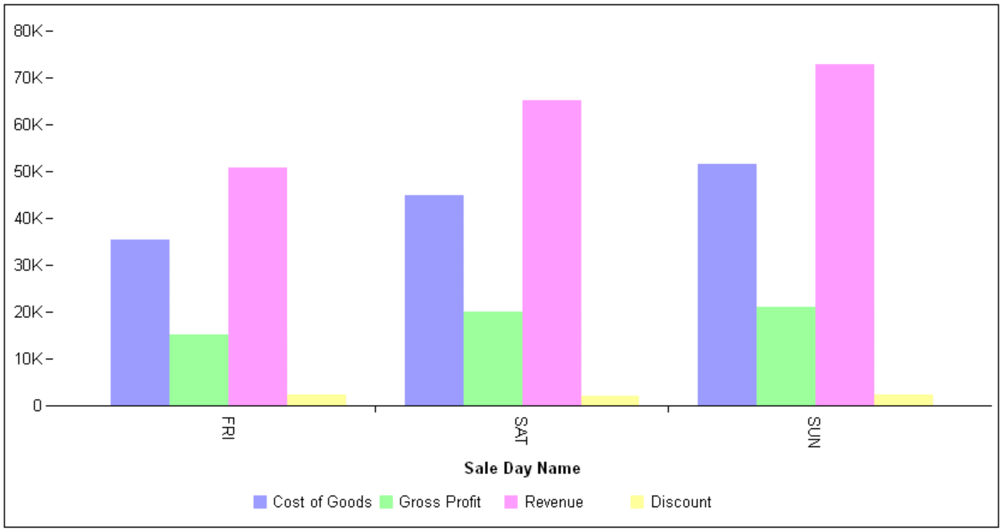
Note: Other chart objects, such as the chart frame and legend, can be assigned shadows. For more information, see *Formatting the Chart Frame* on page 231 and *Applying a Shadow to the Legend Area* on page 306.

Example: Applying a Shadow to Risers and Markers

The following request does not enable riser shadows:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
END
```

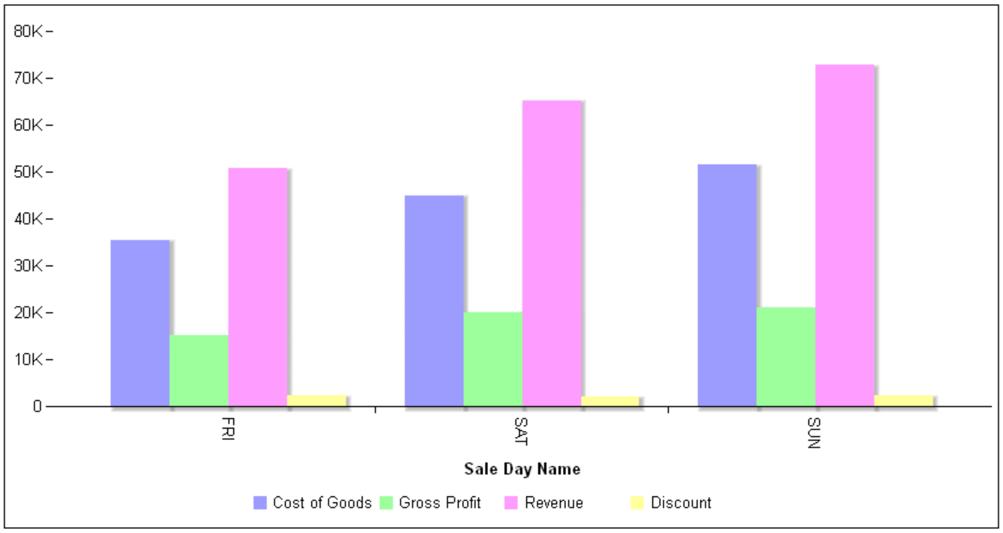
On the output, the risers have no shadows:



The following version of the request enables the default shadow by including the riserShadow: true property:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
   "riserShadow": true
*END
ENDSTYLE
END
```

On the output, the risers have a slight shadow to the right:



Swapping Series and Group Orientation

The swapData property can be used to swap series and group orientation. When false (the default), values in a row in a data set are represented as series. When true, values in a column are represented as series.

Syntax: How to Swap Series and Group Orientation

```
"swapData": boolean
```

where:

`"swapData": `*`boolean`*

Can be:

- true, which swaps the series and group orientation.
- false, which does not swap the series and group orientation. This is the default value.

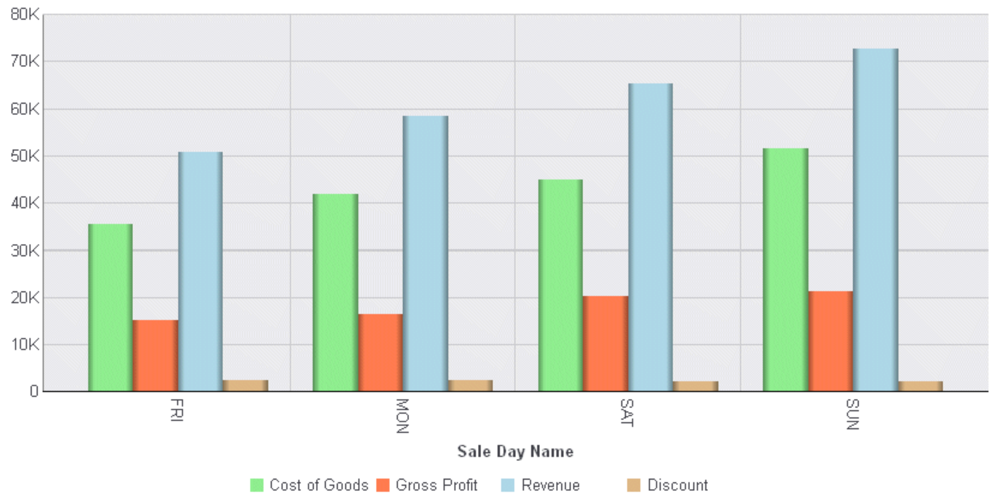
Example: Swapping Series and Group Orientation

The following request generates a vertical bar chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN' OR 'MON'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"swapData": false,
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

On the output:

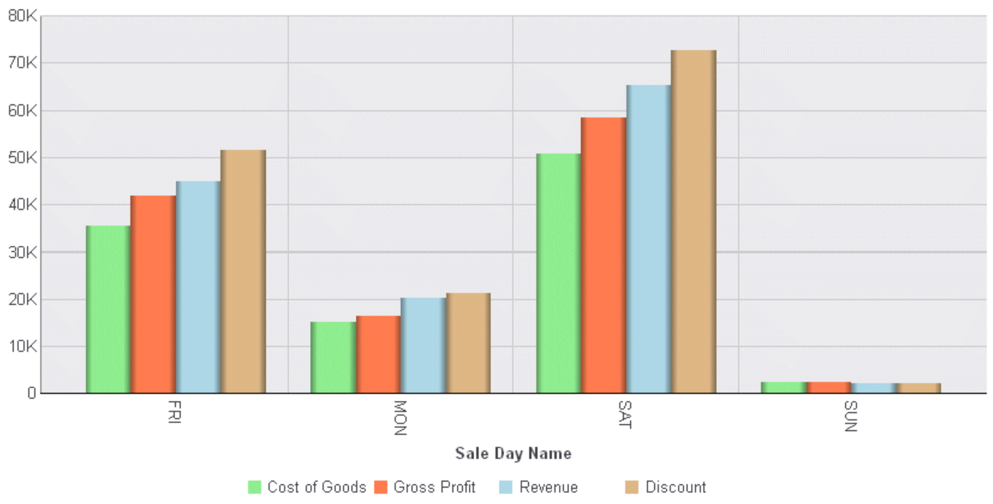
- The first set of bars represents all series (COGS_US, GROSS_PROFIT_US, REVENUE_US, and DISCOUNT_US) values for the first group value (day name FRI).
- The second set of bars represents all series values for the second group value (day name SAT).
- The third set of bars represents all series values for the third group value (day name SUN).
- The fourth set of bars represents all series values for the third group value (day name MON).



Changing the swapData property to *"true"* generates the following chart, in which:

- ❑ The first set of bars represents COGS_US (series 0) for each day name (group) value.
- ❑ The second set of bars represents GROSS_PROFIT_US (series 1) for each day name (group) value.
- ❑ The third set of bars represents REVENUE_US (series 2) for each day name (group) value.
- ❑ The fourth set of bars represents DISCOUNT_US (series 3) for each day name (group) value.

Although the labels did not change, the series and groups have been swapped:



To swap the labels as well as the series and group orientation, see *Swapping Series, Group, and Label Orientation* on page 258.

Swapping Series, Group, and Label Orientation

The swapDataAndLabels property can be used to swap series and group orientation, and to swap the labels as well. When false (the default), values in a row in a data set are represented as series. It is the same as the swapData property, except it also swaps the series and group labels. When true, values in a column are represented as series.

Syntax: How to Swap Series and Group Orientation and Their Labels

```
"swapDataAndLabels": boolean
```

where:

`"swapDataAndLabels":` *`boolean`*

Can be:

true, which swaps the series, group, and label orientation.

false, which does not swap the series, group, and label orientation. This is the default value.

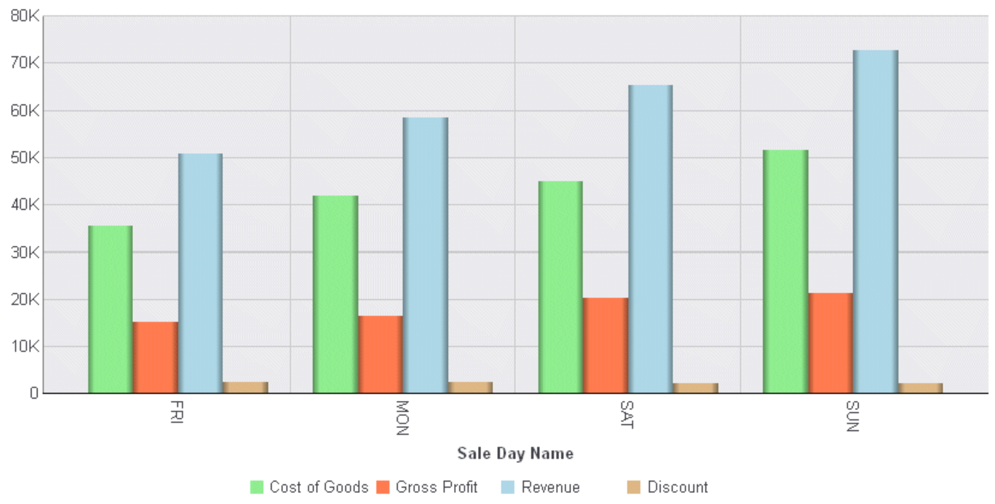
Example: Swapping Series, Group, and Label Orientation

The following request generates a vertical bar chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS TIME_DAYNAME
WHERE TIME_DAYNAME EQ 'FRI' OR 'SAT' OR 'SUN' OR 'MON'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"swapDataAndLabels": false,
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

On the output:

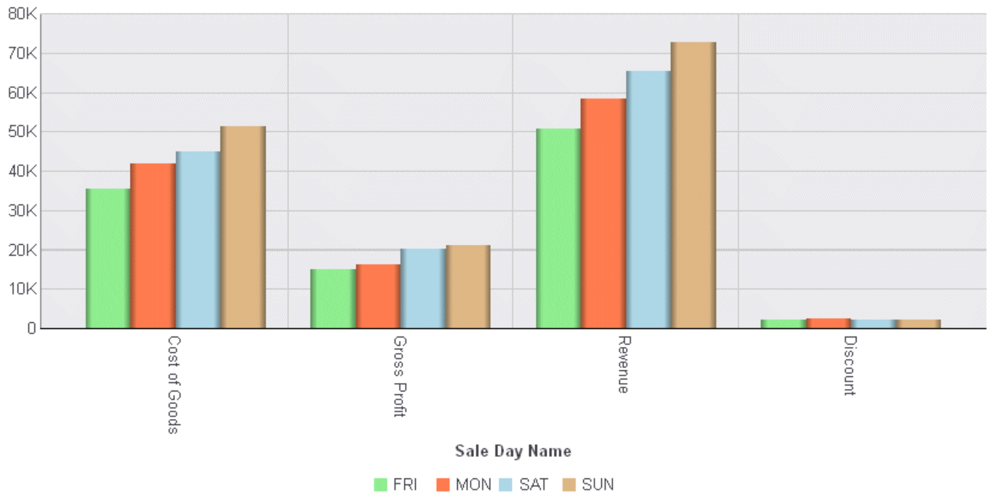
- The first set of bars represents all series (COGS_US, GROSS_PROFIT_US, REVENUE_US, and DISCOUNT_US) values for the first group value (day name FRI).
- The second set of bars represents all series values for the second group value (day name SAT).
- The third set of bars represents all series values for the third group value (day name SUN).
- The fourth set of bars represents all series values for the third group value (day name MON).



Changing the swapDataAndLabels property to *"true"* generates the following chart, in which:

- The first set of bars represents COGS_US (series 0) for each day name (group) value.
- The second set of bars represents GROSS_PROFIT_US (series 1) for each day name (group) value.
- The third set of bars represents REVENUE_US (series 2) for each day name (group) value.
- The fourth set of bars represents DISCOUNT_US (series 3) for each day name (group) value.

The labels have also been swapped, to represent the new orientation:



Chapter 6

Chart Title Properties

This chapter describes how to control the format of the chart title, subtitle, and footnote.

In this chapter:

Chart Title Properties Overview

These properties control the visibility and format of chart titles (Title, Subtitle, and Footnote). The following code shows the property names and default values:

```
"title": {
   "text": "Chart Title",
   "visible": false,
   "align": "center",  // One of "left", "center", "right", "chartFrame"
   "font": "bold 10pt Sans-Serif",
   "color": "black",

},


"subtitle": {
   "text": "Chart Subtitle",
   "visible": false,
   "align": "center",  // One of "left", "center", "right", "chartFrame"
   "font": "10pt Sans-Serif",
   "color": "black",

},


"footnote": {
   "text": "Chart Footnote",
   "visible": false,
   "align": "center",  // One of "left", "center", "right", "chartFrame"
   "font": "10pt Sans-Serif",
   "color": "black",

 },
```

Each axis also has a title object. For information, see *Axis Properties* on page 325.

Formatting the Chart Footnote

The footnote properties control the content, visibility, and format of the chart footnote.

Syntax: How to Format the Chart Footnote

```
"footnote": {
   "text": "string",
   "visible": boolean,
   "align": "string",
   "font": "string",
   "color": "string"

}
```

where:

`"text": "string"`

Is a string that defines the footnote text. The default value is "Chart Footnote".

`"visible": boolean`

Controls the visibility of the chart footnote. Valid values are:

- ❑ true, which makes the chart footnote visible.
- ❑ false, which makes the chart footnote not visible. This is the default value.

`"align": "string"`

Is a string that defines the alignment of the footnote, Valid values are:

- ❑ "center", which centers the footnote in the draw area. This is the default value.
- ❑ "chartFrame", which centers the footnote aligned with the chart frame.
- ❑ "left", which left-justifies the chart footnote in the draw area.
- ❑ "right", which right-justifies the chart footnote in the draw area.

`"font": "string"`

Is a string that defines the size, style, and, typeface of the chart footnote. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the chart footnote using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"backgroundColor": "`*`string`*`"`

Is a color specification string or gradient definition that defines the background fill color of the chart title object. The default value is *undefined*.

`"border"`

Defines the properties of the border around the chart title object.

`"width":` *`number`*

Defines the width of the border in pixels. The default value is 0 (zero).

`"color": "`*`string`*`"`

Is a color specification string that defines the color of the border around the chart title object. The default value is *undefined*.

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border around the chart title object. Enter the length of the dash in pixels followed by the length of the space between dashes in pixels. The default value is no dash (" ").

`"cornerRadius": {"x":` *`value`*`, "y":` *`value`*`}`

Defines the properties of the corners of the box around the title object.

The cornerRadius property can consist of one value or two values. When there is one value that is zero (0), the corners are square. This is the default corner shape. Rounded corners can be circular or elliptical. The x and y values denote the size of the circle radius or the semi-major and semi-minor axes of the ellipse.

Valid values for x and y are:

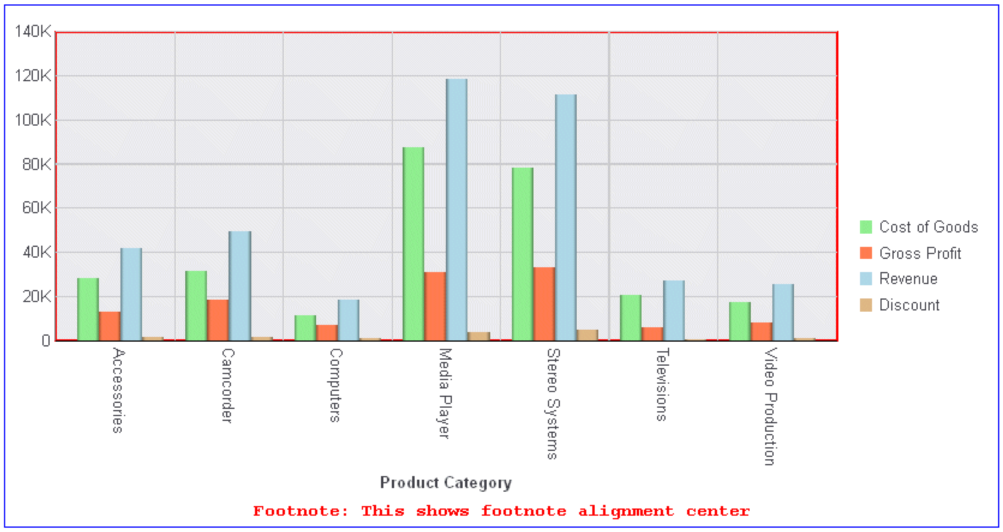
- A number that specifies the radius of the corner in pixels.
- A percent string such as "20%", that represents a percentage of the height (y) or width (x) of the box around the title object.
- A CSS length string such as "5em" or "10px" that specifies the radius of the corner in pixels. For information about CSS length strings, see *https://developer.mozilla.org/en-US/docs/Web/CSS/length*.
- An object with "x" and "y" properties (where "x" and "y" can be any of the above) to specify the corner radius along the width (x) and height (y) of the title object box.

Example: Formatting the Chart Footnote

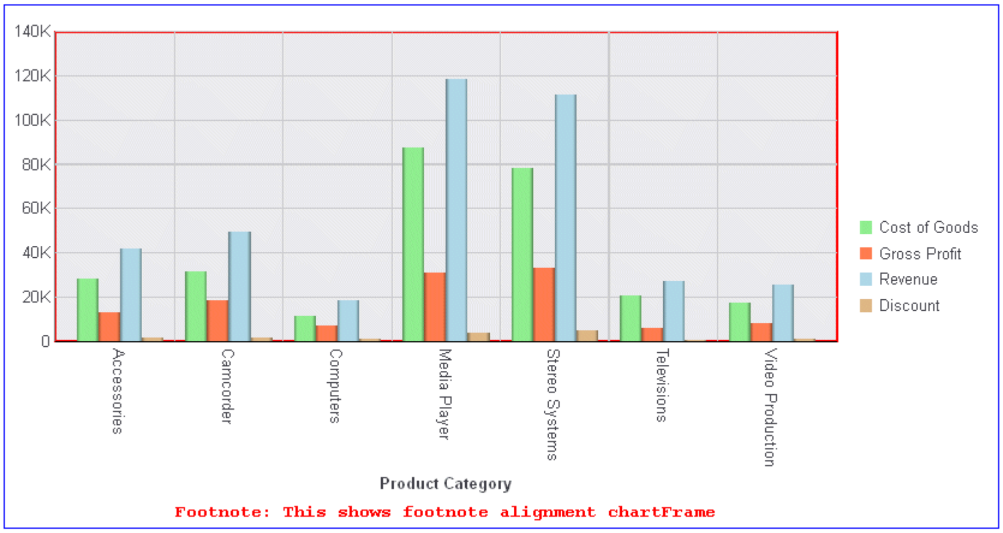
The following request generates a vertical bar chart with a footnote. The alignment of the footnote, and part of the text, depends on the Dialogue Manager variable named &FOOTALIGN, which is set to the value 'center':

```
-SET &FOOTALIGN = 'center';
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"color": "blue"},
"chartFrame": {"border": {
    "width": 2, "color": "red"}},
"legend": {"position": "right"},
"footnote": {
    "text": "Footnote: This shows footnote alignment &FOOTALIGN",
    "font": "Bold 10pt Courier", "visible": true,
    "align": "&FOOTALIGN","color": "red"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

On the output, with alignment 'center', the footnote is centered in the draw area:



Changing the value of &FOOTALIGN to 'chartFrame' generates a footnote centered aligned with the chart frame:



Formatting the Chart Subtitle

The subtitle properties control the content, visibility, and format of the Chart Subtitle.

Syntax: How to Format the Chart Subtitle

```
"subtitle": {
   "text": "string",
   "visible": boolean,
   "align": "string",
   "font": "string",
   "color": "string"

   }
```

where:

`"text": "string"`

Is a string that defines the subtitle text. The default value is "Chart Subtitle".

`"visible": boolean`

Controls the visibility of the chart subtitle. Valid values are:

- true, which makes the chart subtitle visible.
- false, which makes the chart subtitle not visible. This is the default value.

`"align": "string"`

Is a string that defines the alignment of the subtitle, Valid values are:

- "center", which centers the subtitle in the draw area. This is the default value.
- "chartFrame", which centers the subtitle aligned with the chart frame.
- "left", which left-justifies the chart subtitle in the draw area.
- "right", which right-justifies the chart subtitle in the draw area.

`"font": "string"`

Is a string that defines the size, style, and, typeface of the chart subtitle. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the chart subtitle using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"backgroundColor": "string"`

Is a color specification string or gradient definition that defines the background fill color of the chart title object. The default value is *undefined*.

`"border"`

Defines the properties of the border around the chart title object.

`"width": number`
Defines the width of the border in pixels. The default value is 0 (zero).

`"color": "string"`
Is a color specification string that defines the color of the border around the chart title object. The default value is *undefined*.

`"dash": "string"`
Is a string that defines the dash style of the border around the chart title object. Enter the length of the dash in pixels followed by the length of the space between dashes in pixels. The default value is no dash (" ").

`"cornerRadius": {"x": value, "y": value}`
Defines the properties of the corners of the box around the title object.

The cornerRadius property can consist of one value or two values. When there is one value that is zero (0), the corners are square. This is the default corner shape. Rounded corners can be circular or elliptical. The x and y values denote the size of the circle radius or the semi-major and semi-minor axes of the ellipse.

Valid values for x and y are:

❑ A number that specifies the radius of the corner in pixels.

❑ A percent string such as "20%", that represents a percentage of the height (y) or width (x) of the box around the title object.

❑ A CSS length string such as "5em" or "10px" that specifies the radius of the corner in pixels. For information about CSS length strings, see *https://developer.mozilla.org/en-US/docs/Web/CSS/length*.

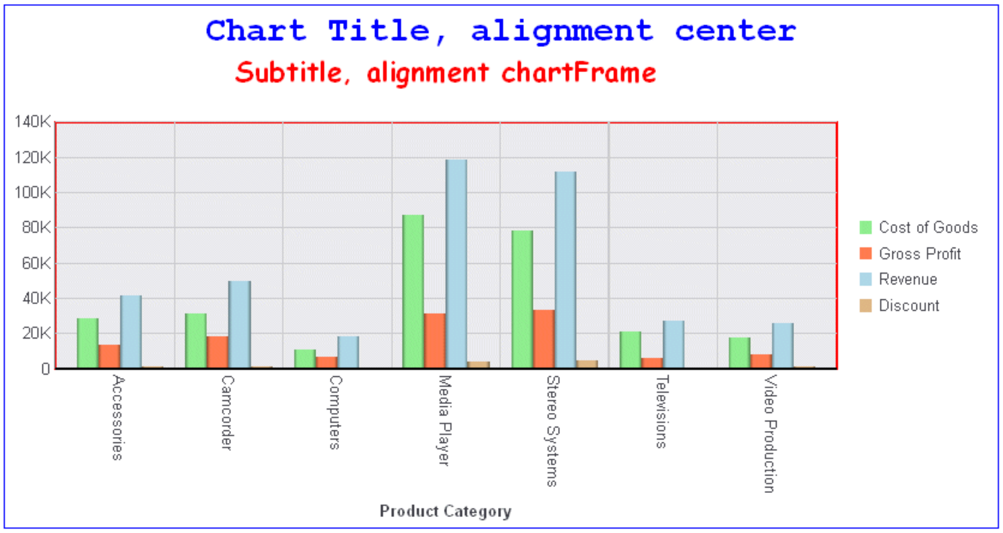
❑ An object with "x" and "y" properties (where "x" and "y" can be any of the above) to specify the corner radius along the width (x) and height (y) of the title object box.

Example: **Formatting the Chart Subtitle**

The following request generates a vertical bar chart with a subtitle. The alignment of the subtitle, and part of the text, depends on the Dialogue Manager variable named &SUBTALIGN, which is set to the value 'chartFrame':

```
-SET &SUBALIGN = 'chartFrame';
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"color": "blue"},
"chartFrame": {"border": {
    "width": 2, "color": "red"}},
"legend": {"position": "right"},
"title": {
    "text": "Chart Title, alignment center",
    "visible": true,
    "font": "Bold 20pt Courier",
    "align": "center", "color": "blue"},
"subtitle": {
    "text": "Subtitle, alignment &SUBALIGN",
    "font": "Bold 16pt Comic Sans MS", "visible": true,
    "align": "&SUBALIGN", "color": "red"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}]
*END
ENDSTYLE
END
```

On the output, the title is centered in the draw area (align: "center"), and the subtitle is centered aligned with the chart frame (align: "chartFrame"):



Formatting the Chart Title

The title properties control the content, visibility, and format of the Chart Title.

Note: A chart title is not supported for sparkline charts and cannot be realigned in bullet charts.

Syntax: How to Format the Chart Title

```
"title": {
   "text": "string",
   "visible": boolean,
   "align": "string",
   "font": "string",
   "color": "string"

  }
```

where:

`"text": "string"`

Is a string that defines the title text. The default value is "Chart Title".

`"visible": boolean`

Controls the visibility of the chart title. Valid values are:

- ❏ true, which makes the chart title visible.
- ❏ false, which makes the chart title not visible. This is the default value.

`"align": "string"`

Is a string that defines the alignment of the title, Valid values are:

- ❏ "center", which centers the title in the draw area. This is the default value.
- ❏ "chartFrame", which centers the title aligned with the chart frame.
- ❏ "left", which left-justifies the chart title in the draw area.
- ❏ "right", which right-justifies the chart title in the draw area.

`font: 'string'`

Is a string that defines the size, style, and, typeface of the chart title. The default value is "10pt Sans-Serif".

`color: 'string'`

Is a string that defines the color of the chart title using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"backgroundColor": "string"`

Is a color specification string or gradient definition that defines the background fill color of the chart title object. The default value is *undefined*.

`"border"`

Defines the properties of the border around the chart title object.

`"width": number`

Defines the width of the border in pixels. The default value is 0 (zero).

`"color": "string"`

Is a color specification string that defines the color of the border around the chart title object. The default value is *undefined*.

`"dash": "string"`

Is a string that defines the dash style of the border around the chart title object. Enter the length of the dash in pixels followed by the length of the space between dashes in pixels. The default value is no dash (" ").

`"cornerRadius": {"x": *value*, "y": *value*}`

Defines the properties of the corners of the box around the title object.

The cornerRadius property can consist of one value or two values. When there is one value that is zero (0), the corners are square. This is the default corner shape. Rounded corners can be circular or elliptical. The x and y values denote the size of the circle radius or the semi-major and semi-minor axes of the ellipse.

Valid values for x and y are:

❑ A number that specifies the radius of the corner in pixels.

❑ A percent string such as "20%", that represents a percentage of the height (y) or width (x) of the box around the title object.

❑ A CSS length string such as "5em" or "10px" that specifies the radius of the corner in pixels. For information about CSS length strings, see *https://developer.mozilla.org/en-US/docs/Web/CSS/length*.

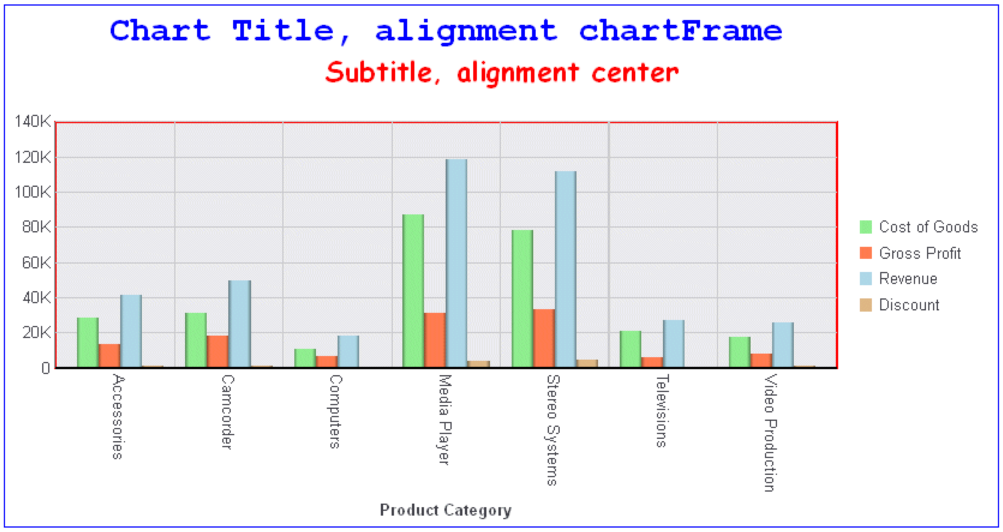
❑ An object with "x" and "y" properties (where "x" and "y" can be any of the above) to specify the corner radius along the width (x) and height (y) of the title object box.

Example: Formatting the Chart Title

The following request generates a vertical bar chart with a title and subtitle. The alignment of the title is "chartFrame", and the alignment of the subtitle is "center":

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
border:{color:'blue'},
chartFrame: {border: {width:2, color:'red'}},
legend: {position:'right'},
 "border": {"color": "blue"},
"chartFrame": {"border": {
    "width": 2, "color": "red"}},
"legend": {"position": "right"},
"title": {
    "text": "Chart Title, alignment chartFrame",
    "visible": true, "font": "Bold 20pt Courier",
    "align": "chartFrame", "color": "blue"},
"subtitle": {
    "text": "Subtitle, alignment center",
    "font": "Bold 16pt Comic Sans MS", "visible": true,
    "align": "center", "color": "red"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

On the output, the title is centered aligned with the chart frame (align: 'chartFrame'), and the subtitle is centered in the draw area (align: 'center'):

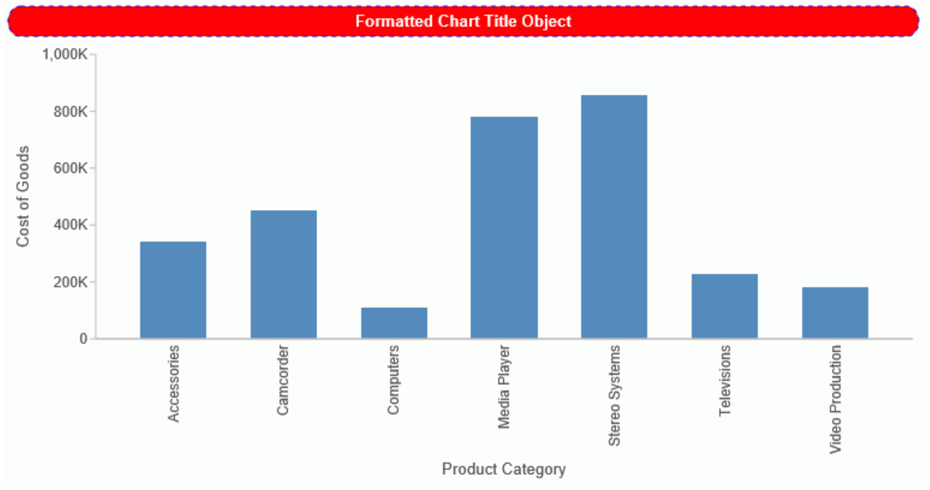


Example: Formatting the Background Fill and Border Properties of the Chart Title

The following request generates a title box with red background fill, a blue dashed border, and rounded corners.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET STYLE *

INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$

TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $

*GRAPH_JS
"title": {
      "visible": true,
      "text": "Formatted Chart Title Object",
      "color": "white",
      "backgroundColor": "red",
      "border":
              {
               "width": 1,
               "color": "blue",
               "dash": "4 4",
               "cornerRadius": {"x": 10, "y":"25%"}
               }
          }
*END
ENDSTYLE
END
```

The output is shown in the following image.

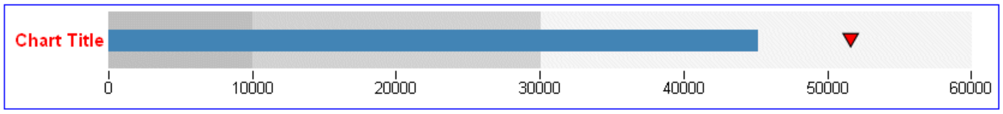


Example: Formatting a Bullet Chart Title

The following request generates a bullet chart with a chart title.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY TIME_DAYOFWEEK
WHERE TIME_DAYNAME EQ 'SAT' OR 'SUN'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET VAXIS 80
ON GRAPH SET LOOKGRAPH CUSTOM
ON GRAPH SET STYLE *
*GRAPH_JS
"chartType": "bullet",
"border": {"width": 1, "color": "blue"},
"title": {"visible": true, "text": "Chart Title", "color": "red"},
"yaxis": {"colorBands": [
    {"start": 0, "stop": 10000, "color": "silver"},
    {"start": 10000, "stop": 30000, "color": "lightgrey"},
    {"start": 30000, "stop": 60000, "color": "whitesmoke"}]},
"series": [
    {"series": 0, "group": 0, "color": "steelblue"},
    {"series": 0, "group": 1, "color": "red", "marker":{"shape":"triangle"}}
    ]
*END
ENDSTYLE
END
```

The output is:



Chapter 7

Legend Properties

This chapter describes how to control the format of the legend area.

In this chapter:

Legend Properties Overview

The legend properties control the visibility and format of the legend area.

The following code segment shows the default values:

```
"legend": {
    "visible": true,
    "position": "right",
  // {
  //    "side": undefined,
  //    "anchor": "chartFrame",
  //    "excludeFromLayout": undefined
  //    "left": undefined,
  //    "right": undefined
  //    "top": undefined,
  //    "bottom": undefined,
  //    "align": undefined
  // },
 "maxSize": '50%',
 "orientation": "auto",
 "reverseOrder": false,
 "markerSize": 8,
 "markerPosition": "left",
 "matchSeriesMarkers": false,
 "maxEntries": "undefined",
 "title": {
          "visible": false,
          "text": "Legend Title",
          "font": "10pt Sans-Serif",
          "color": "black",
    "tooltip": undefined,
    "wrap": 1},
 "labels": {
          "font": "7.5pt Sans-Serif",
          "color": "black",
    "tooltip": undefined,
    "wrap": 1},
 "lineStyle": {
          "width": 0,
          "color": "black",
          "dash": ""},
 "backgroundcolor": "transparent",
 "shadow": false,

    "scroll": {
        "enabled": false,
        "size": 15,
        "color": "rgb(240, 240, 240)",
        "handle": {
            "color": "grey",
            "hoverColor": "rgb(88, 88, 88)",
            "border": {
                "width": 0,
                "color": "transparent",
                "dash": ""
        }}},
    "dock": {
        "enabled": false,
        "animate": true,
        "minimized": false,
        "resizeFrame": true,
        "showMarkers": true,
        "expandDirection": "undefined",
        "button": {
            "size": "auto",
            "color": "grey",
            "hoverColor": "black",
            "border": {
                "width": 0,
                "color": "transparent",
                "dash": ""
        }}},

"sizeLegend":
 {
    "layout": "circle",
    "lineStyle":
      {
      "width": 0,
      "color": "black",
      "dash": ""
      }
  }

}
```

Defining the Background Color of the Legend Area

The backgroundcolor property defines the background color of the legend area.

Syntax: How to Define the Background Color of the Legend Area

```
"legend": {
   "backgroundcolor": "color"}
```

where:

`"backgroundcolor": "`*`color`*`"`

Can be:

- ❑ A string, enclosed in single quotation marks, that defines a color (by name or numeric specification string) or a gradient. The default value is "transparent".
- ❑ A JSON object that defines a gradient.

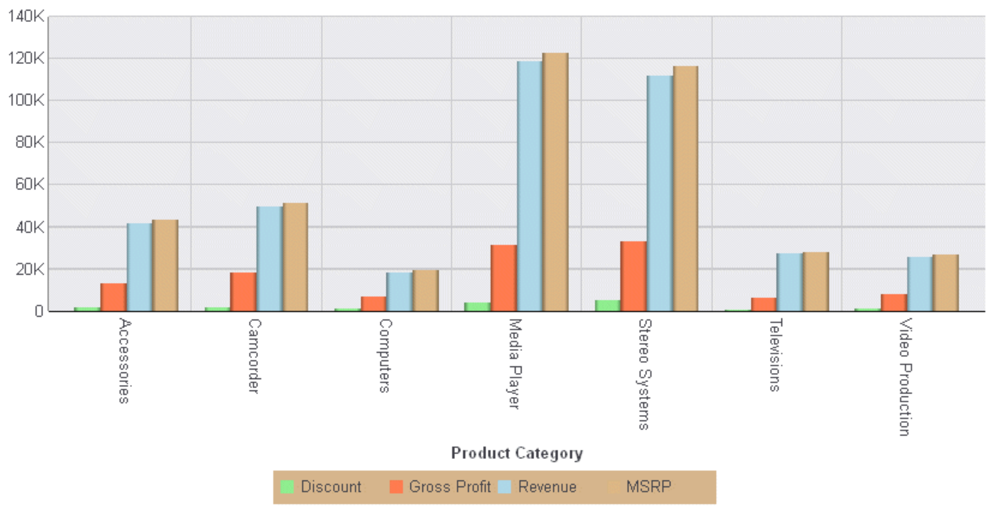
For information about defining colors and gradients, see *Colors and Gradients* on page 85.

Example: Defining the Background Color of the Legend Area

The following request makes the background color of the legend area tan:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"backgroundcolor": "tan"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]

*END
ENDSTYLE
END
```

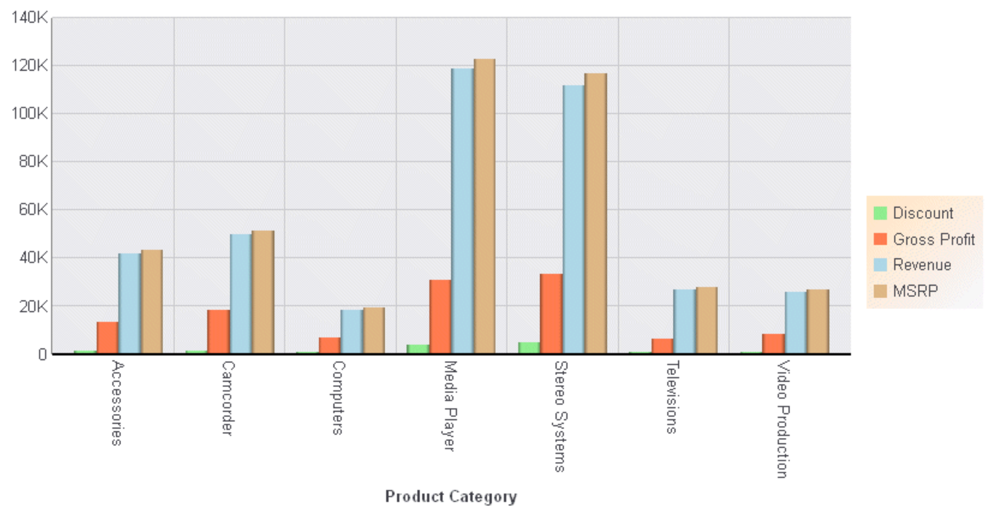
The output is:



The following request applies a linear gradient that transitions from bisque to ghostwhite to the legend background area:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {
    "position": "right",
    "backgroundcolor": {
        "type": "linear",
        "start": {"x": "0%", "y": "0%"},
        "end": {"x": "100%", "y": "100%"},
        "stops": [[0, "bisque"], [1, "ghostwhite"]],
        "lineStyle": {"width": 2, "color": "navy"}}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Formatting the Legend Labels

The labels property controls the format of legend labels.

Syntax: **How to Format the Legend Labels**

```
"legend":
 {
   "labels":
    {
      "font": "string",
      "color": "string"
      "tooltip": "string",
      "wrap": number
    }
 }
```

where:

`"font": "string"`

Is a string that defines the size, style, and typeface of the legend labels. The default value is "7.5pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the legend labels using a color name or numeric specification string. The default value is "black".

For information about defining colors, see *Colors and Gradients* on page 85.

`"tooltip": "string"`

Is a string, or a function that returns a string, to display when the mouse hovers over the label.

`"wrap": number`

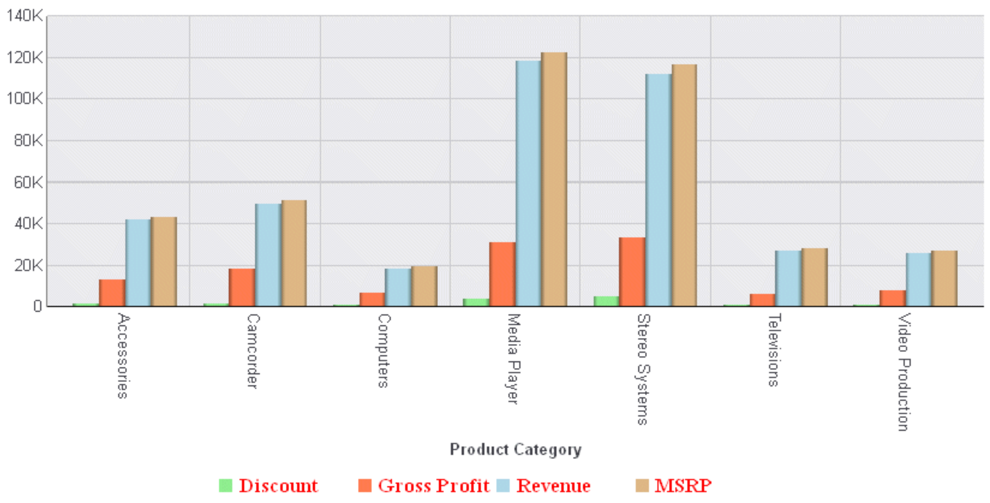
Defines the number of lines for wrapping long labels. Can be an integer, which wraps long labels to that number of lines (if necessary), or "auto", which wraps long labels into as many lines as necessary. The default value is 1.

Example: Formatting Legend Labels

The following request makes the legend labels red, bold, 12pt Times Roman:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"labels": {
    "font": "Bold 12pt Times Roman",
    "color": "red"
    }},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Generating a Line Around the Legend Area

The lineStyle property controls the width and color of a line that can be drawn around the legend area.

Syntax: How to Generate a Line Around the Legend Area

```
"legend": {
            "lineStyle": {
      "width": number,
      "color": "color",
      "dash": "string"'
   }
}
```

where:

`"width": number,`

Is a number that defines the width of the line in pixels. The default value is 0, no line.

`"color": "color"`

Defines the color of the line as either:

- A string, enclosed in single quotation marks, that defines a color by a name or numeric specification string or defines a gradient string. The default value is "black".
- A JSON object that defines a gradient.

`"dash": "string"`

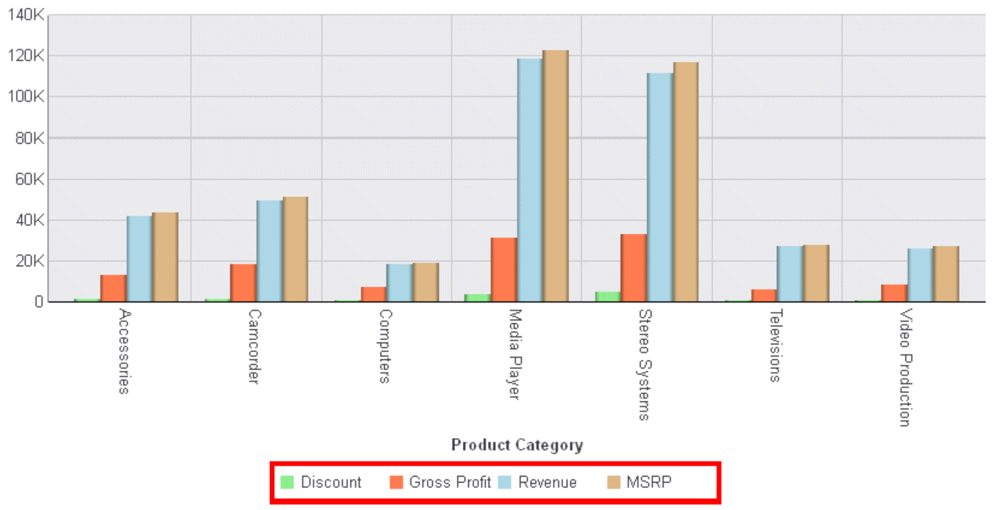
Is a string that defines the dash style of the line. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Example: Generating a Line Around the Legend Area

The following request generates a four-pixel wide red line around the legend area:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"lineStyle": {
    "width": 4,
    "color": "red"}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Controlling the Location of the Legend Markers

The markerPosition property controls the location of the legend markers relative to the legend labels.

Syntax: How to Control the Position of the Legend Markers

```
"legend": {"markerPosition": "string"}
```

where:

`"markerPosition": "string"`

Is a string that defines the location of legend markers relative to the legend label. Valid values are:

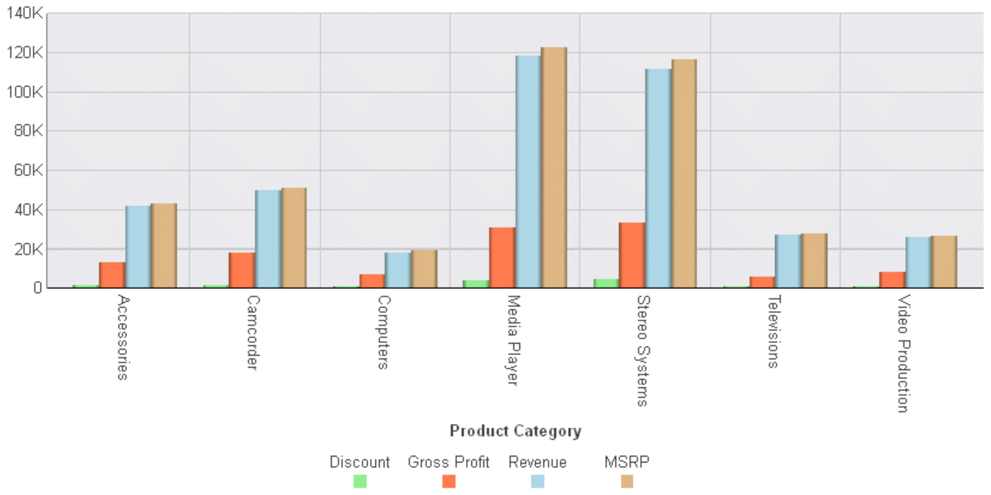
- "bottom"
- "left". This is the default value.
- "right"
- "top"

Example: Controlling the Position of the Legend Markers

The following request places the legend markers below the legend labels:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"markerPosition": "bottom"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

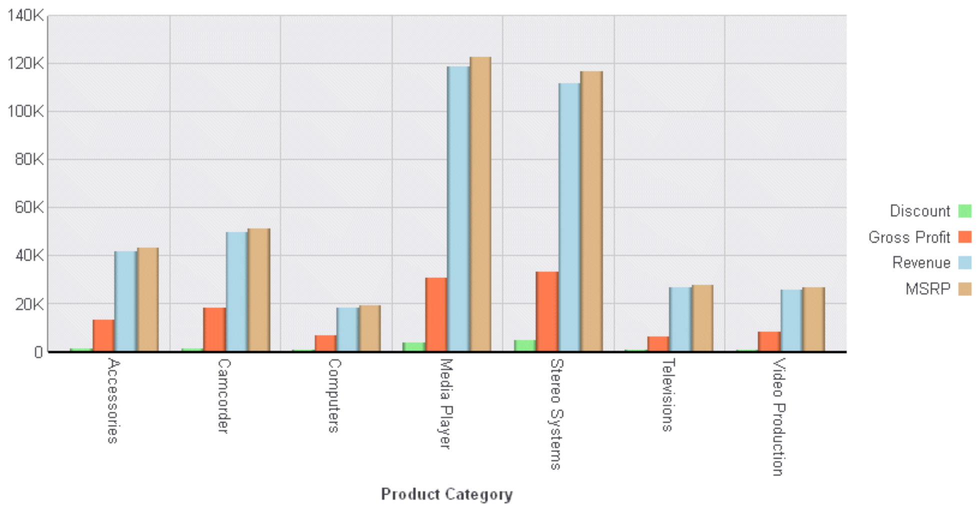
The output is:



The following version of the request moves the legend to the right of the chart and places the legend markers to the right of the legend labels:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {
    "position": "right",
    "markerPosition": "right"
    },
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

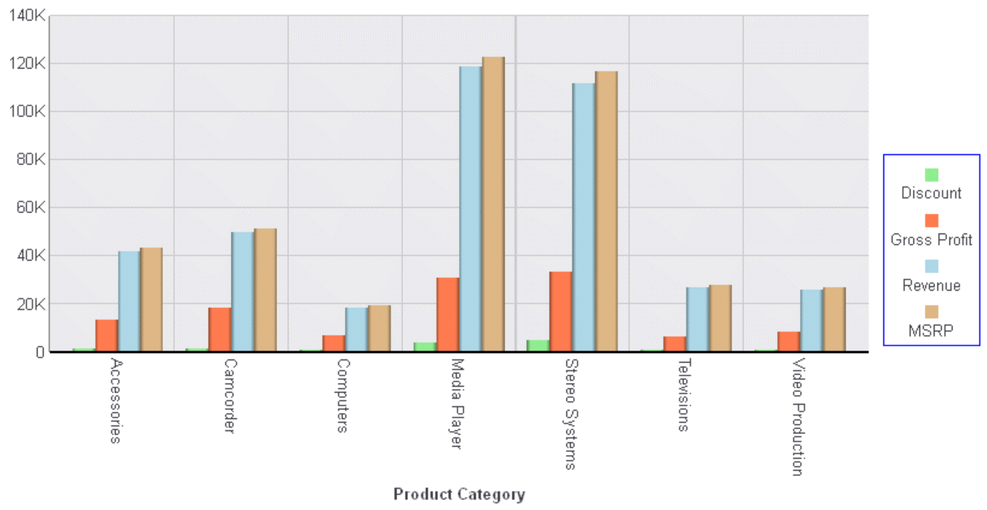
The output is:



The following legend properties change the marker position to 'top' with the legend position to the right. They also draw a border around the legend:

```
"legend": {
    "position": "right",
    "markerPosition": "top",
    "lineStyle": {
        "color": "blue"}}
```

These legend properties generate the following chart:



Controlling the Size of Legend Markers

The markerSize property controls the size of legend markers.

Syntax: How to Control the Size of Legend Markers

```
"legend": {"markerSize": number}
```

where:

`"markerSize": number`

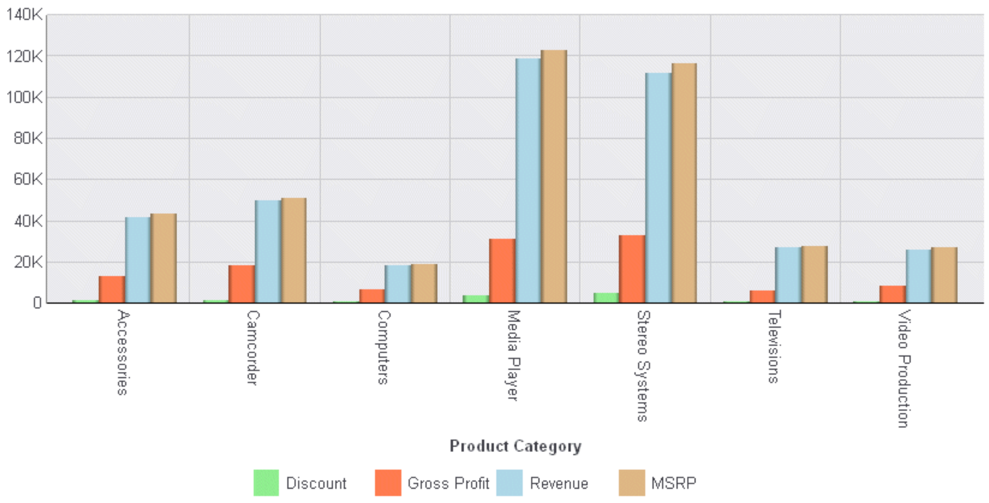
Defines the size of legend markers in pixels. The default value is 8.

Example: Controlling the Size of Legend Markers

The following request increases the size of the legend markers to 20:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"markerSize": 20},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Matching Legend Markers to Series Markers for Lines

You can use the legend:matchSeriesMarkers property to make the legend markers for line series accurately reflect the look of the line in a line chart or combo chart. When this property is *true*, legend entries for lines include a line, and will only include a marker if markers are visible on that line. If the line has markers, the legend has the same markers. If the line has no markers, the legend has no markers.

Syntax: How to Match Legend Markers to Series Markers

```
"legend":{"matchSeriesMarkers":boolean}
```

where:

`"matchSeriesMarkers":`*`boolean`*

Controls the look of the legend markers for line series. Valid values are:

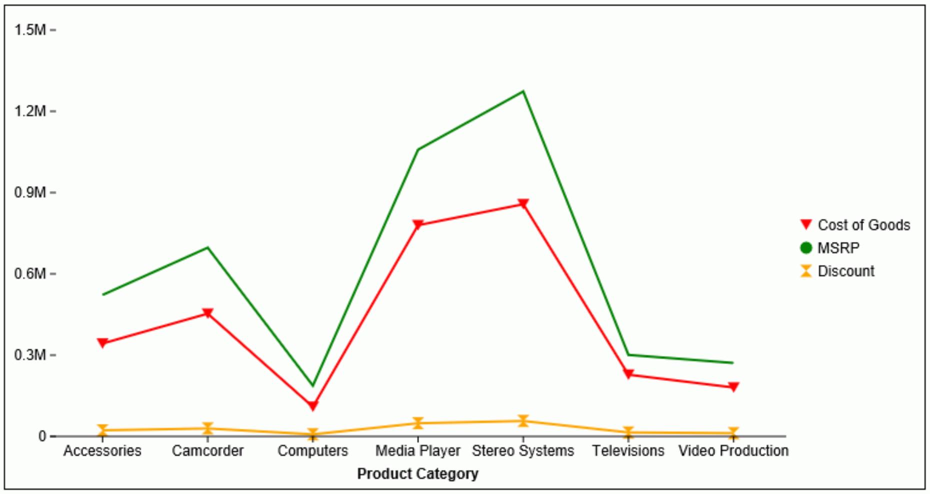
- ❏ true, which generates legend markers that have a line. If the line on the chart has a marker, the line in the legend has the same marker. If the line in the chart has no marker, the line in the legend has no marker.
- ❏ <u>false</u>, which generates a legend that has no lines, only markers. If the line in the chart has a marker, the legend has the same marker. If the line in the chart has no marker, the legend has a default marker. This is the default value.

Example: Matching Legend Markers to Series Markers for Lines

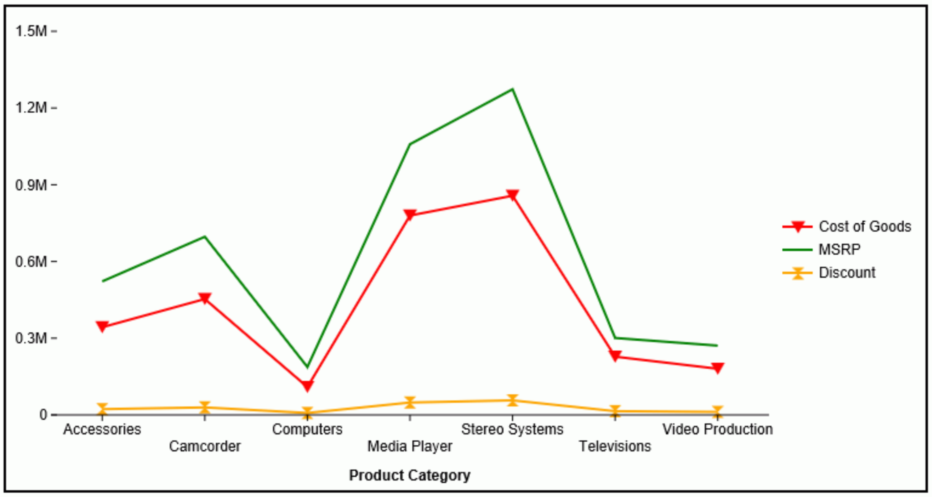
In the following request, the legend property matchSeriesMarkers is set to false, the default:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US  DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis,$
TYPE=DATA, COLUMN=MSRP_US , BUCKET=y-axis,$
TYPE=DATA, COLUMN=DISCOUNT_US, BUCKET=y-axis,$
*GRAPH_JS
"legend": {
           "matchSeriesMarkers": false
          },
"yaxis":{"majorGrid":{"visible":false}},
"xaxis":{"majorGrid":{"visible":false}},
"series":[
          {"series":0, "marker":{"shape":"triangle", "size":10}},
          {"series":1,  "marker":{"visible": false}},
          {"series":2, "marker":{"shape":"hourglass", "size":10}}
          ]

*END
ENDSTYLE
END
```

The output is shown in the following image. Series 1 (MSRP_US) does not have markers in the chart, so the legend marker for series 1 is a default marker. The legend markers for series 0 and 2 are the same markers as those for the line series in the chart.



Changing matchSeriesMarkers to true generates the output shown in the following image. Series 1 (MSRP_US) does not have markers in the chart, so the legend marker for series 1 is a line with no markers. The legend markers for series 0 and 2 are lines with the same markers as the line series in the chart.



Setting the Maximum Percent of Space for the Legend

You can use the legend:maxSize property to set the maximum percent of the draw area that can be used for the legend.

Syntax: How to Set the Maximum Percent of Space for the Legend

```
"legend":
{
 "maxSize": "string"

}
```

where:

`"maxSize": "string"`

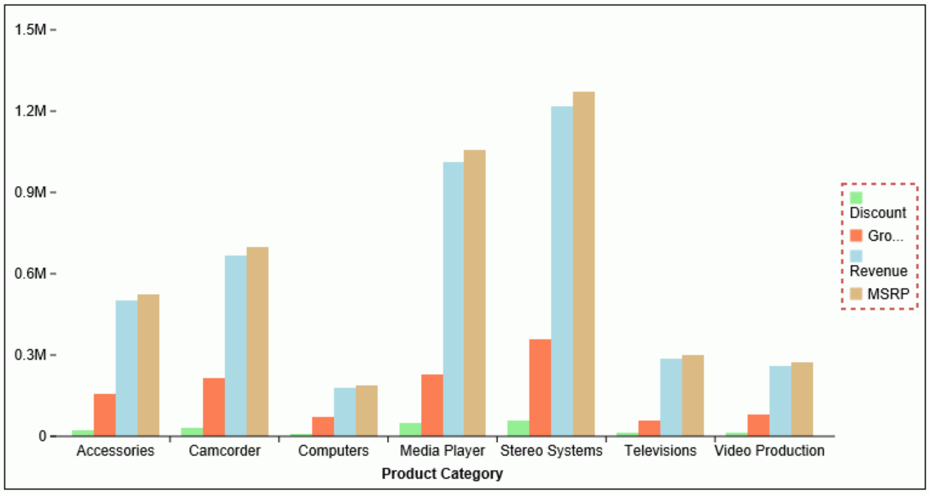
Is a string that includes the percent symbol, representing the percent of space for the legend. The default value is "50%".

Example: Setting the Maximum Percent of Space for the Legend

The following request sets the maxSize property to 10%.

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column = discount_us, bucket = y-axis,$
type=data, column = gross_profit_us, bucket = y-axis,$
type=data, column = revenue_us, bucket = y-axis,$
type=data, column = msrp_us, bucket = y-axis,$
type=data, column = product_category, bucket = x-axis,$
*GRAPH_JS
"xaxis":{"majorGrid": {"visible": false},
"minorGrid": {"visible": false}},
"yaxis":{"majorGrid": {"visible": false},
"minorGrid": {"visible": false}},
"legend": {
"maxSize": "10%",
"lineStyle": {
"width": 2,
"color": "indianred",
"dash": "4 4"}},
"series": [
{"series": 0, "color": "lightgreen"},
{"series": 1, "color": "coral"},
{"series": 2, "color": "lightblue"},
{"series": 3, "color": "burlywood"}
]
*END
ENDSTYLE
END
```

The output is shown in the following image. The legend does not fit in the space allocated to it.



Controlling the Legend Orientation

The legend: orientation property controls whether the legend items are listed vertically or horizontally.

Syntax: How to Control the Legend Orientation

```
"legend":
{
"orientation": "string"
}
```

where:

`"string"`

Can be one of the following values.

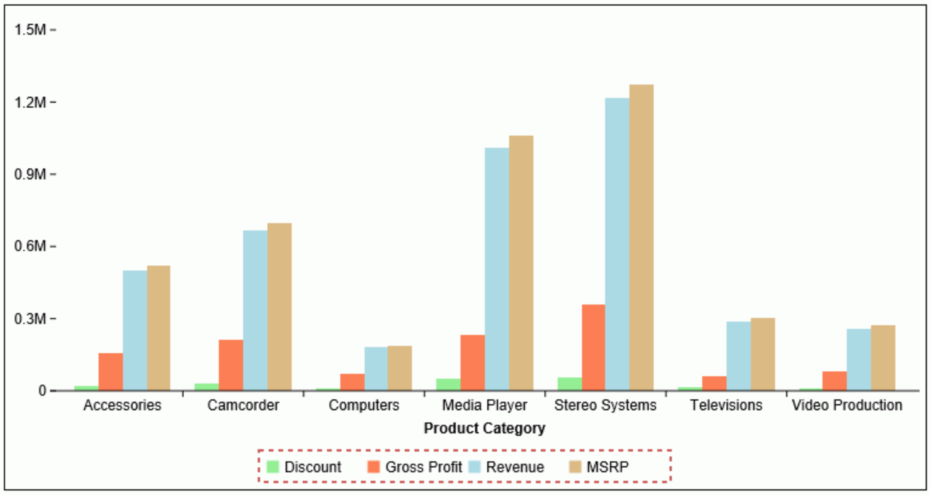
- **"auto".** The chart engine determines the orientation. This is the default value.
- **"vertical".** Lists the legend items vertically.
- **"horizontal".** Lists the legend items horizontally.

Example: Controlling Legend Orientation

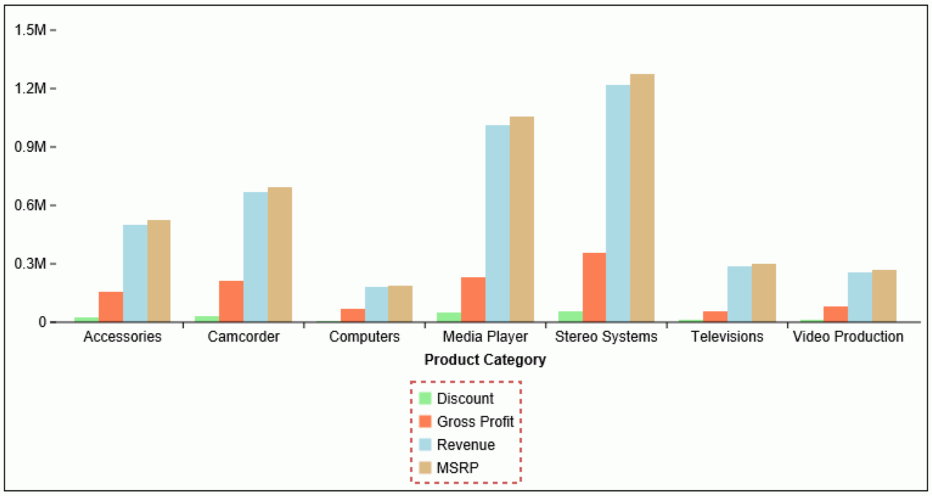
The following request positions the legend at the bottom of the chart and uses the default value for legend orientation.

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column = discount_us, bucket = y-axis,$
type=data, column = gross_profit_us, bucket = y-axis,$
type=data, column = revenue_us, bucket = y-axis,$
type=data, column = msrp_us, bucket = y-axis,$
type=data, column = product_category, bucket = x-axis,$

*GRAPH_JS
"xaxis":{"majorGrid": {"visible": false},
         "minorGrid": {"visible": false}},
"yaxis":{"majorGrid": {"visible": false},
         "minorGrid": {"visible": false}},
"legend": {
    "position": "bottom",
    "orientation": "auto",
"lineStyle": {
    "width": 2,
    "color": "indianred",
    "dash": "4 4"}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
]
*END
ENDSTYLE
END
```

The output is shown in the following image. The chart engine has listed the legend items horizontally.



Changing the orientation property to *"orientation": "vertical"* produces the output shown in the following image.



Controlling the Position of the Legend

The position property controls the location of the legend.

Syntax: How to Control the Legend Position

```
"legend": {"position": position}
```

where:

`"position": position`

Can be a string that defines the location of the legend or a JSON object.

Valid values for a string are:

- "bottom".
- "left".
- "right". This is the default value.
- "top".
- "auto". The chart engine positions the legend.

For more precise positioning, use a JSON object. The following properties are supported.

```
{
   "side": "string",
   "anchor": "string",
  "align": "string",
   "excludeFromLayout": boolean
   "left": number,
   "right": number
   "top": number,
   "bottom": number
 }
```

where:

`"side": "string"`

Can be one of "left", "right", "bottom", "top", or undefined.

`"align": "string"`

For "side": "left" or "side": "right", this can be "top", "middle", or "bottom". For "side": "top" or "side": "bottom", this can be "left", "middle", "right".

`"anchor": "string"`

Is the object used as an alignment reference for the legend. Can be one of "chartBackground", "chartFrame", or "chartAxisFrame". The default is "chartFrame".

`"excludeFromLayout": boolean`

Specifies whether space in the frame should be reserved for the legend. If it is set to *true*, the legend will not take space in the chart area, it will float above the chart. The default is false if the legend side is defined, true otherwise.

`"left": number`

If the legend is on the left, this defined the number of pixels between the anchor object and the left edge of the legend. If the legend is on the right, this is ignored.

`"right": number`

If the legend is on the right, this defined the number of pixels between the anchor object and the right edge of the legend. If the legend is on the left, this is ignored.

`"top": number`

If the legend is on the top, this is the number of pixels between the anchor object and the top edge of legend. If the legend is on the bottom, this is ignored.

`"bottom": number`

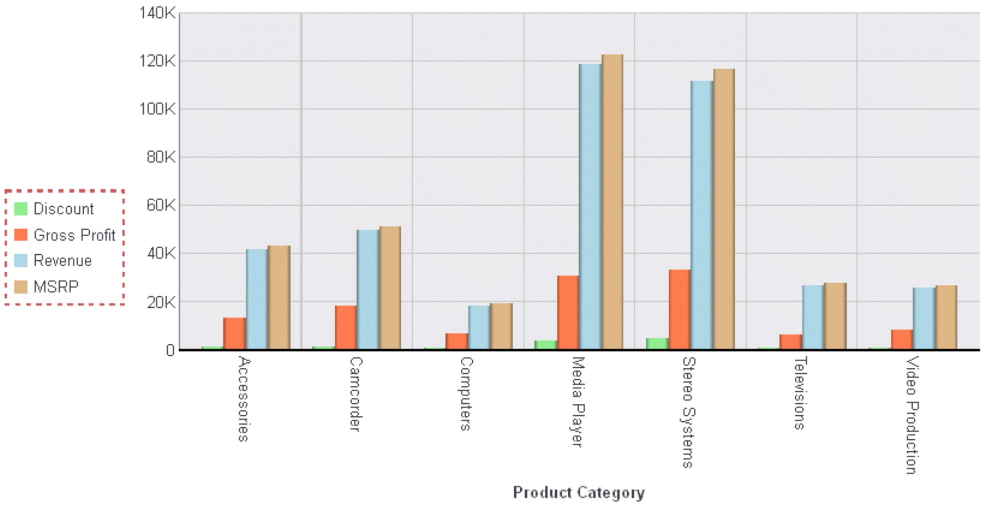
If the legend is on the bottom, this is the number of pixels between the anchor object and the bottom edge of legend. If the legend is on the top, this is ignored.

Example: Controlling the Legend Position Using a String

The following request moves the legend to the left of the chart and draws a border around it:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {
    "position": "left",
    "lineStyle": {
        "width": 2,
        "color": "indianred",
        "dash": "4 4"}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:

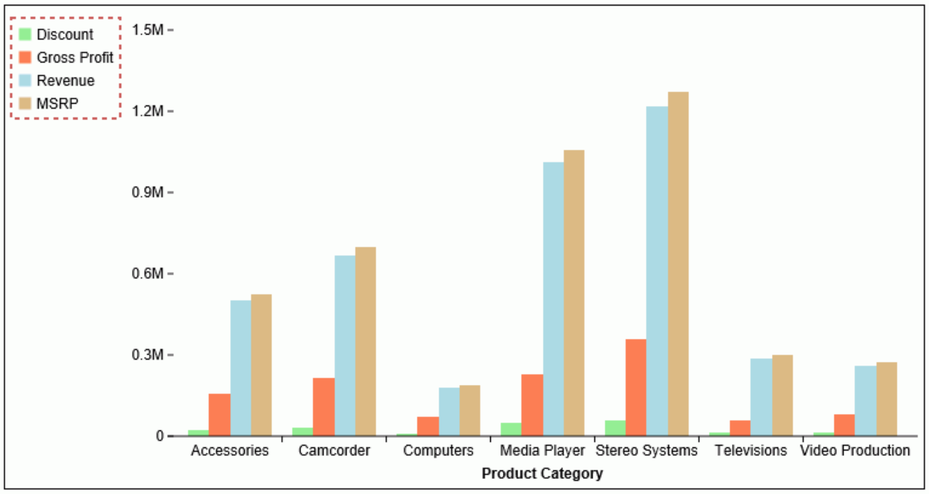


Example: Controlling the Legend Position Using a JSON Object

The following request positions the legend at the top left of the chart background, 5 pixels from the left edge and 10 pixels from the top edge. Space is reserved for the legend.

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column = discount_us, bucket = y-axis,$
type=data, column = gross_profit_us, bucket = y-axis,$
type=data, column = revenue_us, bucket = y-axis,$
type=data, column = msrp_us, bucket = y-axis,$
type=data, column = product_category, bucket = x-axis,$
*GRAPH_JS
"xaxis":{"majorGrid": {"visible": false},
        "minorGrid": {"visible": false}},
"yaxis":{"majorGrid": {"visible": false},
         "minorGrid": {"visible": false}},
"legend": {
"position":
 {
   "side": "left",
   "align": "top",
   "anchor": "chartBackground",
   "excludeFromLayout": false,
   "left": 5,
   "top": 10,
 },
"lineStyle": {
"width": 2,
"color": "indianred",
"dash": "4 4"}
},
"series": [
{"series": 0, "color": "lightgreen"},
{"series": 1, "color": "coral"},
{"series": 2, "color": "lightblue"},
{"series": 3, "color": "burlywood"}
]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Reversing the Order of Legend Items

You can use the legend: reverseOrder property to reverse the default order of legend items.

Syntax: How to Reverse the Order of Legend Items

```
"legend":
{
 "reverseOrder": boolean
}
```

where:

`"reverseOrder": boolean`

Controls the order of legend items. Can be one of the following.

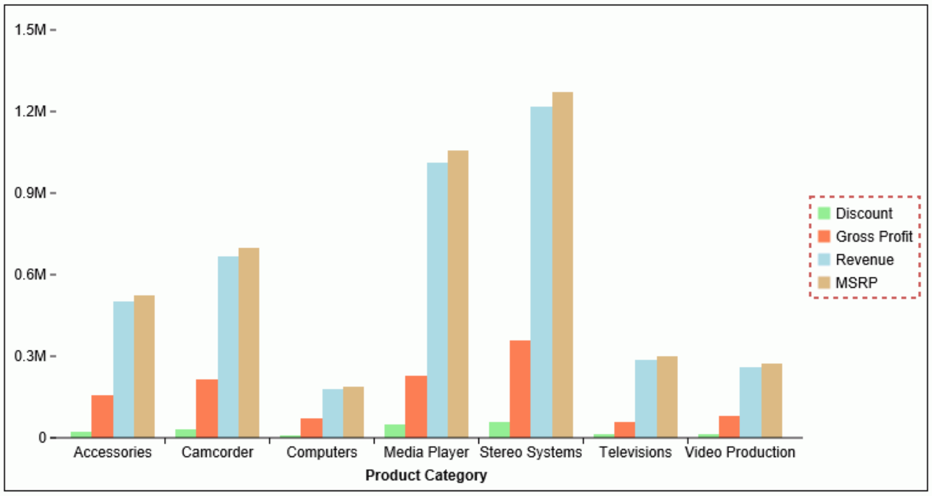
- **"auto".** The chart engine controls the order of legend items.
- **true.** Draws vertical legend items ordered bottom up and horizontal legend items ordered right to left.
- **false.** Draws vertical legend items ordered top down and horizontal legend items ordered left to right. This is the default value.

Example: Reversing the Legend Order

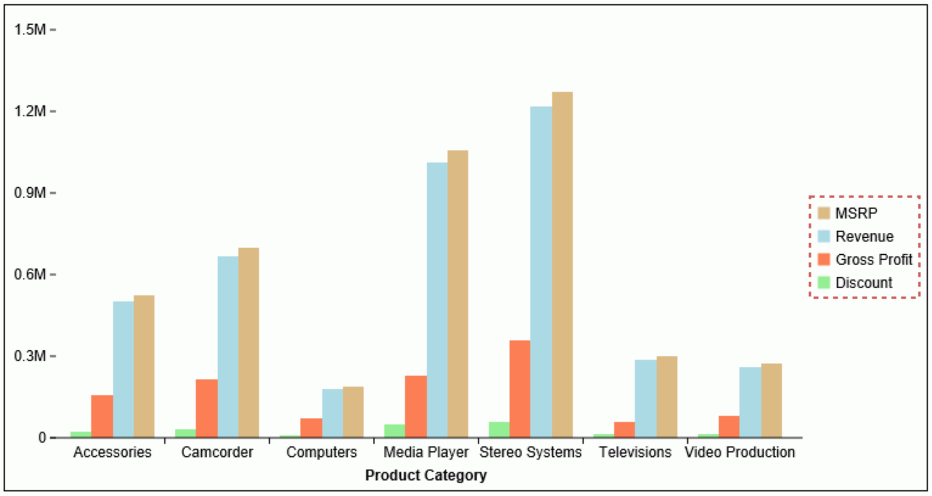
The following request generates a bar chart with the default legend order (reverseOrder:false).

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column = discount_us, bucket = y-axis,$
type=data, column = gross_profit_us, bucket = y-axis,$
type=data, column = revenue_us, bucket = y-axis,$
type=data, column = msrp_us, bucket = y-axis,$
type=data, column = product_category, bucket = x-axis,$

*GRAPH_JS
"xaxis":{"majorGrid": {"visible": false},
         "minorGrid": {"visible": false}},
"yaxis":{"majorGrid": {"visible": false},
         "minorGrid": {"visible": false}},
"legend": {
    "reverseOrder": false,
    "lineStyle": {
        "width": 2,
        "color": "indianred",
        "dash": "4 4"}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Changing reverseOrder to true produces the output shown in the following image. The order of the legend items is reversed.



Applying a Shadow to the Legend Area

The shadow property applies a shadow to the legend area.

Syntax: How to Apply a Shadow to the Legend Area

To enable or disable a default shadow on the legend area, use the following:

```
"legend": {
   "shadow": boolean}
```

where:

`"shadow": boolean`

Can be:

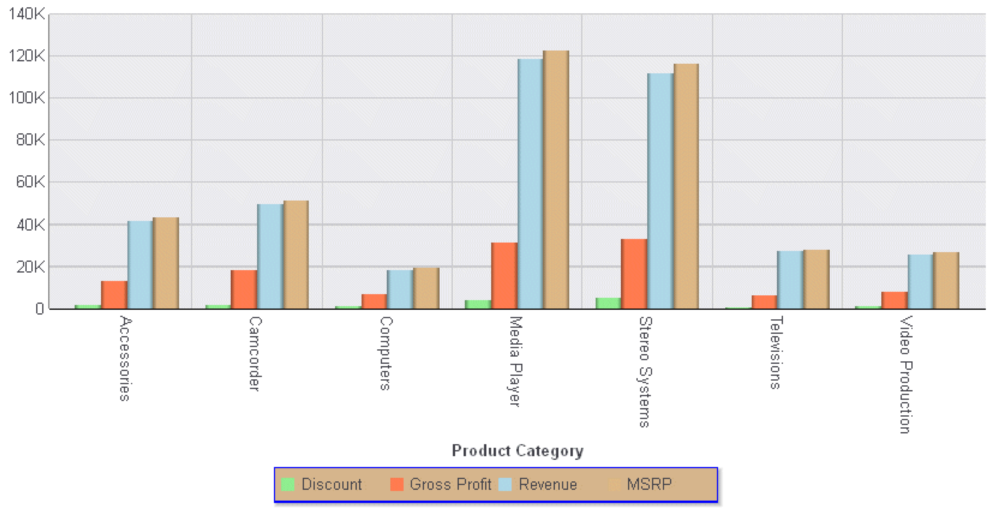
- ❏ true, which enables a default shadow on the legend area.
- ❏ false, which disables a default shadow on the legend area. This is the default value.

Example: Applying a Shadow to the Legend Area

The following request generates a default shadow around the legend area:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {
    "shadow": true,
    "backgroundcolor": "tan",
    "lineStyle": {"color": "blue"}
    },
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Generating a Scrolling Legend

If there are too many series to fit in the chart frame, you can enable a scrolling legend and format the properties of the scroll bar and scroll handle. The scrollbar will not appear unless it is necessary in order to display the entire legend.

Syntax: How to Generate a Scrolling Legend

```
"legend":
  {
  "scroll": {
     "enabled": boolean,
     "size": number,
     "color": "string",
     "handle": {
        "color": "string",
        "hoverColor": "string",
        "border": {
            "width": number,
            "color": "string",
            "dash": "string"
                    }
                  }
           }
  }
```

where:

`"enabled": `*`boolean`*

Controls whether the legend will be scrollable when necessary. Valid values are:

- ❏ true, which enables legend scrolling.
- ❏ <u>false</u>, which disables legend scrolling. This is the default value.

`"size": `*`number`*

Defines the width of the scroll bar in pixels. The default value is 15.

`"color": "`*`string`*`"`

Is a color or gradient definition string that defines the fill color of the scroll bar. The default color is "rgb(240, 240, 240)".

`"handle":`

Defines properties of the scroll bar handle.

`"color": "`*`string`*`"`

Is a color definition string that defines the color of the scroll bar handle. The default color is "grey".

`"hoverColor": "`*`string`*`"`

Is a color definition string that defines the color of the scroll bar handle when the mouse hovers over it or clicks on it. The default color is "rgb(88, 88, 88)".

`"border":`

Defines properties of the scroll bar handle border.

`"width": `*`number`*

Defines the width of the handle border in pixels. The default value is 0.

`"color": "`*`string`*`"`

Is a color definition string that defines the color of the handle border. The default value is 'transparent'.

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

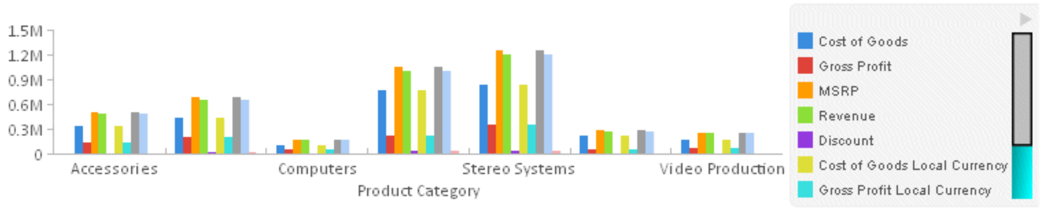
Multiple dashes can be defined within the dash string (for example, "2 4 4 2"), in which case, the dash types will alternate along the border.

Example: Generating a Scrolling Legend

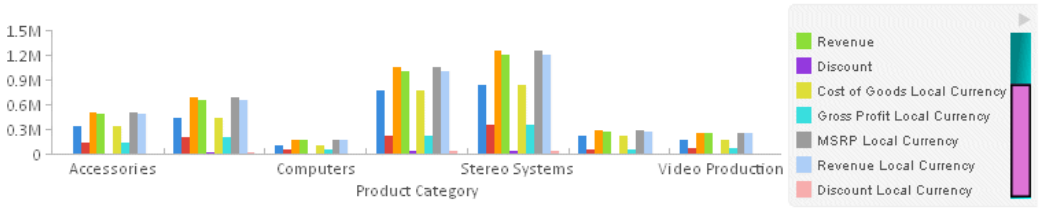
The following request generates a scrolling legend. The scroll bar fill color is a linear gradient that transitions from teal to cyan, the handle color is silver with a black border, and the fill color becomes orchid when the mouse hovers over or clicks it.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US REVENUE_US DISCOUNT_US
COGS_LOCAL GROSS_PROFIT_LOCAL MSRP_LOCAL REVENUE_LOCAL DISCOUNT_LOCAL
BY PRODUCT_CATEGORY
ON GRAPH SET VAXIS 150
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENID_Default.sty,$
*GRAPH_JS
"xaxis": {"labels": {"rotation": 0}},
"legend": {"scroll": {
    "enabled": true,
    "size": 15,
    "color": "linear-gradient(0,0,100%,100%, 20% teal, 95% cyan)",
    "handle": {
        "color": "silver",
        "hoverColor": "orchid",
        "border": {
            "width": 3,
            "color": "black",
            "dash": ""
            }
        }}}
*END
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N4, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N5, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N6, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N7, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N8, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N9, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N10, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N11, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
ENDSTYLE
END
```

The following image shows the scroll bar before it is clicked:



The following image shows the scroll bar handle dragged to the bottom of the legend area:



Generating a Docked Legend

A docked legend is a minimized (collapsed) legend that shows only an active expand button and can also show pictograms of the legend.

Syntax: How to Generate a Docked Legend

```
"legend": {
                        "dock"{
             "enabled": boolean,
             "animate": boolean,
             "minimized": boolean,
             "resizeFrame": boolean,
             "showMarkers": boolean,
             "expandDirection": "string",
             "button": {
                                                        "size": number,
               "color": "string",
               "hoverColor": "string",
               "border": {
                     "width": number,
                     "color": "string",
                     "dash": "string"
                       }
                     }
                  }
        }
```

where:

`"enabled": boolean`

Controls whether a docked legend will be generated. Valid values are:

- ❏ true, which enables a docked legend.
- ❏ false, which disables a docked legend. This is the default value.

`"animate": boolean`

Controls whether the expansion and contraction of the legend is animated using a slide action. Valid values are:

- ❏ true, which animates the expansion and contraction of the legend. This is the default value.
- ❏ false, which disables animation.

`"minimized": boolean`

Controls whether a docked legend is initially collapsed or expanded. Valid values are:

- ❏ true, which initially displays a collapsed docked legend.
- ❏ false, which initially displays an expanded docked legend. This is the default value.

`"resizeFrame": boolean`

Controls whether the frame shrinks and expands as the legend is collapsed and expanded. Valid values are:

- ❏ true, which resizes the frame as the legend collapses or expands. This is the default value
- ❏ false, which does not resize the frame as the legend collapses or expands.

`"showMarkers": boolean`

Controls whether a collapsed docked legend displays the legend markers. Valid values are:

- ❏ true, which displays legend markers on a collapsed legend. This is the default value.
- ❏ false, which disables does not display markers on a collapsed legend.

`"expandDirection": "string"`

Specifies the direction of expansion ("left", "right", "top", "bottom" for a legend whose position is *free*. For legends whose positions are not free, the value is *undefined*, and the chart engine determines the direction of expansion.

`"button":`

Defines properties of the expand button.

`"size":` *`number`*

Defines the size of the button (defined as two times the distance from the center of the object to the furthest corner of the shape) in pixels, or can be 'auto' to draw the button the same size as the legend markers. The default value is 'auto'.

`"color": "`*`string`*`"`

Is a color specification string that defines the button color. The default value is "grey".

`"hoverColor": "`*`string`*`"`

Is a color specification string that defines the button color when the mouse hovers over it or clicks it. The default value is 'black'.

`"border":`

Defines properties of the expand button border.

`"width":` *`number`*

Defines the width of the button border in pixels. The default value is zero (0).

`"color": "`*`string`*`"`

Is a color specification string that defines the color of the button border. The default value is "transparent".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

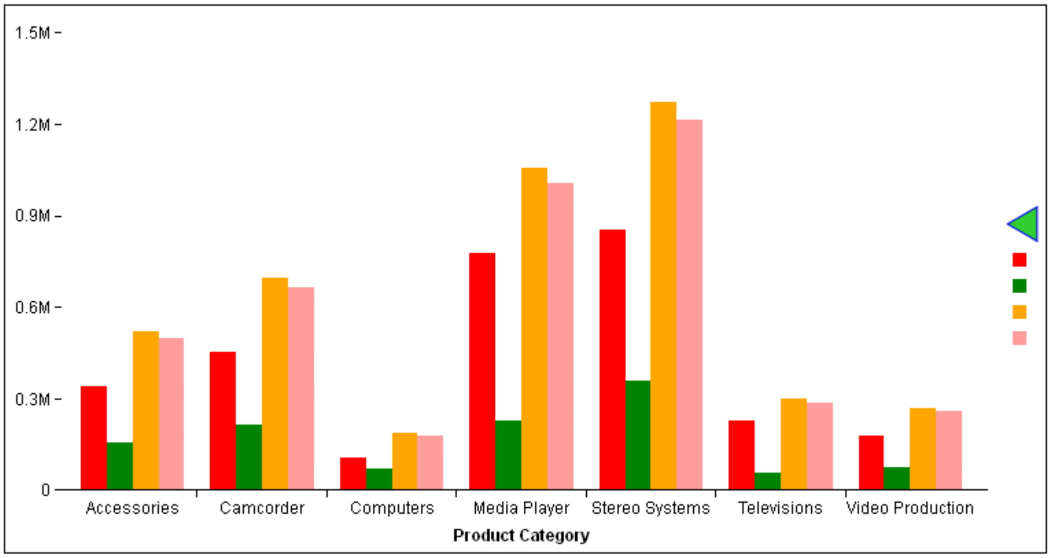
Multiple dashes can be defined within the dash string (for example, "2 4 4 2"), in which case, the dash types will alternate along the border.

Example: Generating a Docked Legend

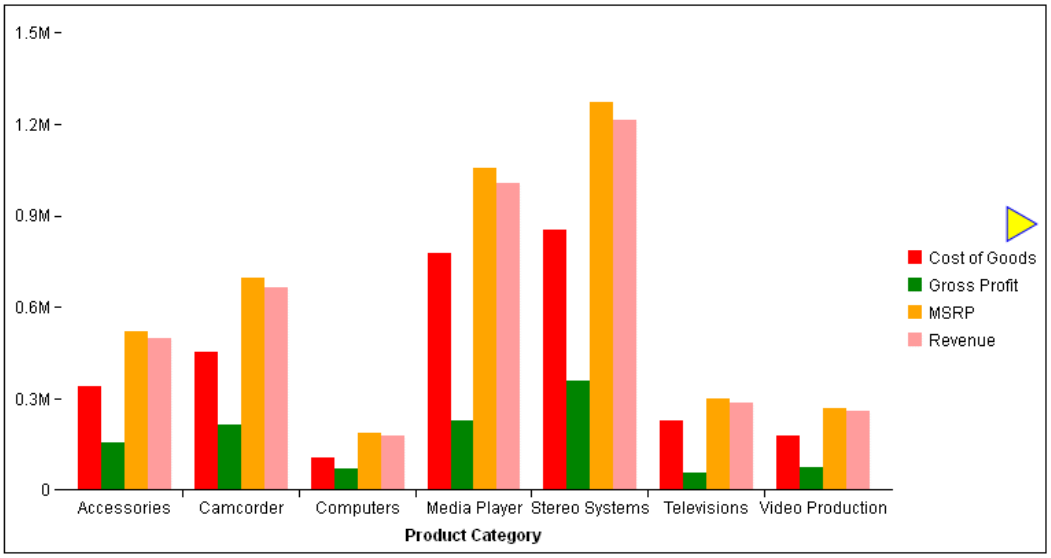
The following request generates a legend that is initially collapsed but shows the legend markers. The expand button has a diameter of 25 pixels, is lime green with a blue border, and turns yellow when the mouse hovers over or clicks it:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
MSRP_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
*GRAPH_JS
"xaxis": {"majorGrid": {"visible": false}},
"yaxis": {"majorGrid": {"visible": false}},
"legend": {"dock": {
    "enabled": true,
    "animate": true,
    "minimized": true,
    "resizeFrame": true,
    "showMarkers": true,
    "expandDirection": "undefined",
    "button": {
        "size": 25,
        "color": "limegreen",
        "hoverColor": "yellow",
        "border": {
            "width": 1,
            "color": "blue",
            "dash": ""
            }}
    }}
*END

TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=MSRP_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The following image shows the initial view, with the legend collapsed:



The following image shows the legend expanded, with the mouse hovering over the button (the hover color is displaying). The chart frame has resized:



Formatting the Legend Title

The title property controls the visibility and format of the legend title.

Syntax: How to Format the Legend Title

```
"legend":
  {
   "title":
   {
      "visible": boolean,
      "text": "string",
      "font": "string",
      "color": "string"
      "tooltip": "string",
      "wrap": number
   }
 }
```

where:

`"visible": boolean`
Can be:

- ❏ true, which makes the legend title visible.
- ❏ false, which makes the legend title not visible. This is the default value.
- ❏ "auto", which lets the chart engine determine legend visibility and placement.

`"text": "string"`

Is a string that defines the legend title text. The default value is "Legend Title".

`"font": "string"`

Is a string that defines the size, style, and typeface of the legend title. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the legend title using a color name or numeric specification string. The default value is "black".

For information about defining colors, see *Colors and Gradients* on page 85.

`"tooltip": "string"`
Is a string, or a function that returns a string, to display when the mouse hovers over the legend title.

"wrap": *number*

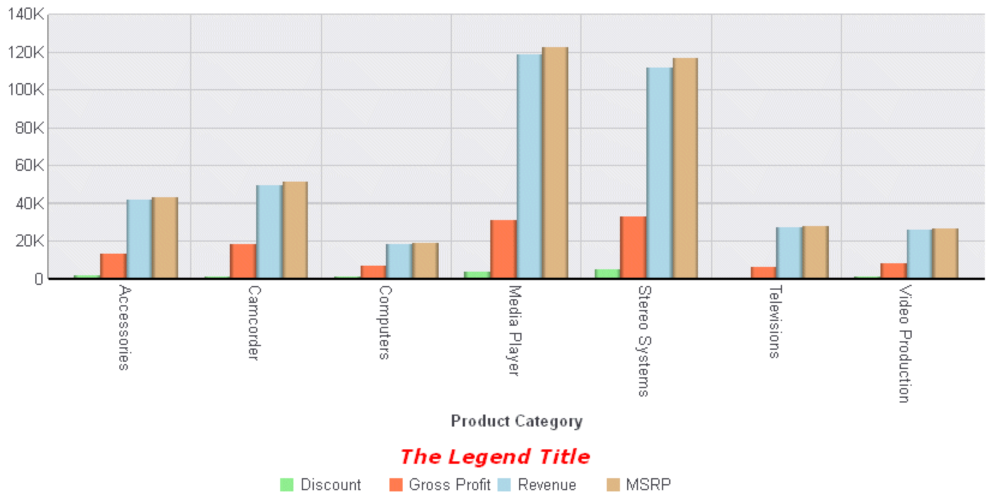
Defines the number of lines for wrapping long titles. Can be an integer, which wraps long labels to that number of lines (if necessary), or "auto", which wraps long titles into as many lines as necessary. The default value is 1.

Example: Formatting the Legend Title

The following request defines a legend title that is red with a bold-italic 12pt Verdana font:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"title": {
        "visible": true,
        "text": "The Legend Title",
        "font": "bold italic 12pt Verdana",
        "color": "red"
        }},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Controlling the Visibility of the Legend Area

The visible property controls the visibility of the legend area.

Syntax: How to Control the Visibility of the Legend Area

```
"legend": {"visible": boolean}
```

where:

`"visible": boolean`

Can be:

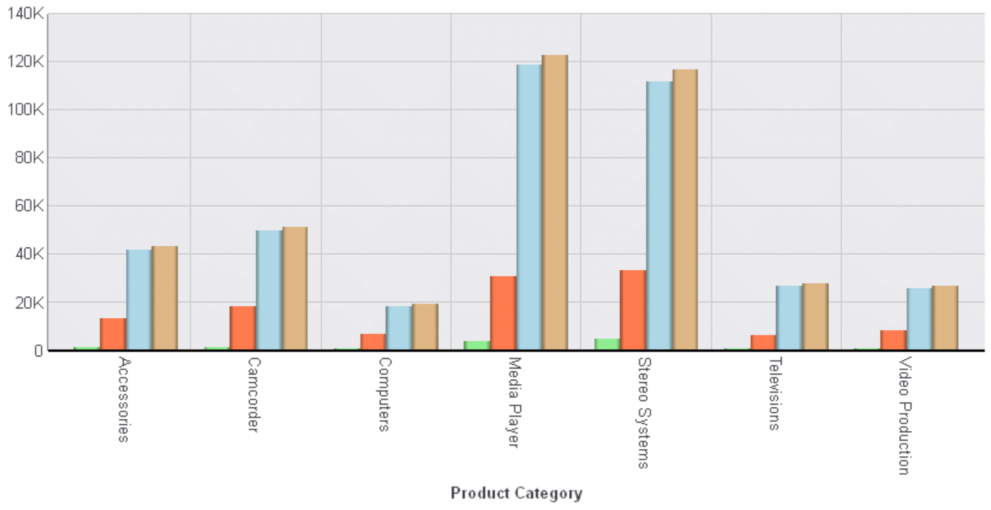
- true, which makes the legend visible. This is the default value.
- false, which makes the legend not visible.
- "auto", which makes the legend visible if there is more than one series on the chart.

Example: Controlling Legend Visibility

The following request removes the legend from the chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}
    ]
*END
ENDSTYLE
END
```

The output is:



Formatting the Size Legend

A size legend is generated when a measure is assigned to the size attribute category. The size legend is drawn as two or three circles within one another, each sized to match the size of markers in the chart.

Syntax: How to Format the Size Legend

```
"legend":
{
  "sizeLegend":
  {
    "layout": "string",
    "lineStyle":
     {
      "width": number,
      "color": "string",
      "dash": "string"
      }
   }
}
```

where:

`"layout": "string"`

Defines the look of the size legend. Valid values are "circle" or "quarterCircle". The default value is "circle".

`"lineStyle"`

Defines the properties of the circles or quarter circles.

`"width": number`

Defines the width, in pixels, of the circles or quarter circles. The default value is 0.

`"color": "string"`

Is a string that defines the color of the circles or quarter circles. The default value is "black".

`"dash": "string"`

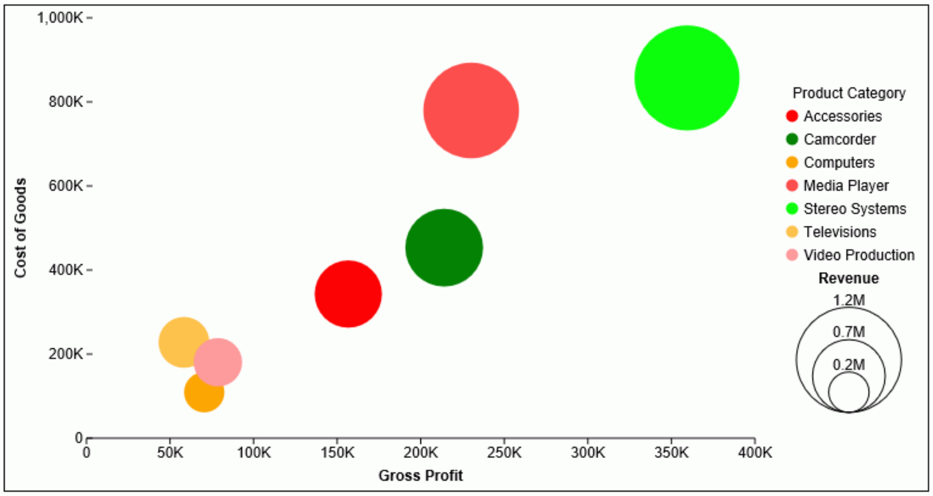
Is a string that defines the dash style of the circles or quarter circles. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Example: Formatting the Size Legend

The following request generates a bubble chart that has a measure in the size attribute category.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=size,
*GRAPH_JS
"xaxis": {"majorGrid": {"visible":false},
           "minorGrid": {"visible":false}},
"yaxis": {"majorGrid": {"visible":false},
           "minorGrid": {"visible":false}},
*END
ENDSTYLE
END
```

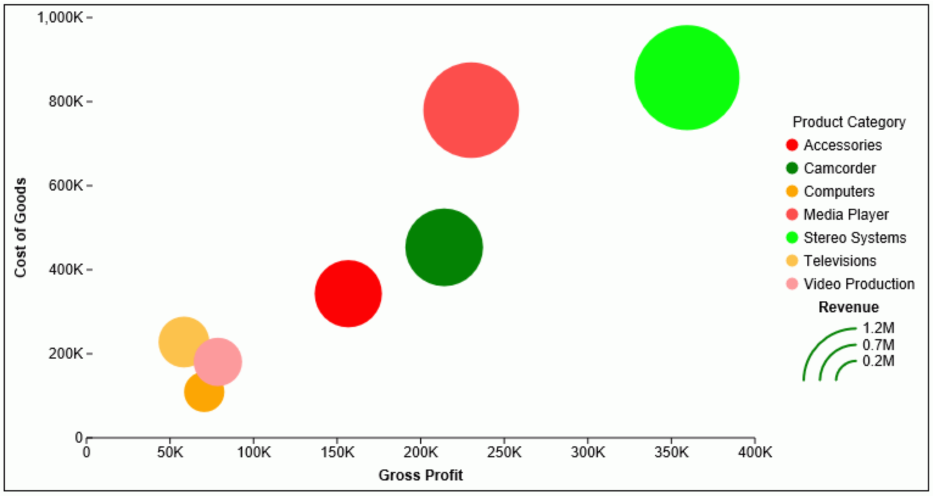
The output is shown in the following image. The default size legend is visualized as three black circles.



The following version of the request formats the size legend as green quarter-circles.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=size,
*GRAPH_JS
"legend":
    {
  "sizeLegend":
  {
    "layout": "quarterCircle",
    "lineStyle":
     {
      "width": 2,
      "color": "green",
      "dash": ""
      }
   }
},
"xaxis": {"majorGrid": {"visible":false},
           "minorGrid": {"visible":false}},
"yaxis": {"majorGrid": {"visible":false},
           "minorGrid": {"visible":false}},
*END
ENDSTYLE
END
```

The output is shown in the following image.



Chapter 8

Axis Properties

This chapter describes how to control the appearance of chart axes.

In this chapter:

Axis Properties Overview

The axis properties control the appearance of chart axes (xaxis, yaxis, y2axis, zaxis). An axis can be numeric or ordinal (text strings) depending on the data that is supplied to draw the chart. The y2axis only appears in pareto charts and Bar, Line, and Area charts where the series yAxisAssignment property assigns a series to the y2axis. The zaxis only appears on 3D charts and heatmap charts.

The following code segment shows the default settings:

```
"xaxis": {


  "mode": undefined,
  "min": undefined,
  "max": undefined,
  "mustIncludeZero": true,
  "intervalMode": undefined,
  "intervalValue": undefined,
  "bIsLog": false,
  "numberFormat": "auto",
  "swapChartSide": false,
  "invert": false,
  "altFrameColor": undefined,
  "colorBands": [],


"groupFit":
         {"rule": "auto",
          "value": undefined
         },


  "title": {
    "text": "the WebFOCUS column title",
    "visible": false,
    "font": "bold 9pt Sans-Serif",
    "color": "black"
  },
  "labels": {
    "visible": true,
    "font": "7.5pt Sans-Serif",
    "color": "black",
    "excludeMin": false,
    "excludeMax": false,
    "nestingConcatSymbol": "null",
    "nestingLineStyle": {
                                                 "width": 1,
            "color": "black",
            "dash": ""
        }
    "rotation": undefined
  },
  "baseLineStyle": {
    "width": 1,
    "color": "black",
    "dash": ""
  },
  "bodyLineStyle": {
    "width": 1,
    "color": "transparent",
    "dash": ""
  },
"majorGrid": {
  "visible": true,
  "aboveRisers": true,
  "lineStyle": {
    "width": 1,
    "color": "rgba(255, 255, 255, 0.3)",
    "dash": ""
  },
  "ticks": {
    "length": 5,
    "visible": true,
    "style": "inner",
    "lineStyle": {
      "width": 1,
      "color": "black"
    }
  }
},
"minorGrid": {
  "visible": false,
  "count": undefined,
  "lineStyle": {
    "width": 1,
    "color": "black",
    "dash": ""
  },
  "ticks": {
    "length": 5,
    "visible": false,
    "style": "inner",
    "lineStyle": {
      "width": 1,
      "color": "black"
    }
  }
},

"labelLayout": {
  "stagger": "auto",
  "skip": "auto",
  "truncate": "auto"
            },
"scroll": {
  "style": "simple",
     "miniChartProperties": {
     "height": n,
     "dataselection": "selectionRect",
     "intervalButtons": "auto"
                                                     },

                               }
},
"yaxis": {
  "mode": undefined,
  "min": undefined,
  "max": undefined,
  "mustIncludeZero": true,
  "intervalMode": undefined,
  "intervalValue": undefined,
  "bIsLog": false,
  "numberFormat": "auto",
  "swapChartSide": false,
  "invert": false,
  "altFrameColor": undefined,
  "colorBands": [],
  "title": {
    "text": "Y Axis Title",
    "visible": false,
    "font": "bold 9pt Sans-Serif",
    "color": "black"
    "position": "orthogonal"
},
  "labels": {
    "visible": true,
    "font": "7.5pt Sans-Serif",
    "color:" "black",
    "excludeMin": false,
    "excludeMax": false,
    "replaceMin": undefined,
    "replaceMax": undefined,
    "rotation": undefined
  },
  "baseLineStyle": {
    "width": 1,
    "color": "black",
    "dash": ""
  },
  "bodyLineStyle": {
    "width": 1,
    "color": "transparent",
    "dash": ""
  },
 "majorGrid: {
    "visible: true,
    "aboveRisers: true,
    "lineStyle: {
      "width: 1,
      "color: "rgba(255, 255, 255, 0.3)",
      "dash: ""
    },
    "ticks: {
      "length: 5,
      "visible: true,
      "style: "inner",
      "lineStyle: {
        "width: 1,
        "color: "black"
      }
    }
  },
  "minorGrid": {
    "visible": false,
    "count": undefined,
    "lineStyle": {
      "width": 1,
      "color": "black",
      "dash": ""
    },
    "ticks": {
      "length": 5,
      "visible": false,
      "style": "inner",
      "lineStyle": {
        "width": 1,
        "color": "black"
      }
  },
  "colorScale":{ (deprecated, use the chart level colorScale object)
    "colors": ["#253494", "#2C7FB8", "#41B6C4", "#A1DAB4"]
  }
},
"y2axis": {
  "min": undefined,
  "max": undefined,
  "mustIncludeZero": true,
  "intervalMode": undefined,
  "intervalValue": undefined,
  "bIsLog": false,
  "numberFormat": "auto",
  "invert": false,
  "altFrameColor": undefined,
  "colorBands": [],
  "title": {
    "text": "Y2 Axis Title",
    "visible": false,
    "font": "bold 9pt Sans-Serif",
    "color": "black"
    "position": "orthogonal"
},
  "labels": {
    "visible": true,
    "font": "7.5pt Sans-Serif",
    "color:" "black",
    "excludeMin": false,
    "excludeMax": false,
    "replaceMin": undefined,
    "replaceMax": undefined,
  },
  "baseLineStyle": {
    "width": 1,
    "color": "black",
    "dash": ""
  },
  "bodyLineStyle": {
    "width": 1,
    "color": "transparent",
    "dash": ""
  },
 "majorGrid": {
    "lockToY1": false,
    "visible": true,
    "aboveRisers": true,
    "lineStyle": {
      "width": 1,
      "color": "rgba(255, 255, 255, 0.3)",
      "dash": ""
    },
    "ticks": {
      "length": 5,
      "visible": true,
      "style": "inner",
      "lineStyle": {
        "width": 1,
        "color": "black"
      }
    }
  },
  "minorGrid": {
    "visible": false,
    "count": undefined,
    "lineStyle": {
      "width": 1,
      "color": "black",
      "dash": ""
    },
    "ticks": {
      "length": 5,
      "visible": false,
      "style": "inner",
      "lineStyle": {
        "width": 1,
        "color": "black"
      }
    }
  }
},
zaxis: {
  "mode": undefined,
  "min": undefined,
  "max": undefined,
  "mustIncludeZero": true,
  "intervalMode": undefined,
  "intervalValue": undefined,
  "bIsLog": false,
  "numberFormat": "auto",
  "swapChartSide": false,
  "invert": false,
  "altFrameColor": undefined,
  "colorBands": [],
  "title": {
    "text": "Z Axis Title",
    "visible": false,
    "font": "bold 9pt Sans-Serif",
    "color": "black"
    "position": "orthogonal"
},
  "labels": {
    "visible": true,
    "font": "7.5pt Sans-Serif",
    "color:" "black",
    "excludeMin": false,
    "excludeMax": false,
    "replaceMin": undefined,
    "replaceMax": undefined,
    "rotation": undefined
  },
  "baseLineStyle": {
    "width": 1,
    "color": "black",
    "dash": ""
  },
  "bodyLineStyle": {
    "width": 1,
    "color": "transparent",
    "dash": ""
  },
 "majorGrid": {
    "visible": false,
    "aboveRisers": true,
    "lineStyle": {
      "width": 1,
      "color": "rgba(255, 255, 255, 0.3)",
      "dash": ""
    },
    "ticks": {
      "length": 5,
      "visible": true,
      "style": "outer",
      "lineStyle": {
        "width": 1,
        "color": "black"
      }
    }
  },
  "minorGrid": {
    "visible": false,
    "count": undefined,
    "lineStyle": {
      "width": 1,
      "color": "black",
      "dash": ""
    },
    "ticks": {
      "length": 5,
      "visible": false,
      "style": "inner",
      "lineStyle": {
        "width": 1,
        "color": "black"
      }
    }
  }
},
```

Note: The yaxis:colorScale properties have been deprecated and replaced by the colorScale properties described in *Defining a Color Scale Color Mode* on page 771.

Creating an Axis List

Instead of using xaxis and yaxis properties, you can create an axisList, which supports any number of x- and y-axes. All of the xaxis and yaxis properties are supported for the axes listed in the axisList.

Using an axis list is the recommended way to specify axis properties.

Syntax: How to Specify Axis Properties Using an Axis LIst

```
axisList: {
                x1: {xaxis_properties},
                x2: {xaxis_properties},
                      ...
                xi: {xaxis_properties},
                y1: {yaxis_properties},
                y2: {yaxis_properties},
                     ...
                yj: {yaxis_properties}
}
```

where:

`x1, ... xi, y1 ... yj`
: Are any number of x-axes and y-axes.

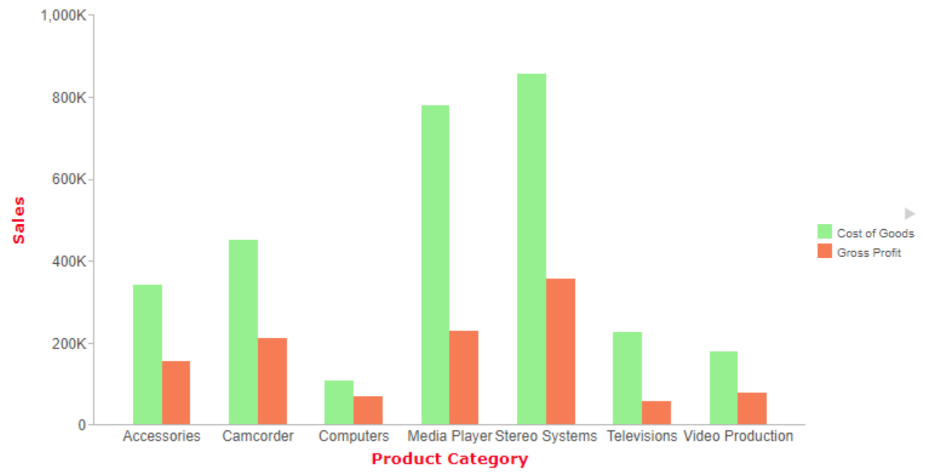
`xaxis_properties, yaxis_properties`
: Are supported properties for the axis.

Example: Using an Axis List

The following request formats the x-axis and y-axis titles on a bar chart, using an axis list:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=n1, bucket = x-axis,$
type=data, column=n2, bucket = y-axis,$
type=data, column=n3, bucket = y-axis,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
*GRAPH_JS_FINAL
"axisList": {
"x1": {"title": {
"visible": true, "color": "red", "font": "bold 10pt Verdana"}},
"y1": {"title": {
"visible": true, "color": "red", "font": "bold 10pt Verdana", "text":
"Sales"}}
},
"series": [
{"series": 0, "color": "lightgreen"},
{"series": 1, "color": "coral"}]
*END
ENDSTYLE
END
```

The output is shown in the following image:



Filling Alternate Segments of Axis Grid Lines With a Specified Color

The altFrameColor property fills every other segment of axis grid lines with a specified color.

Syntax: How to Fill Alternate Segments of Axis Grid Lines With a Specified Color

```
"axisname":
{
  "altFrameColor": "color"},
```

where:

axisname

Can be one of the axes on the axisList, or xaxis, yaxis, y2axis, or zaxis.

"altFrameColor": "*color*"

Can be:

- A color defined by a color name or numeric specification string, or a gradient defined by a string.
- A JSON gradient definition.

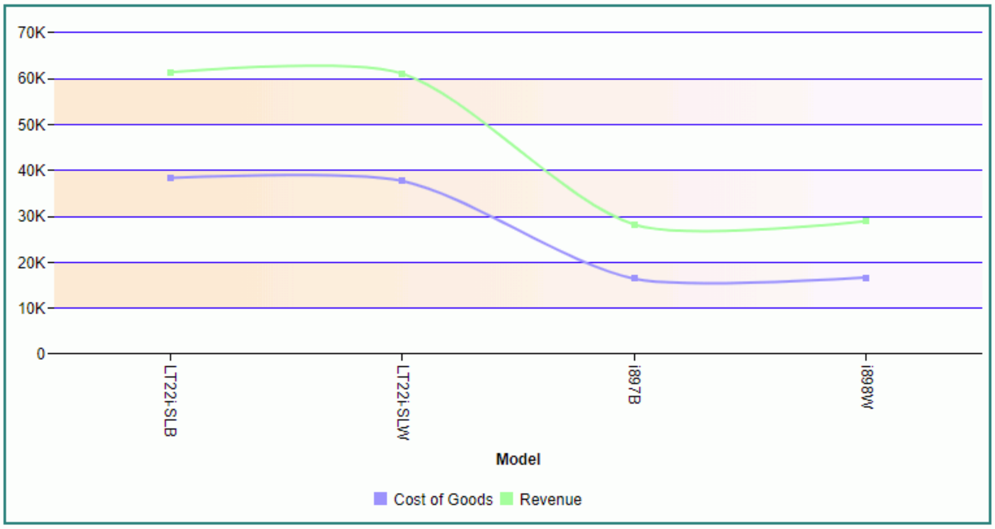
For information about defining colors and gradients, see *Colors and Gradients* on page 85.

Example: Filling Alternate Segments of Axis Grid Lines With a Specified Color

The following request against the WF_RETAIL_LITE data source generates a vertical line chart in which the sections between the yaxis major grid lines alternate between being transparent and having a gradient fill that transitions from antique white to ghost white:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "altFrameColor":
        "linear-gradient(0%,0%,100%,0%, 20% antiquewhite, 95% ghostwhite)",
    "majorGrid": {"visible": true,
        "lineStyle": {"width": 1, "color": "blue"}
    }}}
*END
ENDSTYLE
END
```

The output is:



Controlling Riser Width

By default, the chart engine fits chart output to the size of the container. It generates a scroll bar when all of the risers will not fit in the container nicely. if the AUTOFIT parameter is set to RESIZE and the container is resized, the risers will also be resized.

Using the xaxis:groupFit properties, you can control riser width. You can set a minimum, maximum, or exact width for risers, or you can specify a minimum number of groups to show in the frame if a scroll bar is generated.

Syntax: How to Control Riser Width

```
"axisname": {
 "groupFit":
   {"rule": "string",
      "value": value      }
   }
```

where:

`"axisname"`

Is an axis on the axisList, or xaxis.

`"rule": "string"`

Is one of the following rules that controls how the riser widths are determined.

- **"auto".** Size risers according to the width of the chart frame. This is the default value.
- **"minSize".** Risers are at least *value* pixels wide.
- **"maxSize".** Risers are never more than *value* pixels wide.
- **"exactSize".** Risers are always exactly *value* pixels wide.
- **"maxCount".** If the bars do not fit nicely, add a scroll bar but ensure that *value* number of risers are visible on the chart at the same time.
- **"labelHeight".** Sizes the width of risers on a horizontal bar chart. to the group label font size on the x-axis.

`"value": value`

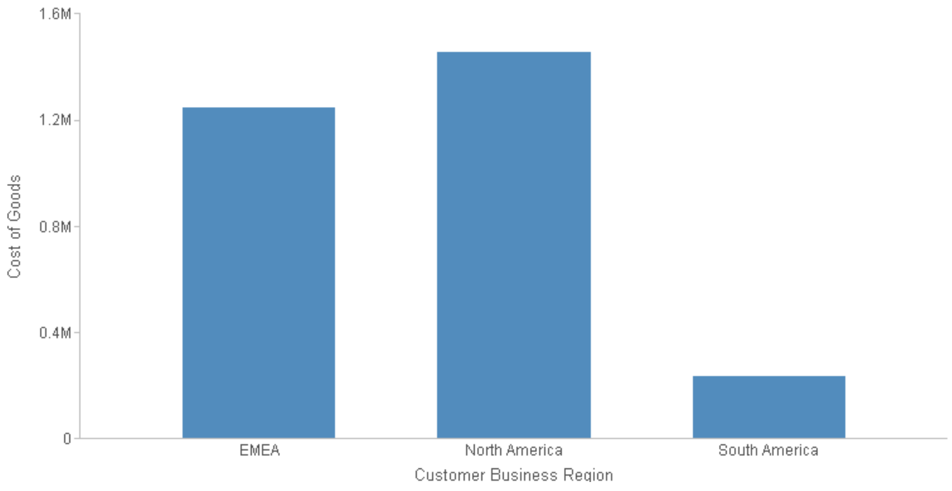
Is the width in pixels for "minSize", "maxSize", or "exactSize", or the number of groups for "scrollCount". It is ignored for "auto". For "labelHeight", adds additional padding to the label height. Can be a percentage string or a number in pixels. The default value is *undefined*.

Example: Controlling Riser Width

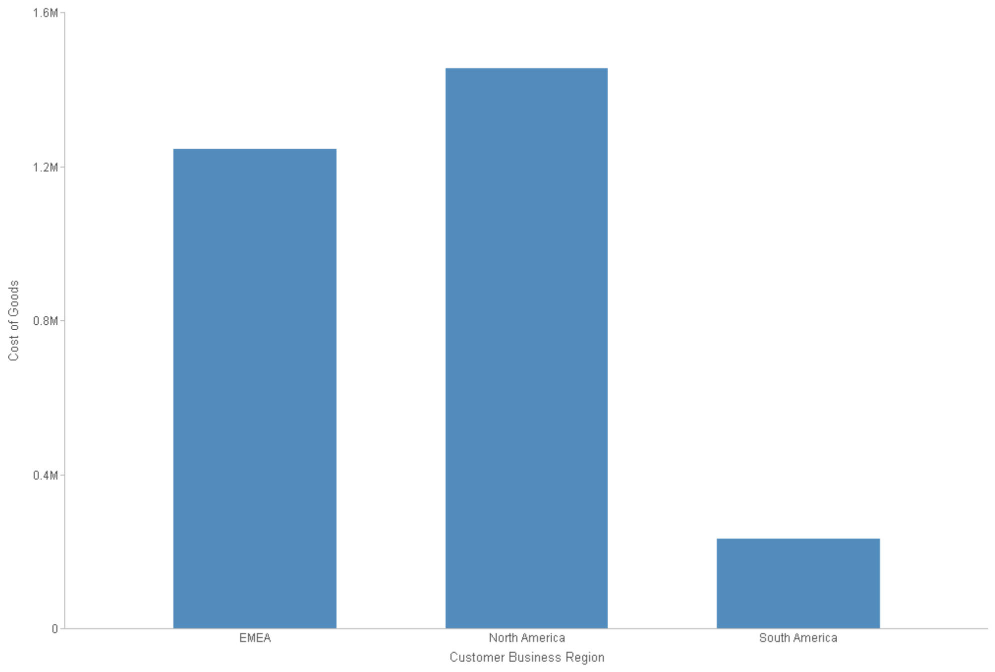
The following request generates a chart with risers that are sized automatically. The AUTOFIT parameter is set to RESIZE, so when the frame is resized, the riser widths are resized.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT RESIZE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=side-by-side,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=x-axis, $
*END
ENDSTYLE
END
```

The initial chart display is shown in the following image.



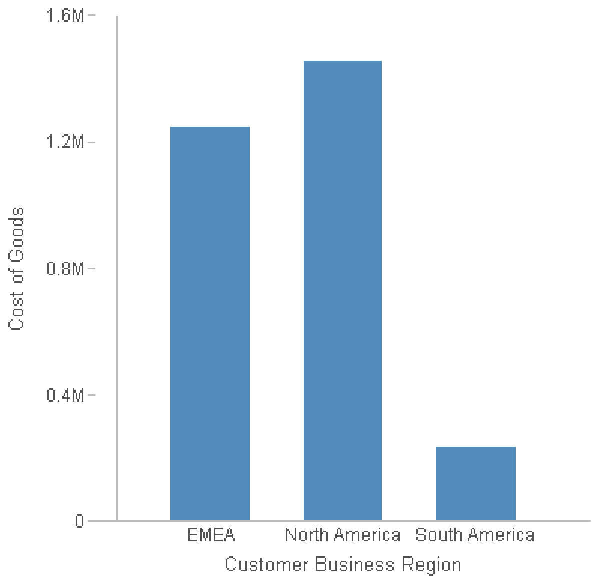
The following image shows the same chart when the frame is resized to its maximum. The riser widths become wider to fill the chart frame.



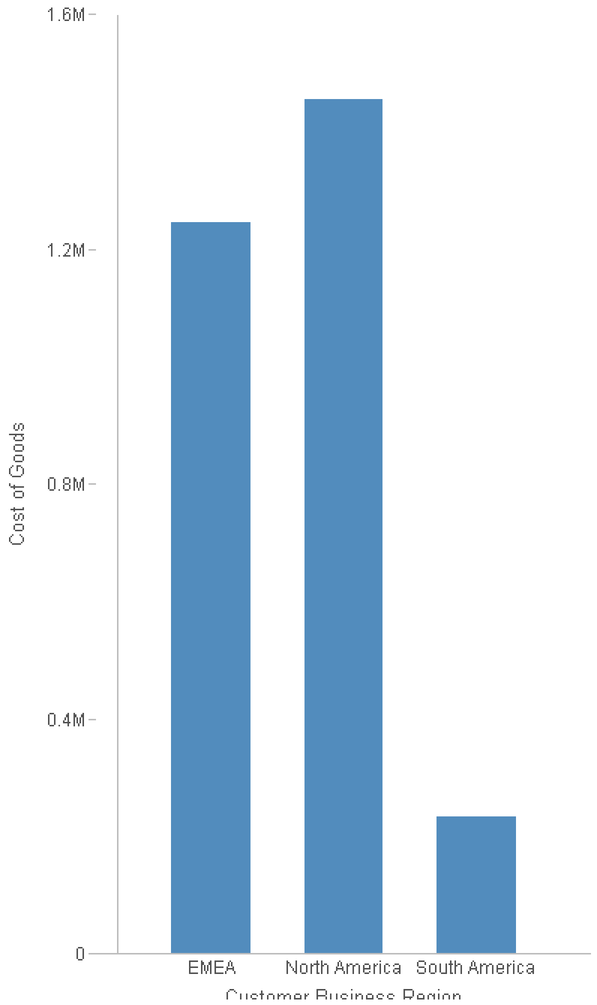
The following version of the request uses the groupFit properties to make the risers exactly 100 pixels wide.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT RESIZE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=side-by-side,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=x-axis, $
*GRAPH_JS
"xaxis": {"groupFit": {
    "rule": "exactSize",
    "value": 100
    }}
*END
ENDSTYLE
END
```

The initial chart display is shown in the following image.



The following image shows the chart when the frame is resized to its maximum size. The risers are still the same width.

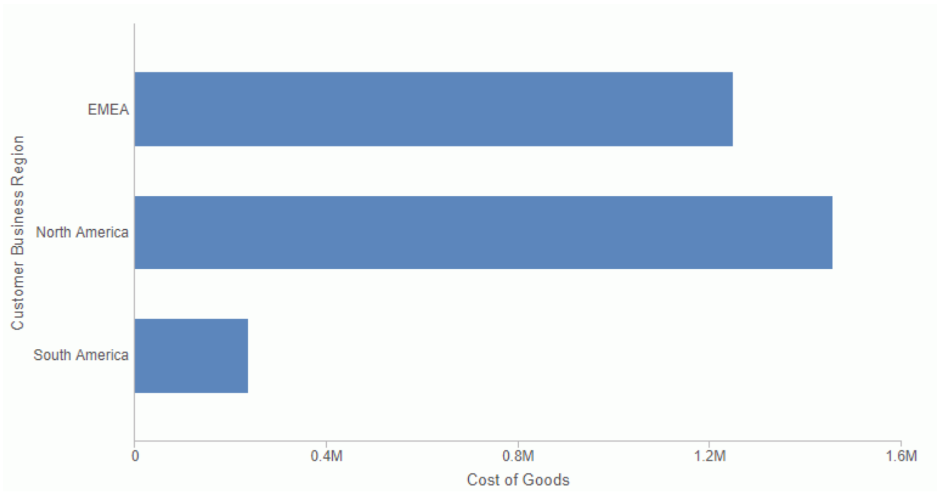


Example: Sizing the Width of Horizontal Bars to the Font Size of the Group Labels

The following request generates a horizontal bar chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
TYPE=REPORT, CHART-ORIENTATION=HORIZONTAL,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
ENDSTYLE
END
```

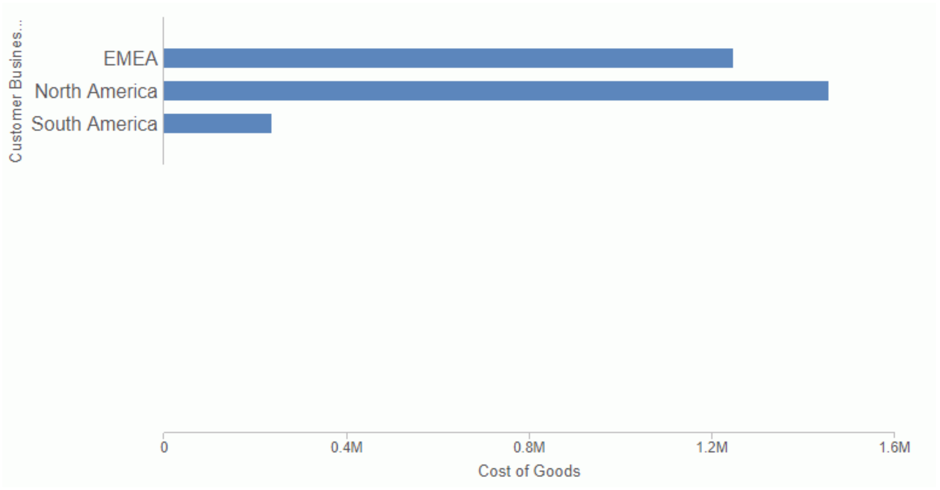
The output is shown in the following image. The bars are much wider than the group labels.



The following version of the request sets a font size for the labels and uses the groupFit labelHeight rule to match the bar width to the font size, with a padding factor of 80%.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
TYPE=REPORT, CHART-ORIENTATION=HORIZONTAL,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
*GRAPH_JS
"xaxis":
       {
       "labels": {"font": "12pt Sans-Serif"},
       "groupFit":
         {
          "rule": "labelHeight",
          "value": "80%"
          }
        }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Sorting Stacked Bar Chart Risers

In a stacked bar chart, one riser represents the total value for all series in a group. You can use the axis sort property to sort these stacked risers in ascending or descending order of the numeric axis values.

Syntax: How to Sort Stacked Bar Chart Risers

```
"axisname": {"sort": {"order": "string", "by": "string"}}
```

where:

`"axisname"`

Is the axis along which the sorted risers will be drawn. It can be any axis on the axisList, or xaxis, yaxis, y2axis, or zaxis.

`"sort"`

Defines the sort properties.

`"order": "string"`

Defines how the risers will be sorted. Valid values are:

- "ascending"
- "descending"

The default is to not to sort the risers.

`"by": "string"`

Defines the numeric axis that represents the values to be sorted. Valid values are:

- "x"
- "x*n*", where *n* is an index for split x-axes.
- "y"
- "y*n*", where *n* is an index for split y-axes.
- "z"
- "matrix*n*", where *n* is the row index (for verical charts) or column index (for horizontal charts) to sort by.

  Note: Matrix rows and columns are zero-indexed. That is, "matrix0" is the first row or column.

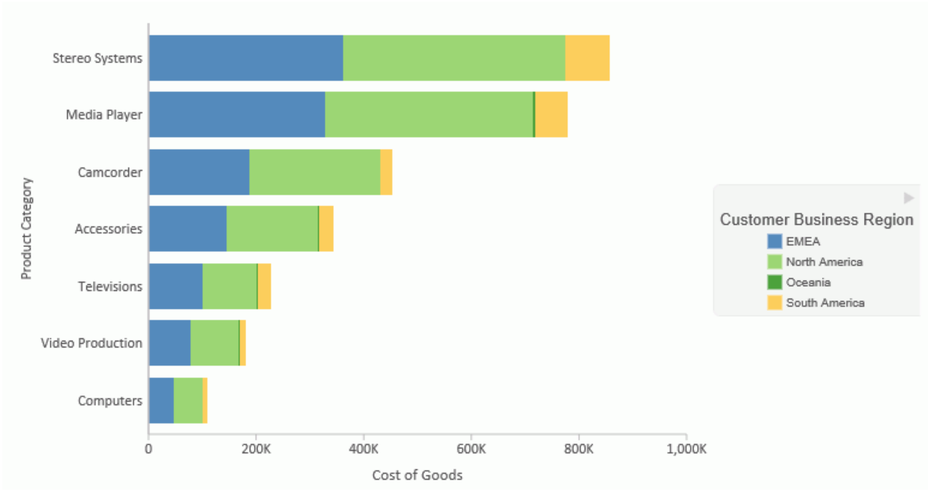
- "matrix*n*, y*i*", to sort by a split y-axis within a matrix row or column.

Example: Sorting Stacked Bar Chart Risers in Descending Order

The following request generates a horizontal stacked bar chart with the risers sorted along the x-axis in descending order of y-axis values.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=stacked, CHART-ORIENTATION=horizontal,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"axisList": {
"x1": {"sort": {"order": "descending", "by": "y"}}
}
*END
ENDSTYLE
END
```

The output is shown in the following image.

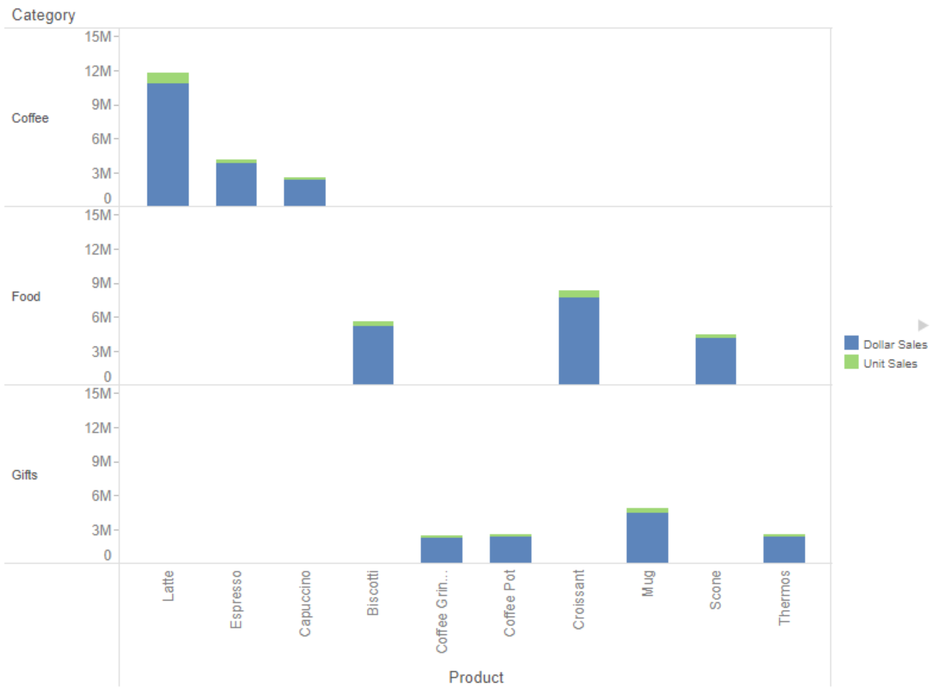


Example: Sorting by a Matrix Row

The following request generates a matrix chart and sorts in descending order by row 0 (zero).

```
GRAPH FILE GGSALES
SUM DOLLARS UNITS
BY CATEGORY
BY PRODUCT
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET VZERO OFF
ON GRAPH SET AUTOFIT ON
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/WFC/Global/Themes/Standard/Default/theme.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=stacked,$
TYPE=DATA, COLUMN=N1, BUCKET=row, $
TYPE=DATA, COLUMN=N2, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N4, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"axisList": {
  "x1": {
    "sort": {
            "order": "descending",
            "by": "matrix0"
            }
       }
           }

*END
ENDSTYLE
END
```

The output is shown in the following image.



Controlling the Appearance of a Numeric Axis Base Line

On a numeric axis, the baseLineStyle properties control the appearance of an axis base line. The base line is parallel to the grid lines and normally passes through the zero value.

Syntax: How to Control the Appearance of a Numeric Axis Base Line

```
"axisname":
{
  "baseLineStyle": {
                 "width": number,
    "color": "string",
    "dash": "string"
  }
}
```

where:

"*axisname*"

Can be one of the axes on the axisList, or xaxis, yaxis, or zaxis.

`"width": number`

Is a number that defines the width of the base line.

`"color": "string"`

Is a string that defines the color of the base line using a color name or numeric specification string.

For information about defining colors, see *Colors and Gradients* on page 85.

`"dash": "string"`

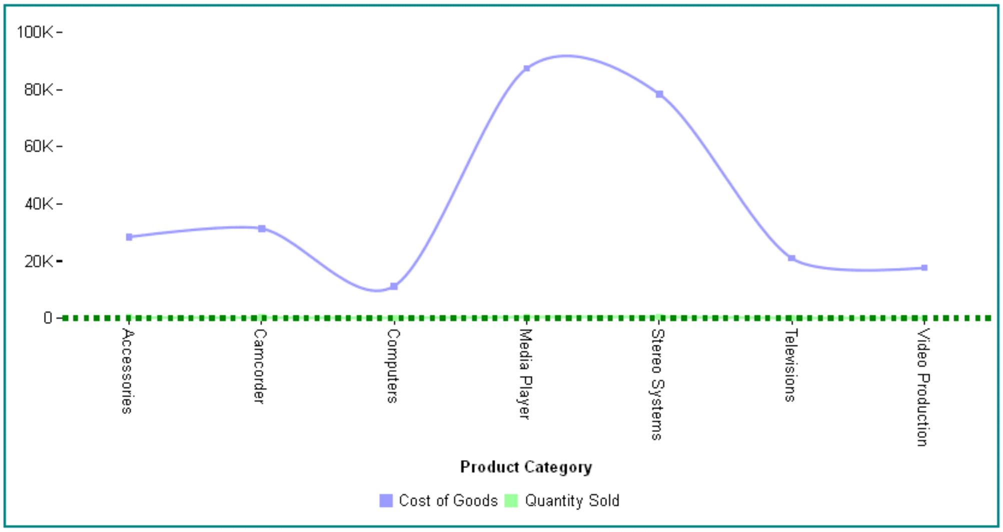
Is a string that defines the dash style. The default value is ""(which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Controlling the Appearance of a Numeric Axis Base Line

The following request against the WF_RETAIL_LITE data source makes the y-axis base line a green dashed line:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US QUANTITY
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "baseLineStyle": {
        "width": 4, "color": "green", "dash": "4 4"}}}
*END
ENDSTYLE
END
```

The output is:



Enabling a Logarithmic Scale

On a numeric axis, the bIsLog property enables or disables a logarithmic scale.

Note: The only log scale base supported is 10.

Syntax: How to Enable a Logarithmic Scale

```
"axisname": {
  "bIsLog": boolean  }
```

where:

"*axisname*"

Specifies the numeric axis for the logarithmic scale. Can be a numeric axis on the axisLIst, or xaxis, yaxis, or zaxis.

"bIsLog": *boolean*

Can be:

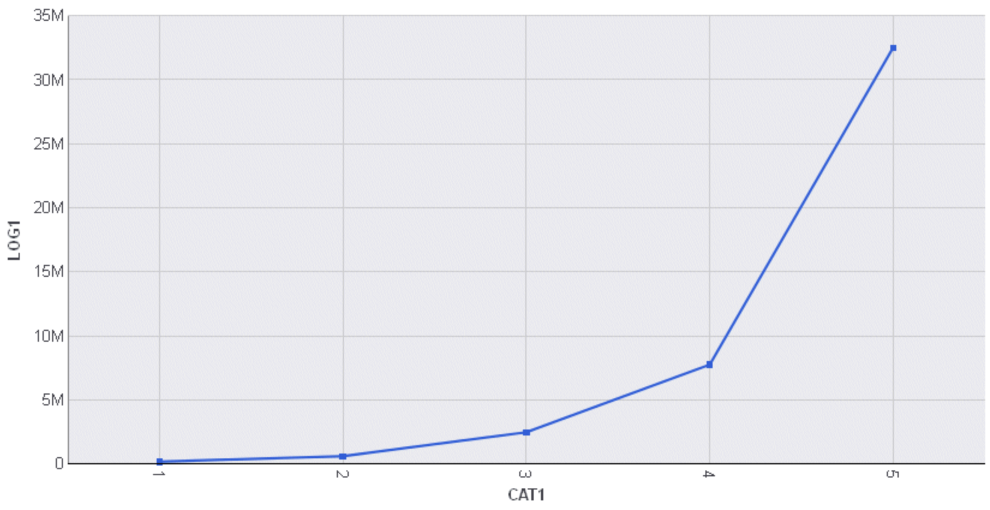
- true, which enables a logarithmic scale on the numeric axis.
- false, which disables a logarithmic scale on the numeric axis. This is the default value.

Example: Enabling a Logarithmic Scale on a Numeric Axis

The following request against the WF_RETAIL_LITE data source generates costs that differ by orders of magnitude, places them into categories based on these costs, and generates a line chart with a numeric y-axis that represents this new cost value:

```
DEFINE FILE WF_RETAIL_LITE
LOG1= IF COGS_US GT 500 AND COGS_US LT 1000 THEN  COGS_US  *10
      ELSE IF COGS_US GE 1000 AND COGS_US LT 2000 THEN COGS_US  * 100
      ELSE IF COGS_US GE 2000 AND COGS_US LT 3000 THEN COGS_US  * 1000
      ELSE IF COGS_US GE 3000  AND COGS_US LT 3300 THEN COGS_US * 10000
      ELSE COGS_US ;
CAT1/I2= IF COGS_US GT 500 AND COGS_US LT 1000 THEN  2
      ELSE IF COGS_US GE 1000 AND COGS_US LT 2000 THEN 3
      ELSE IF COGS_US GE 2000 AND COGS_US LT 3000 THEN 4
      ELSE IF COGS_US GE 3000  AND COGS_US LT 3300 THEN 5
      ELSE 1 ;
END

GRAPH FILE WF_RETAIL_LITE
SUM LOG1
BY CAT1
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```

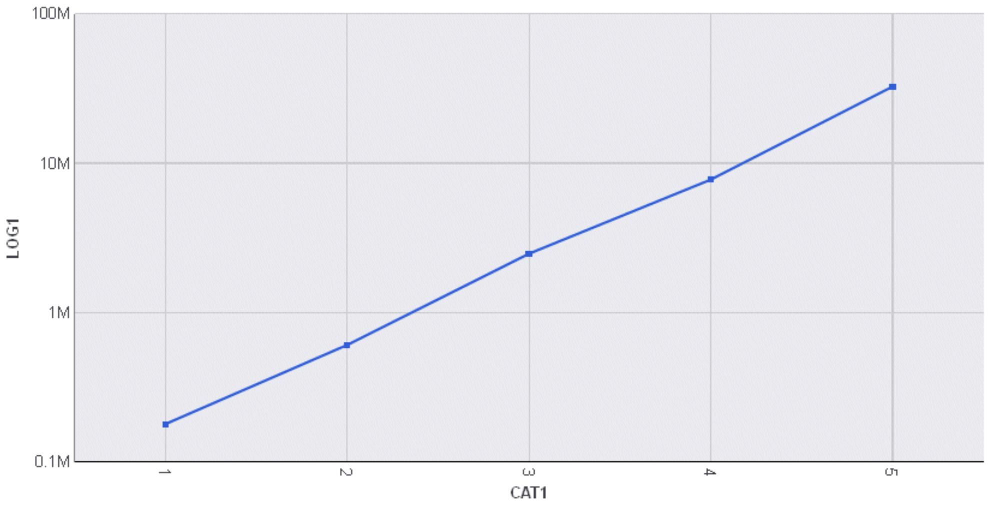
The output is:



The following version of the request enables a logarithmic scale on the y-axis:

```
DEFINE FILE WF_RETAIL_LITE
LOG1= IF COGS_US GT 500 AND COGS_US LT 1000 THEN  COGS_US  *10
      ELSE IF COGS_US GE 1000 AND COGS_US LT 2000 THEN COGS_US  * 100
      ELSE IF COGS_US GE 2000 AND COGS_US LT 3000 THEN COGS_US  * 1000
      ELSE IF COGS_US GE 3000  AND COGS_US LT 3300 THEN COGS_US * 10000
      ELSE COGS_US ;
CAT1/I2= IF COGS_US GT 500 AND COGS_US LT 1000 THEN  2
      ELSE IF COGS_US GE 1000 AND COGS_US LT 2000 THEN 3
      ELSE IF COGS_US GE 2000 AND COGS_US LT 3000 THEN 4
      ELSE IF COGS_US GE 3000  AND COGS_US LT 3300 THEN 5
      ELSE 1 ;
END

GRAPH FILE WF_RETAIL_LITE
SUM LOG1
BY CAT1
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"axisList": {
"y1": {"bIsLog":true}
}
*END
ENDSTYLE
END
```

On the generated chart output, the y-axis scale is logarithmic, making the rates of change between the categories easier to analyze:



Formatting an Axis Body Line

The bodyLineStyle properties control the appearance of an axis body line. The axis body line is perpendicular to the grid lines. It is drawn through all of the tick marks and on top of the chart frame (if visible).

Syntax: How to Format an Axis Body Line

```
"axisname": {
  "bodyLineStyle": {
                 "width": number,
    "color": "string",
    "dash": "string"
  }
}
```

where:

`"axisname"`

Can be an axis on the axisList, or xaxis, yaxis, or zaxis.

`"width": number`

Is a number that defines the width of the axis body line in pixels. The default value is 1.

`"color": "*string*"`

Is a string that defines the color of the axis body using a color name or numeric specification string. The default value is 'transparent'.

For information about defining colors, see *Colors and Gradients* on page 85.

`"dash": "*string*"`

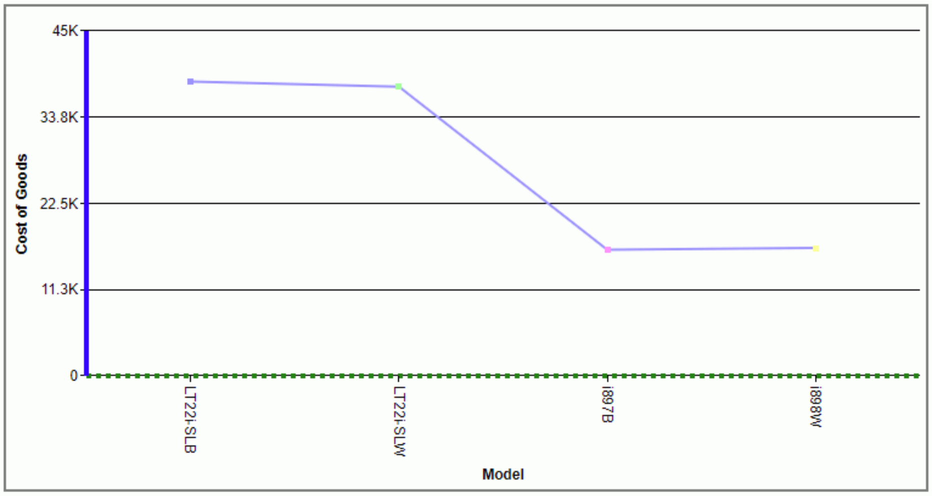
Is a string that defines the dash style. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Formatting an Axis Body Line

The following request generates a vertical line chart with a y-axis body line and base line:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "grey"},
"axisList": {
"y1": {
    "majorGrid": {"visible": true},
    "intervalMode": "count",
    "intervalValue": 5,
    "baseLineStyle": {
        "width": 4, "color": "green", "dash": "4 4"},
    "bodyLineStyle": {"width": 4, "color": "blue"}}
    }
*END
ENDSTYLE
END
```

On the output, the body line is the blue line that is perpendicular to the grid lines. The base line is the green dashed line below the grid lines:



Defining Color Bands

The colorBand properties create blocks of color that span the overall chart area.

Syntax: How to Define Color Bands

```
"axisname":
{
  "colorBands": [
  {
    "start": startval,
    "stop": stopval,
    "color": "color"  }
  ]
},
```

where:

`"axisname"`

Can be any axis on the axisList, or xaxis, yaxis, or zaxis.

`"start": startval`

Is a number or string that defines where on the axis to start the color band.

For a numeric axis:

- ❏ A number must be a value that is visible on the axis.
- ❏ A string must represent a percentage of the length of the axis. The string must be enclosed in double quotation marks and includes a percent symbol (for example, "50%").

For an ordinal axis:

- ❏ a number must be a value between 0 and 1 that represents a percentage of the length of the axis. (for example, .5 will start the color band in the center of the chart).
- ❏ A string must be a group label that is visible on the axis, enclosed in double quotation marks.

`"stop":` *`stopval`*

Is a number or string that defines where on the axis to end the color band.

For a numeric axis:

- ❏ A number must be a value that is visible on the axis.
- ❏ A string must represent a percentage of the length of the axis. The string must be enclosed in double quotation marks and includes a percent symbol (for example, "50%").

For an ordinal axis:

- ❏ a number must be a value between 0 and 1 that represents a percentage of the length of the axis. (for example, .5 will end the color band in the center of the chart).
- ❏ A string must be a group label that is visible on the axis, enclosed in double quotation marks.

`"color": "`*`color`*`"`

Defines the color of the band. Can be:

- ❏ A color defined by a color name or numeric specification string, or a gradient defined by a string.
- ❏ A JSON gradient definition.

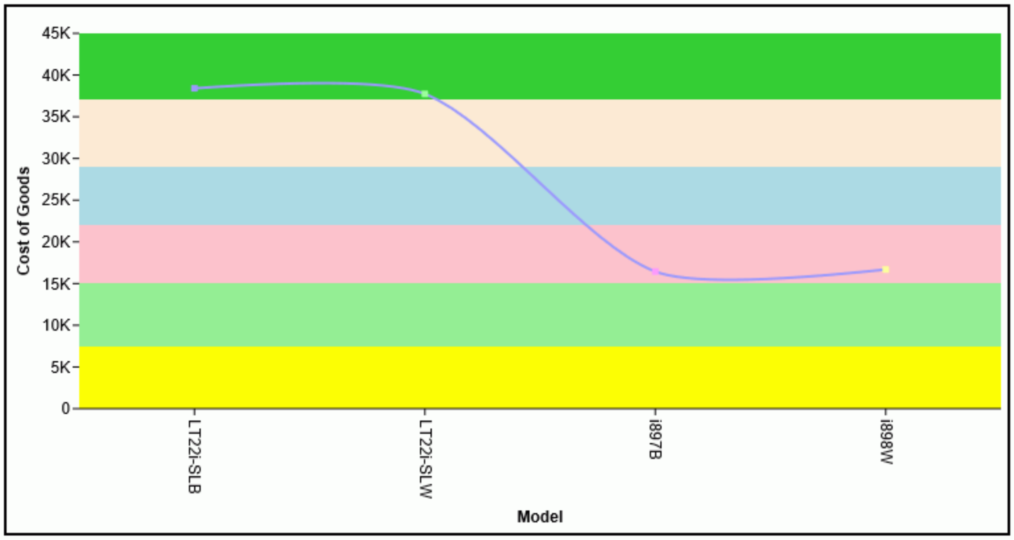
For information about defining colors and gradients, see *Colors and Gradients* on page 85.

Example: Defining Color Bands on a Line Chart

The following request defines color bands on the y-axis. Since the y-axis is numeric, the numbers in the color band definitions are values that appear on the y-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {"colorBands": [
{"start": 0, "stop": 7500, "color": "yellow"},
{"start": 7500, "stop": 15000, "color": "lightgreen"},
{"start": 15000, "stop": 22000, "color": "pink"},
{"start": 22000, "stop": 29000, "color": "lightblue"},
{"start": 29000, "stop": 37000, "color": "antiquewhite"},
{"start": 37000, "stop": 45000, "color": "limegreen"}
    ]}}
*END
ENDSTYLE
END
```

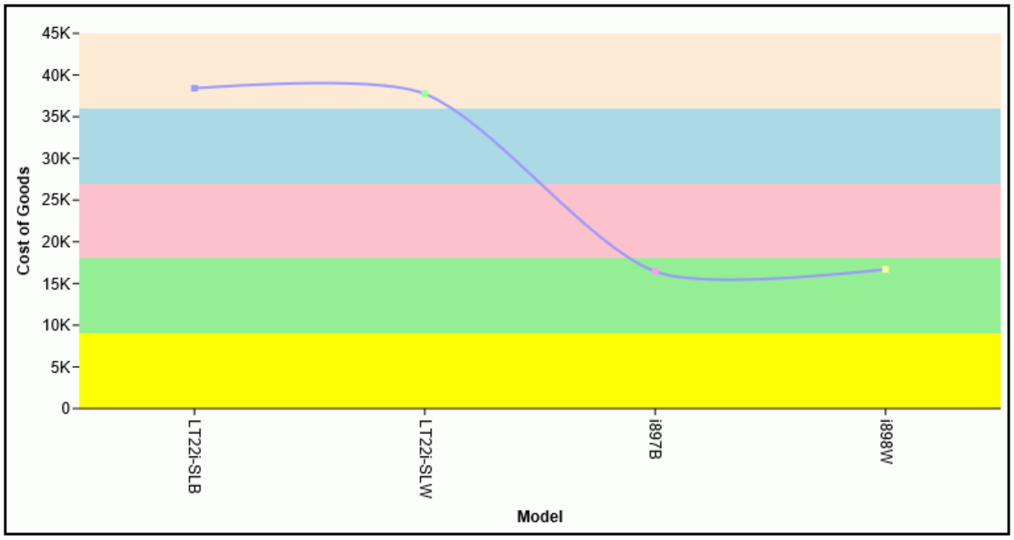
The output is:



The following version of the colorBands object expresses the band start and stop properties as percentages:

```
"y1": {
    "colorBands": [
        {"start": 0, "stop": "20%", "color": "yellow"},
        {"start": "20%", "stop": "40%", "color": "lightgreen"},
        {"start": "40%", "stop": "60%", "color": "pink"},
        {"start": "60%", "stop": "80%", "color": "lightblue"},
        {"start": "80%", "stop": "100%", "color": "antiquewhite"}
        ]}}
```

The output is:

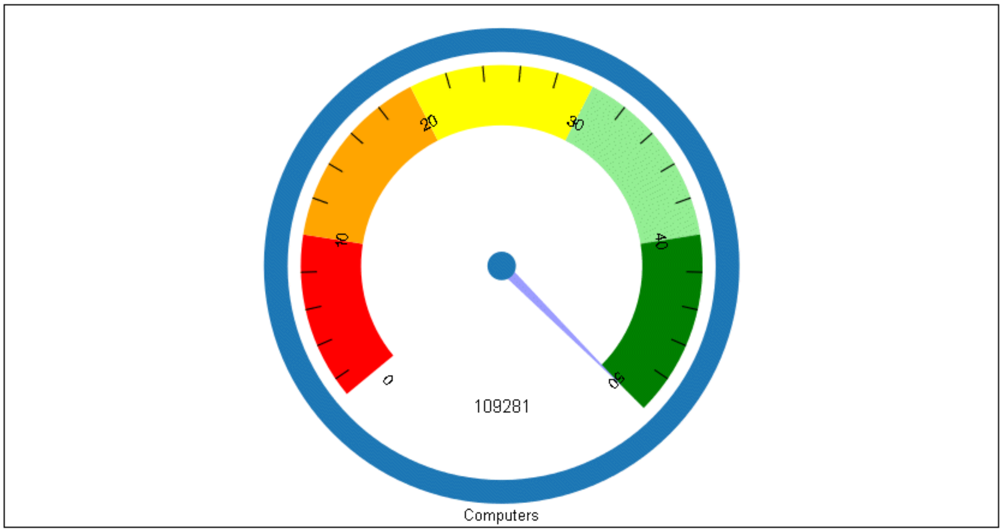


Example: Defining Color Bands on a Gauge Chart

The following request generates a gauge chart and defines the color bands for the gauge axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"yaxis": {
    "min": 0,
    "max": 50,
    "colorBands": [
        {"start": 1, "stop": 10, "color": "red"},
        {"start": 10, "stop": 20, "color": "orange"},
        {"start": 20, "stop": 30, "color": "yellow"},
        {"start": 30, "stop": 40, "color": "lightgreen"},
        {"start": 40, "stop": 50, "color": "green"}
        ]}
*END
ENDSTYLE
END
```

The output is:



Controlling the Number of Major Grid Lines, Ticks, and Labels

On a numeric axis, the intervalMode and intervalValue properties control the number of major grid lines, ticks, and labels.

Syntax: How to Control the Number of Major Grid Lines, Ticks, and Labels

```
"axisname": {
  "intervalMode": "string",
  "intervalValue": number}
```

where:

"*axisname*"

Can be any axis on the axisList, or xaxis, yaxis, or zaxis.

"intervalMode": "*string*"

Is a string that defines the interval mode. Valid values are:

- "count", which indicates that intervalValue will specify the number of major grid lines to draw.
- "interval", which indicates that intervalValue will specify the interval at which major grid lines are drawn.
- "skip", which indicates that intervalValue will specify the number of major grid lines to skip.

The default value is *"undefined"*, which automatically calculates the number and frequency of major grid lines.

"intervalValue": *number*

Is a number that depends on the intervalMode. if intervalMode is "count", set this property to the number of major grid lines to draw. If intervalMode is "interval", set this property to the interval at which major grid lines are drawn. If intervalMode is "skip", set this property to the number of grid lines to skip. The default value is *"undefined"* which automatically calculates the number and frequency of major grid lines.

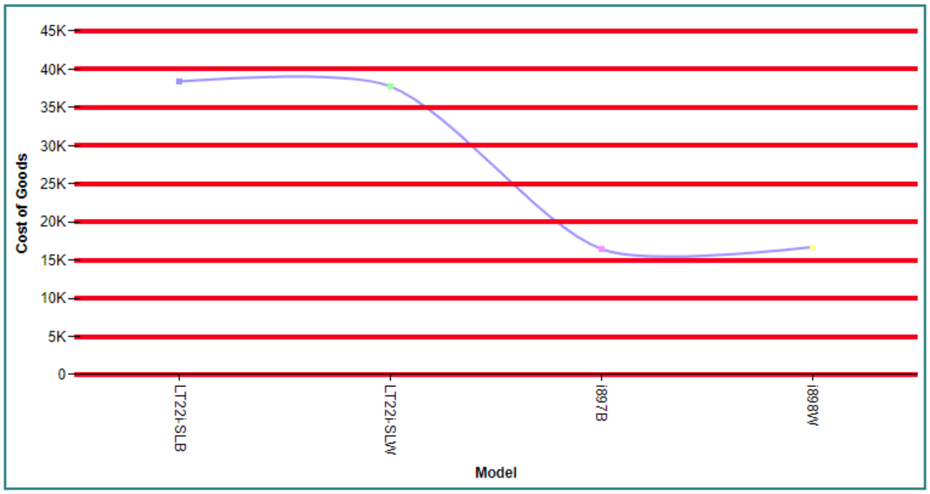
Note: The first and last grid lines are always drawn regardless of the intervalMode or intervalValue.

Example: Setting intervalMode and intervalValue

The following request generates a vertical line chart with major grid lines calculated automatically:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList":
 {"y1": {
    "intervalMode": "undefined",
    "intervalValue": "undefined",
    "majorGrid": {
        "visible": true,
        "aboveRisers": true,
        "lineStyle": {"color": "red"}
        }}}
*END
ENDSTYLE
END
```

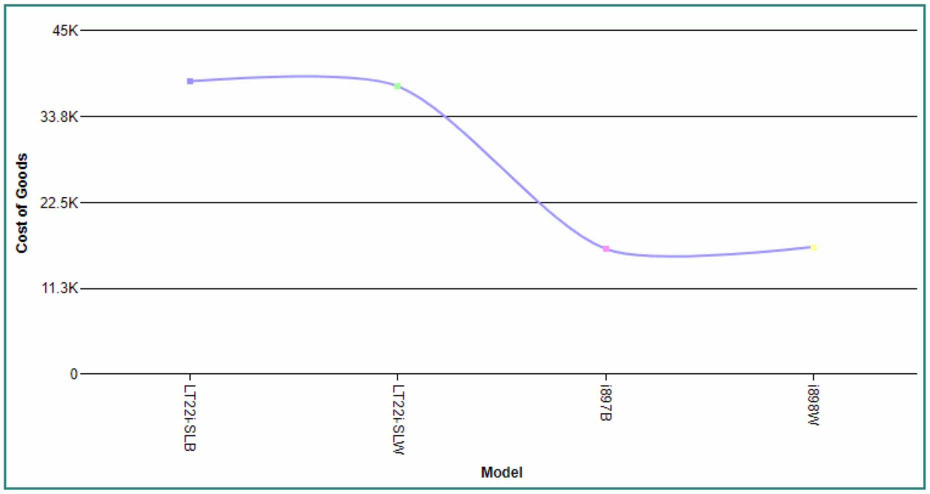
The output is:



The following version of the request draws 5 major grid lines:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "majorGrid": {"visible": true},
    "intervalMode": "count",
    "intervalValue": 5}}
*END
ENDSTYLE
END
```

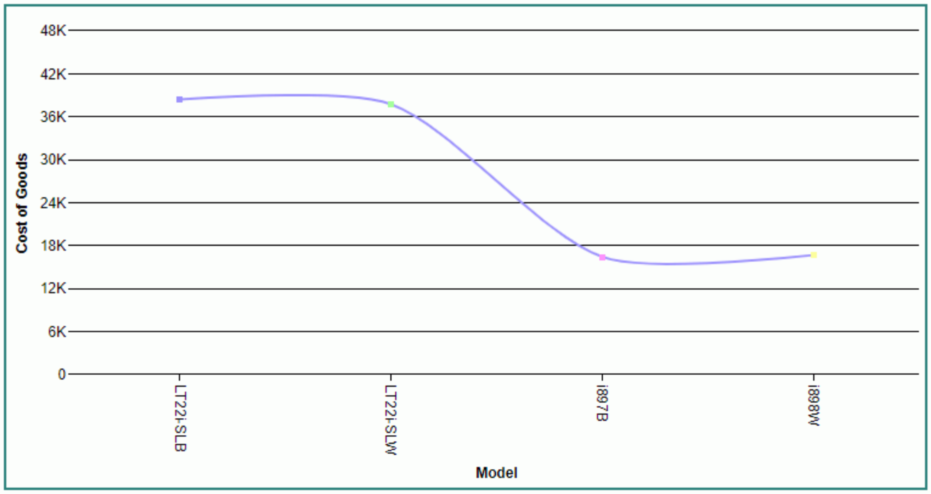
The output is:



The following version of the request draws major grid lines at intervals of six thousand:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "majorGrid": {"visible": true},
    "intervalMode": "interval",
    "intervalValue": 6000}}
*END
ENDSTYLE
END
```

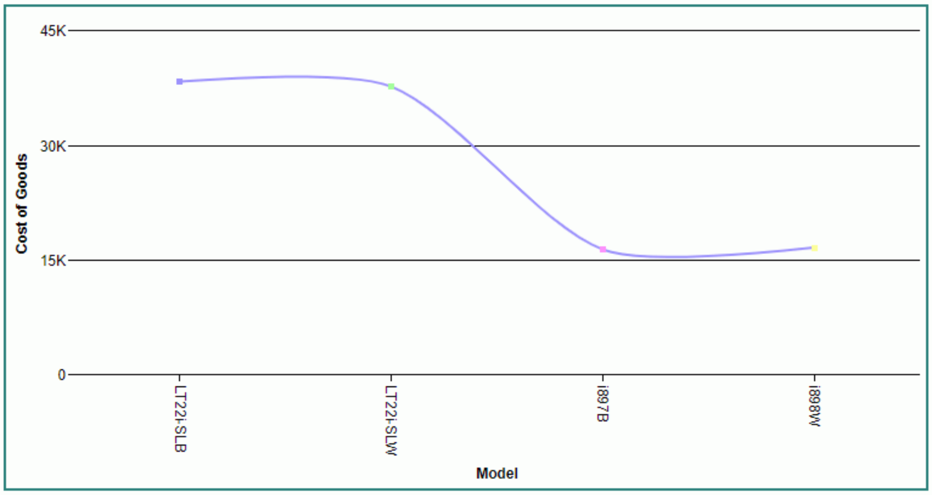
The output is:



The following version skips every other automatic major grid line:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "majorGrid": {"visible": true},
    "intervalMode": "skip",
    "intervalValue": 1
    }}
*END
ENDSTYLE
END
```

The output is:



Synchronizing Y2-Axis Major Grid Lines With Y-Axis Major Grid Lines

By default, when there is a y1-axis and a y2-axis, each generates its own major grid lines. The number of scale labels for each axis depends on the range of its individual data values. If you want to force the y2-axis to use the same number of scale labels as the y1-axis, set the y2-axis majorGrid:lockToY1 property to *true*. The y2-axis will then share the y-axis major grid lines.

Syntax: How to Synchronize the Y-Axis and Y2-Axis Major Grid Lines

```
"axisname": {
        "majorGrid":
        {
            "lockToY1":boolean
  }
     }
```

where:

`"axisname"`

Can be a y2 on the axisList, or y2axis.

`"lockToY1": boolean`

Controls whether the y2-axis will be drawn with the same number of scale labels as the y1-axis so that they can share the same major grid lines. Valid values are:

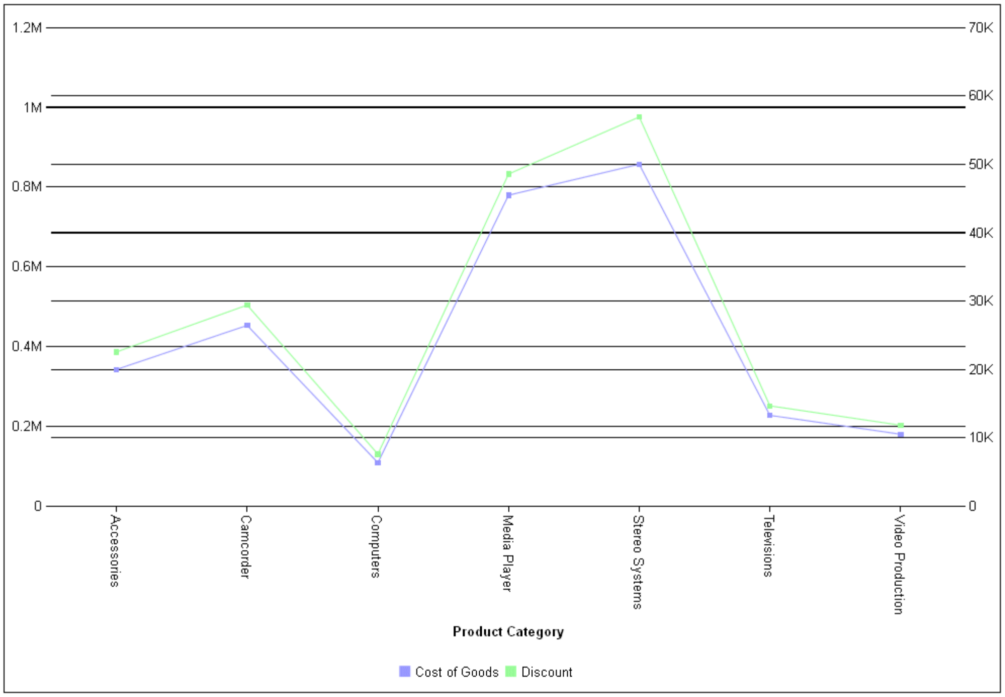
- false, which does not make the number of scale labels the same for the y1-axis and y2-axis. This is the default value.
- true, which makes the number of scale labels for the y2-axis the same as the number for the y1-axis, and causes them to share the y1-axis major grid lines.

Example: Using the Same Major Grid Lines for the Y-Axis and Y2-Axis

The following request generates a vertical line chart with a y1-axis and a y2-axis. It does not specify that they should be drawn to use the number of scale labels:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series": 1, "yAxisAssignment": 2}],
"axisList": {
"y1": {"majorGrid": {"visible": true}},
"y2": {"majorGrid": {"visible": true}}
}
*END
ENDSTYLE
END
```

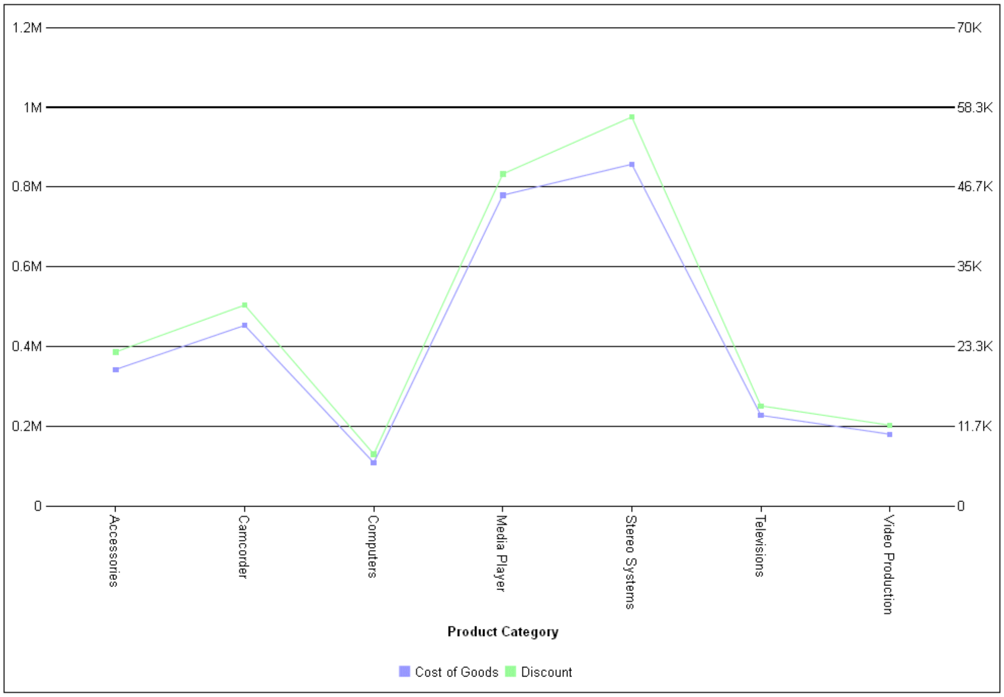
The output is shown in the following image. The y-axis has seven major grid lines, and the y2-axis has eight:



The following version of the request specifies *true* for the lockToY1 property of the y2-axis majorGrid object:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series": 1, "yAxisAssignment": 2}],
"axisList": {
"y1": {"majorGrid": {"visible": true}},
"y2": {"majorGrid": {    "visible": true,     "lockToY1": true}}
}
*END
ENDSTYLE
END
```

The output is shown in the following image. The y2-axis has been adjusted to use the same number of labels as the y-axis. Both axes are drawn to share the seven original y-axis major grid lines:



Controlling the Direction of a Numeric Axis

The invert property controls the direction of a numeric axis, ascending or descending.

Syntax: How to Control the Direction of a Numeric Axis

```
"axisname": {
  "invert": boolean  }
```

where:

`"axisname"`

Can be any axis on the axisList, or xaxis, yaxis, or y2axis.

`"invert": boolean`

Specifies whether the axis should be descending. Can be:

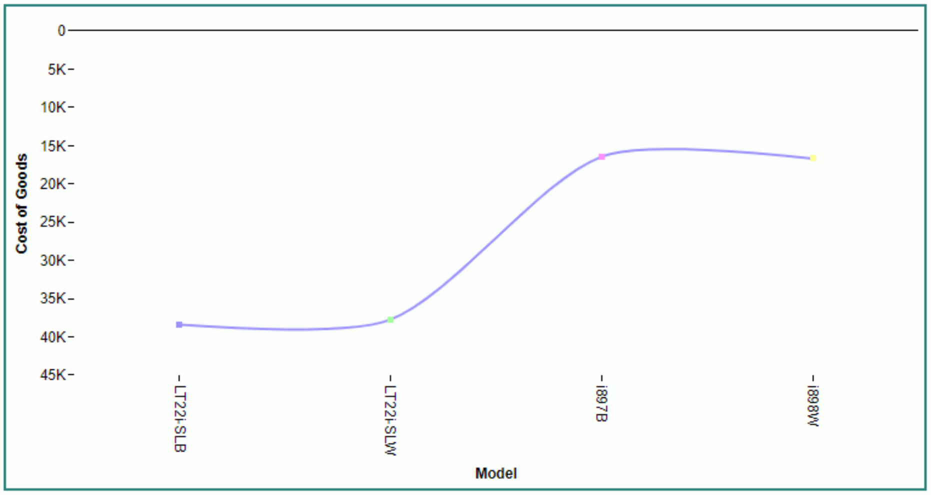
- true, which draws a descending axis.
- false, which draws an ascending axis. This is the default value.

Example: Controlling the Direction of a Numeric Axis

The following request draws a descending y-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "invert": true}
    }
*END
ENDSTYLE
END
```

The output is:



Formatting the Axis Labels

The labels properties control the visibility and format of the axis labels.

Note: HTML5 charts do not support showing axis labels on BOTH right and left axes at the same time (only LEFT or RIGHT is supported).

Syntax: **How to Format Axis Labels**

```
"axisname": {
  "labels": {
    "visible": boolean,
    "font": "string",
    "color": "string",
    "excludeMin": boolean,
    "excludeMax": boolean,
    "nestingConcatSymbol": "string",
    "nestingLineStyle": {
            "width": number,
            "color": "string",
            "dash": "string"
             },
    "rotation": number  }
}
```

where:

`"axisname"`

Can be any axis on the axisList, or xaxis, yaxis, y2axis, or zaxis.

`"visible": boolean`

Controls the visibility of the axis labels. Valid values are:

- true, which makes the axis labels visible. This is the default value.
- false, which makes the axis labels not visible.

`"font": "string"`

Is a string that defines the size, style, and typeface of the axis labels. The default value is "7.5pt Sans-Serif".

`"color": "string"`

Is a string that defines color of the axis labels using a color name or numeric specification string. The default value is "black".

For information about defining colors, see *Colors and Gradients* on page 85.

`"excludeMin": boolean`

Controls the visibility of the label for the minimum value. Valid values are:

- true, which makes the minimum label not visible.
- false, which makes the minimum label visible. This is the default value.
- "auto", which lets the chart engine control whether to draw the minimum label. It will not draw the label if it is too close to or overlaps another label.

`"excludeMax": boolean`

Controls the visibility of the label for the maximum value. Valid values are:

- true, which makes the maximum label not visible.
- false, which makes the maximum label visible. This is the default value.
- "auto", which lets the chart engine control whether to draw the maximum label. It will not draw the label if it is too close to or overlaps another label.

`"nestingConcatSymbol": "string"`

Controls the generation of nested x-axis labels when multiple hierarchical sort fields are assigned to the x-axis.

By assigning multiple sort fields to the x-axis, you can create two different effects (layouts). The x-axis will be drawn with either a nested layout, where the hierarchy of the sorts is presented on the x-axis, or a concatenated layout, where each label contains its fully qualified sort path, separated by a concatenation symbol.

If this property is not set, or is set to null (the default value) or undefined, nested x-axis labels are generated. If this property is set to a string (for example, ":"), the x-axis labels are concatenated, with the specified string placed between the sort field values.

`"nestingLineStyle"`

Controls the style of the line connecting the nested groups.

`"width": number`

Is the width of the line in pixels. The default value is 1.

`"color": "string"`

Is a color specification string that defines the color of the line. The default value is 'black'.

`"dash": "string"`

Is a string that defines the dash style. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

`"rotation": number`

Defines the rotation of axis labels, in degrees. Valid values are 0 (no rotation), 45, 90, 135, 180, 270, or undefined. The default value is undefined. Any value other than undefined will disable automatic layout of axis labels. For information, see *Controlling Automatic Layout of Ordinal Axis Labels*.

Note:

- If you choose 45-degree rotated labels, the labels may appear clipped if you have also activated a scrolling x-axis. A possible workaround is to use rotated 90-degree labels instead.
- Y-axis labels cannot be rotated.

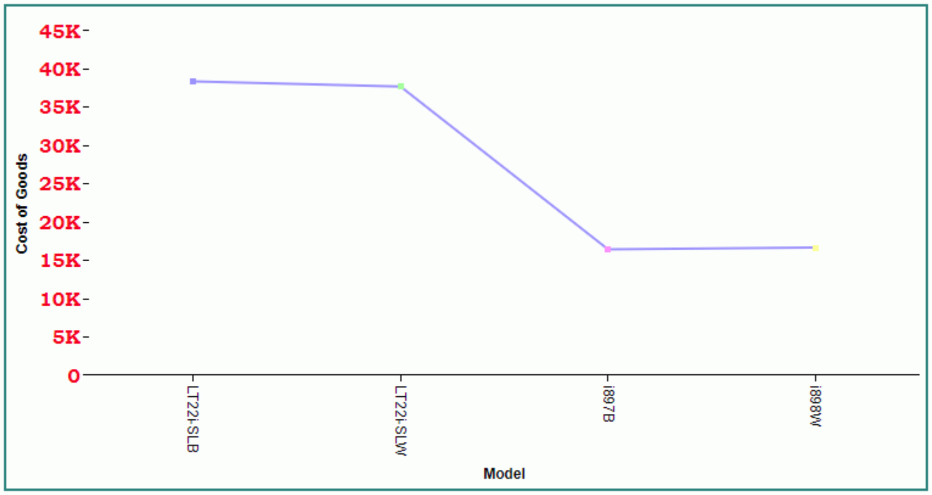
Note: HTML5 charts do not support automatic wrapping of x-axis labels.

Example: Formatting Axis Labels

The following request generates a vertical line chart and makes the y-axis labels bold with a 12pt Bookman Old Style font, in red:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {"labels": {
    "color": "red", "font": "bold 12pt 'Bookman Old Style'"}}}
*END
ENDSTYLE
END
```

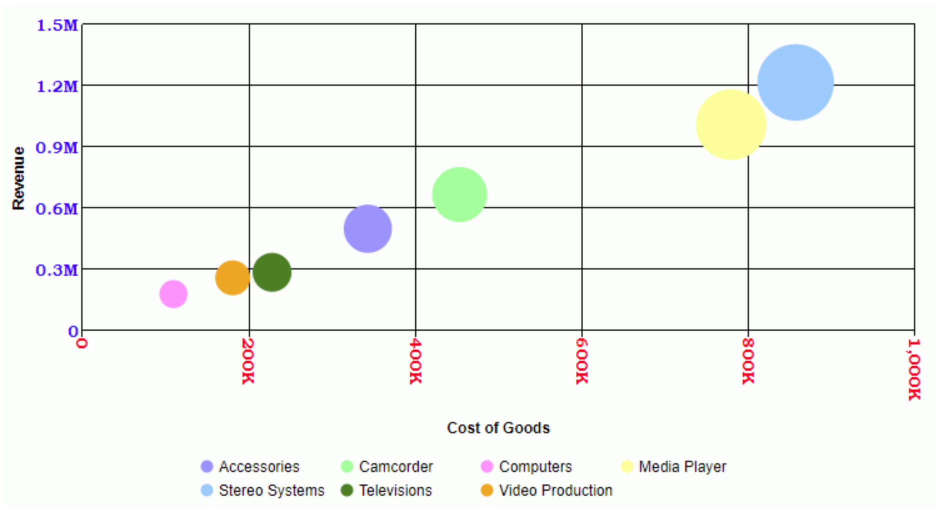
The output is:



The following request generates a bubble chart and formats the axes labels to have the font "bold 10pt 'Bookman Old Style'". The y-axis labels are blue, and the x-axis labels are red:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 0},
"series": [{"series": "all", "marker": {"shape": "circle"}}],
"axisList": {
"x1": {
    "labels": {"font": "bold 10pt 'Bookman Old Style'", "color": "red"}},
"y1": {
    "labels": {"font": "bold 10pt 'Bookman Old Style'", "color":
"blue"}}  }
*END
ENDSTYLE
END
```

The output is:

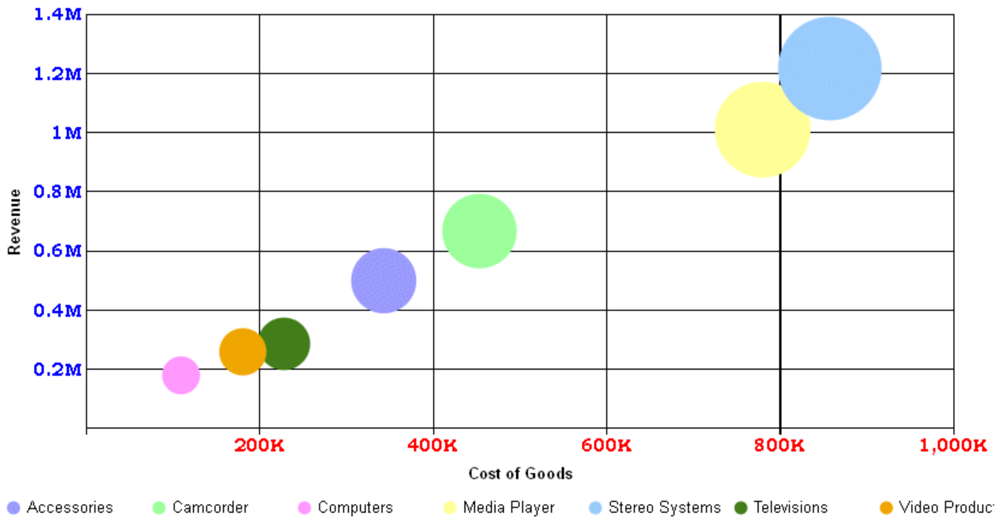


Example: Excluding the Minimum Axis Labels

The following request does not draw the minimum axis labels (0,0):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 0},
"series": [{"series": "all", "marker": {"shape": "circle"}}],
"axisList": {
"x1": {"labels": {
    "font": "bold 10pt Bookman Old Style",
    "color": "red", "excludeMin": true}},
"y1": {"labels": {
    "font": "bold 10pt Bookman Old Style",
    "color": "blue", "excludeMin": true}}}
*END
ENDSTYLE
END
```

The output is:



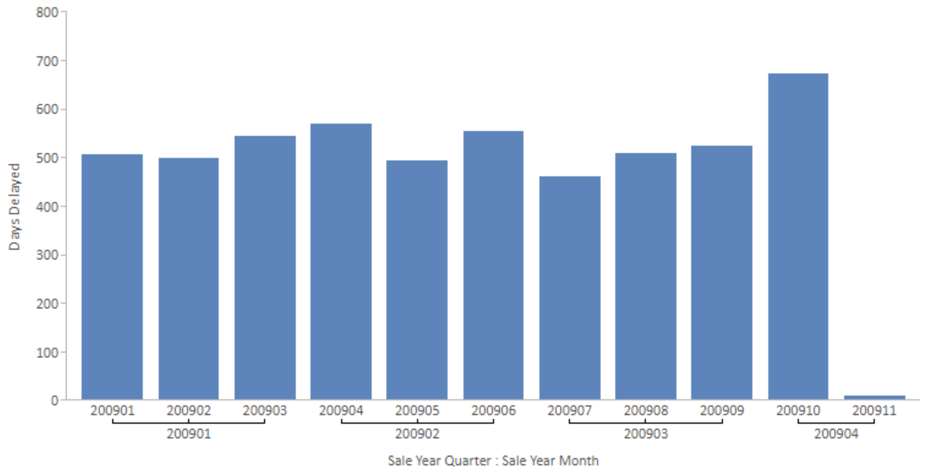
Example: **Controlling Nesting of X-Axis Labels**

By assigning multiple sort fields to the x-axis, you can create two different effects (layouts). The x-axis will be drawn with either a nested layout, where the hierarchy of the sorts is presented on the x-axis, or a concatenated layout, where each label contains its fully qualified sort path, separated by a concatenation symbol.

The following request generates nested x-axis labels.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY TIME_YEARQTR
BY TIME_YEARMTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=DAYSDELAYED, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_YEARQTR, BUCKET=x-axis, $
TYPE=DATA, COLUMN=TIME_YEARMTH, BUCKET=x-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"axisList": {
"x1": {"labels": {"nestingConcatSymbol": null}}}
*END
ENDSTYLE
END
```

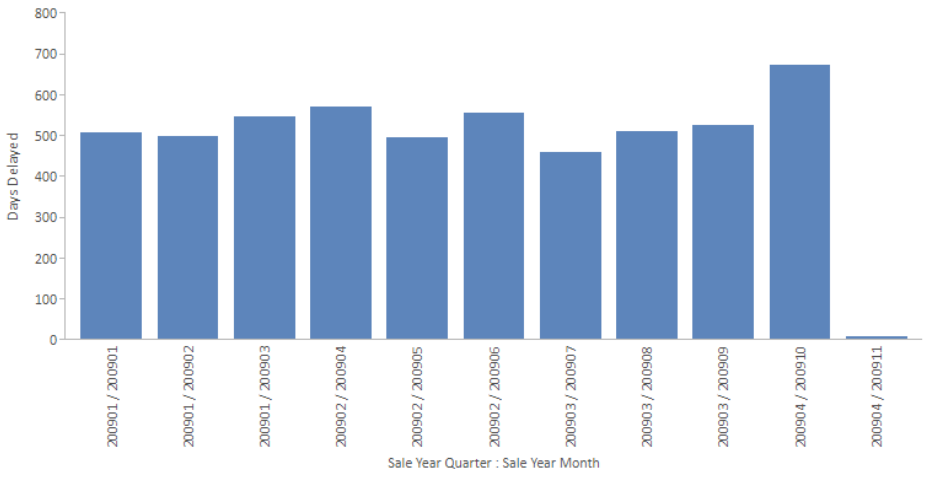
The output is shown in the following image.



The following version of the request makes the concatenation string a slash surrounded by spaces (" / ").

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY TIME_YEARQTR
BY TIME_YEARMTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=DAYSDELAYED, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_YEARQTR, BUCKET=x-axis, $
TYPE=DATA, COLUMN=TIME_YEARMTH, BUCKET=x-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"axisList": {
"x1": {"labels": {"nestingConcatSymbol": " / "}}}
*END
ENDSTYLE
END
```

The output is shown in the following image.

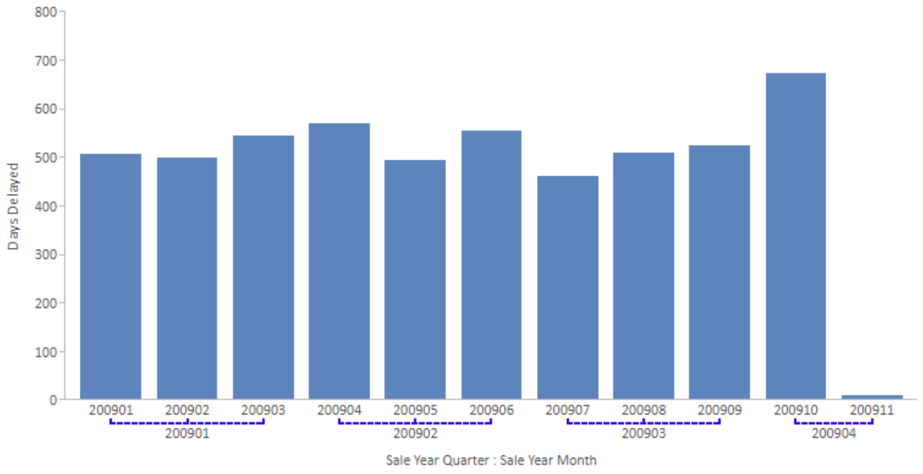


Example: Styling Nested Label Connector Lines

The following request makes the nested label connector lines 2 pixels wide, blue, and dashed.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY TIME_YEARQTR
BY TIME_YEARMTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=DAYSDELAYED, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_YEARQTR, BUCKET=x-axis, $
TYPE=DATA, COLUMN=TIME_YEARMTH, BUCKET=x-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
*GRAPH_JS
"axisList": {
 "x1": {"labels": {"nestingConcatSymbol": null,
 "nestingLineStyle": {
         "width": 2,
         "color": "blue",
         "dash": "3 3"
        }}}
        }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Formatting Major Grid Lines and Tick Marks

The majorGrid properties control the visibility and format of major grid lines and tick marks.

Syntax: How to Format Major Grid Lines and Tick Marks

```
"axisname": {
"majorGrid": {
  "visible": boolean,
  "aboveRisers": boolean,
  "lineStyle": {
    "width": number,
    "color": "string",
    "dash": "string"
   },
  "ticks": {
    "length": number,
    "visible": boolean,
    "style": "string",
    "lineStyle": {
      "width": number,
      "color": "string"}
    }
  }
}
```

where;

"*axisname*"

Can be any axis on the axisList, or xaxis, yaxis, or zaxis.

`"majorGrid":`

Defines the major grid line properties.

`"visible":` *`boolean`*

Controls the visibility of the major grid lines. The default depends on the type of chart being generated. Valid values are:

- ❏ true, which draws the major grid lines.
- ❏ false, which does not draw the major grid lines.

`"aboveRisers":` *`boolean`*

Controls whether the grid lines are drawn in front of or behind the risers. Valid values are:

- ❏ true, which draws major grid lines on top of the risers.
- ❏ false, which draws major grid lines behind the risers. This is the default value.

`"lineStyle":`

Defines the style of the major grid lines.

`"width":` *`number`*

Is a number that defines the width of major grid lines in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of major grid lines using a color name or numeric specification string. The default value is "rgba(255, 255, 255, 0.3)".

For information about defining colors, see *Colors and Gradients* on page 85.

`"dash": "`*`string`*`"`

Is a string that defines the dash style. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

`"ticks":`

Defines the tick marks for the major grid lines.

`"length":` *`number`*

Is a number that defines the length of tick marks in pixels. The default value is 5.

`"visible": boolean`

Controls the visibility of tick marks. Valid values are

- ❑ <u>true</u>, which draws the tick marks. This is the default value.
- ❑ false, which does not draw the tick marks.

`"style": "string"`

Is a string that defines the tick mark style. The default depends on the type of chart being generated. Valid values are:

- ❑ "inner", which places the tick marks inside the major grid lines. This is the default value.
- ❑ "outer", which places the tick marks outside of the major grid lines.
- ❑ "span", which places half of each tick mark outside the major grid lines, and half of the tick mark inside the major grid lines.

`"lineStyle":`

Defines the line style of the major grid line tick marks.

`"width": number`

Is a number that defines the width of the tick marks in pixels. The default value is 1.

`"color": "string"`

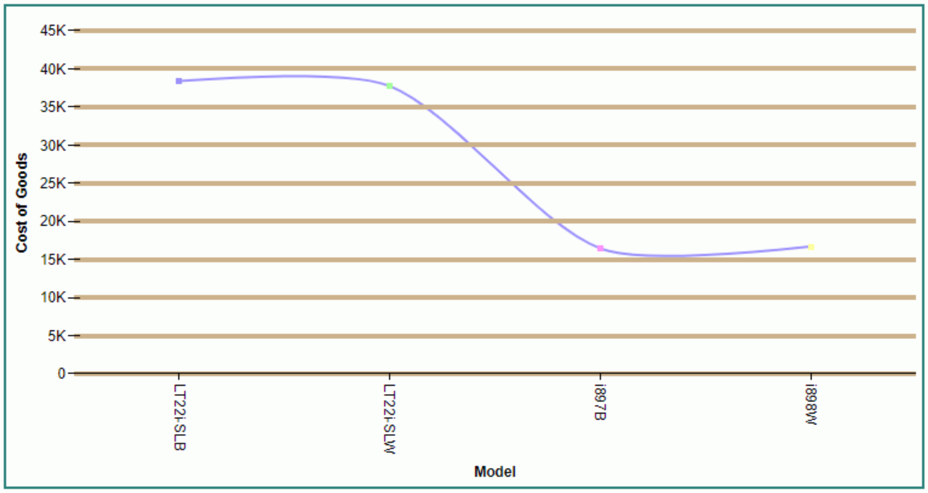
Is a string that defines the color of tick marks using a color name or numeric specification string. The default value is "black".

Example: **Formatting Major Grid Lines and Tick Marks**

The following request makes the major grid lines visible, 4 pixels wide, in front of the risers, and tan. The tick mark style is 'span', so they are both outside and inside the grid lines:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "title": {"visible": true},
    "majorGrid": {
        "visible": true,
        "aboveRisers": true,
        "lineStyle": {"width": 4, "color": "tan"},
        "ticks": {"style": "span"}
        }}}
*END
ENDSTYLE
END
```

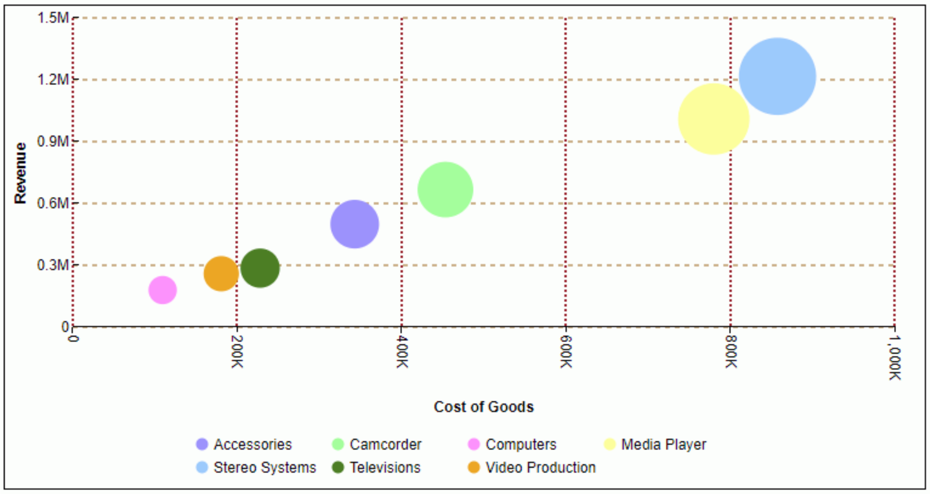
The output is:



The following request formats the x-axis and y-axis major grid lines in a bubble chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series": "all", "marker": {"shape": "circle"}}],
"axisList": {
"x1": {
    "title": {"visible": true},
    "majorGrid": {"lineStyle": {
        "width": 2, "color": "brown", "dash": "2 2"}}},
"y1": {
    "title": {"visible": true},
    "majorGrid": {"lineStyle": {
        "width": 2, "color": "tan", "dash": "4 4"}}}}
*END
ENDSTYLE
END
```

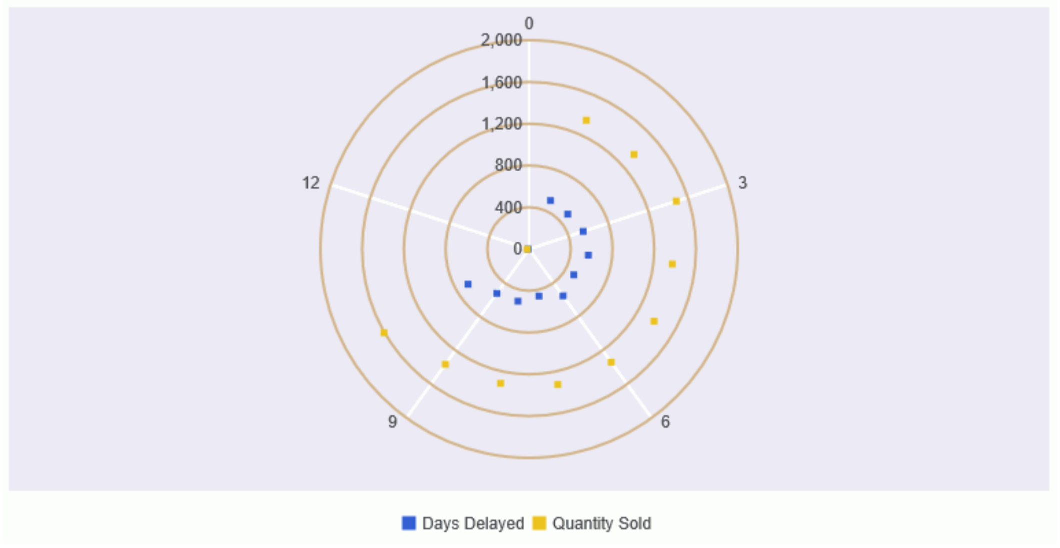
The output is:



The following request generates a polar chart in which the y-axis major grid lines are tan and the x-axis major grid lines are white:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
ACROSS TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH POLAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"polarProperties": {
    "straightGridLines": false, "extrudeAxisLabels": false},
"axisList":
"x1": {
    "title": {"visible": true},
    "majorGrid": {
        "aboveRisers": false,
        "lineStyle": {"width": 2, "color": "white"}}},
"y1": {
    "title": {"visible": true},
    "majorGrid": {
        "aboveRisers": false,
        "lineStyle": {"width": 2, "color": "tan"}}}
}
*END
ENDSTYLE
END
```

The output is:

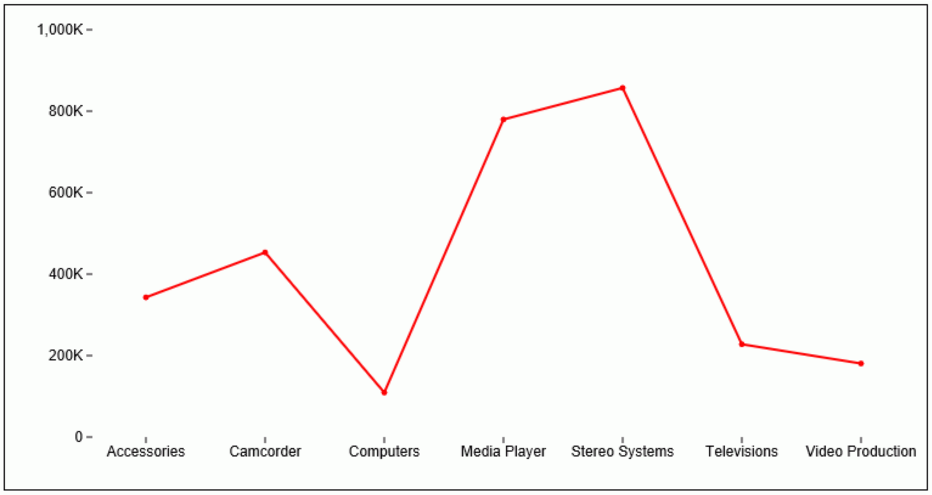


Example: Removing Axis Lines

You can use the majorGrid:{visible:false} property to remove an axis body line and all grid lines for that axis. For a numeric axis, you must also remove the baseline, using the baseLineStyle: {width:0} property. For example:

```
GRAPH FILE wf_retail_lite
SUM COGS_US AS ' '
BY PRODUCT_CATEGORY AS ' '
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis,$
*GRAPH_JS
"yaxis":{"baseLineStyle": {"width": 0},
         "majorGrid": {"visible":false}},
"xaxis":{"majorGrid": {"visible":false}}
*END
ENDSTYLE
 END
```

The output is shown in the following image.



Defining the Minimum and Maximum Values on a Numeric Axis

On a numeric axis, the min and max properties define the minimum and maximum values to draw on the axis.

Syntax: How to Define the Minimum and Maximum Values for a Numeric Axis

```
"axisname": {
  "min": number,
  "max": number}
```

where:

axisname

Can be any axis on the axisList, or xaxis or yaxis.

"min": *number*

Is the minimum value to draw on a numeric axis. The default value is undefined, which automatically calculates the minimum value based on values in the data set.

"max": *number*

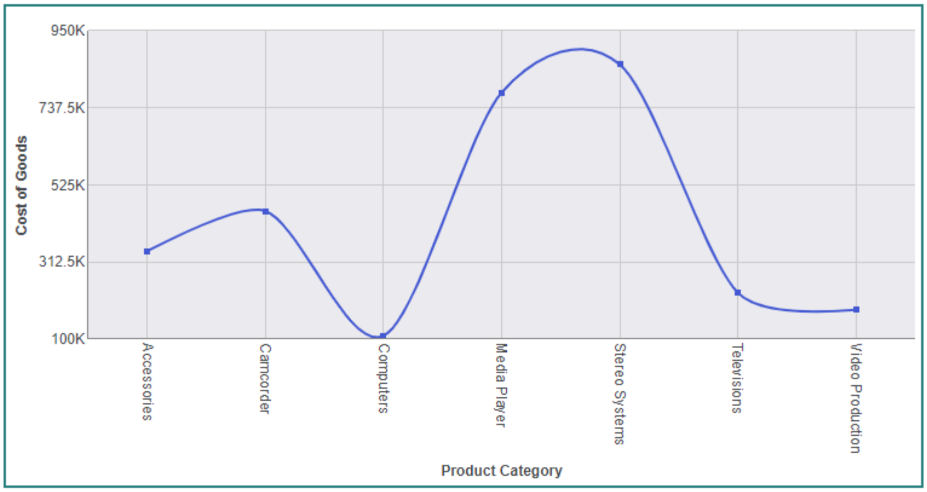
Is the maximum value to draw on a numeric axis. The default value is undefined, which automatically calculates the maximum value based on values in the data set.

Example: Specifying the Minimum and Maximum Values to Display on a Numeric Axis

The following request generates a vertical line chart and makes the minimum value shown on the y-axis 100,000, and the maximum value 950,000:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {"min": 100000, "max": 950000}}
*END
ENDSTYLE
END
```

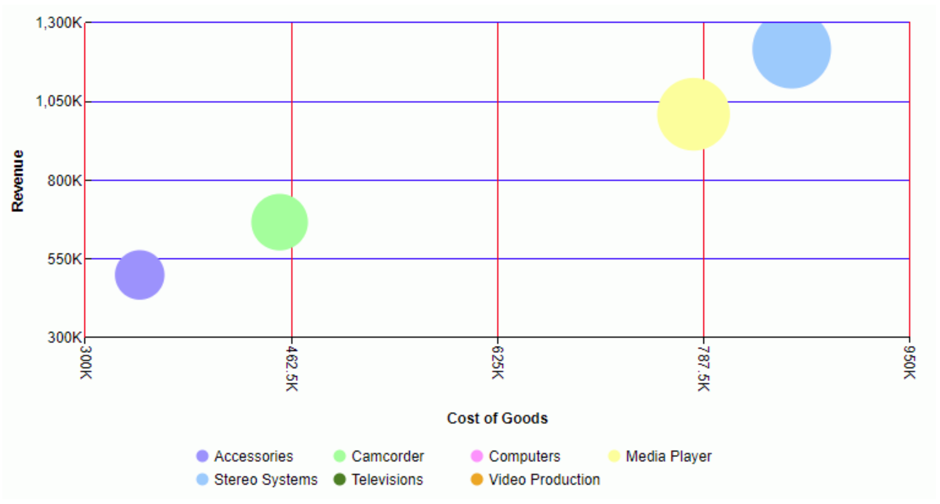
The output is:



The following request generates a bubble chart and sets the maximum and minimum values to draw on each axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 0},
"series": [{"series": "all", "marker": {"shape": "circle"}}],
"axisList": {
"y1": {
    "min": 300000, "max": 1300000,
    "majorGrid": {"lineStyle": {"width": 1, "color": "blue"}}
    },
"x1": {
    "min": 300000, "max": 950000,
    "majorGrid": {"lineStyle": {"width": 1, "color": "red"}}
    }}
*END
ENDSTYLE
END
```

The output is:



Replacing the Minimum and Maximum Labels on a Numeric Axis

You can use the axis replaceMin and replaceMax properties to replace the labels for the minimum and maximum points on a numeric axis.

Syntax: How to Replace the Minimum and Maximum Labels on a Numeric Axis

```
"axisname":
 {

    "labels": {
               "replaceMin": "string",
               "replaceMax": "string"
           }
 }
```

where:

`axisname`

Is a numeric axis name on the axisList, or xaxis or yaxis.

`"replaceMin": "string"`

Can be any string. The default is *undefined*, which uses the calculated minimum label.

`"replaceMax": "string"`

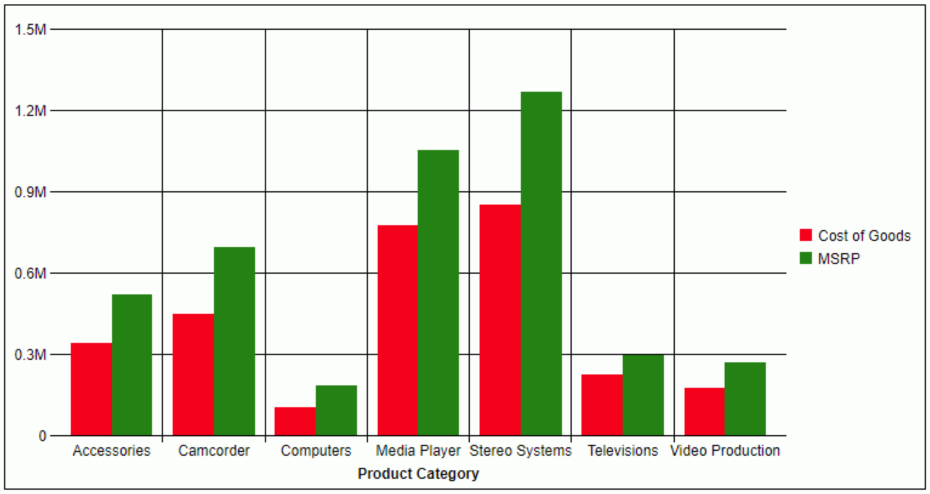
Can be any string. The default is *undefined*, which uses the calculated maximum label.

Example: Replacing the Minimum and Maximum Labels on the Y-Axis

The following request generates a bar chart with the default label values on the y-axis.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=n1, bucket=x-axis,$
type=data, column=n2, bucket=y-axis,$
type=data, column=n3, bucket=y-axis,$
ENDSTYLE
END
```

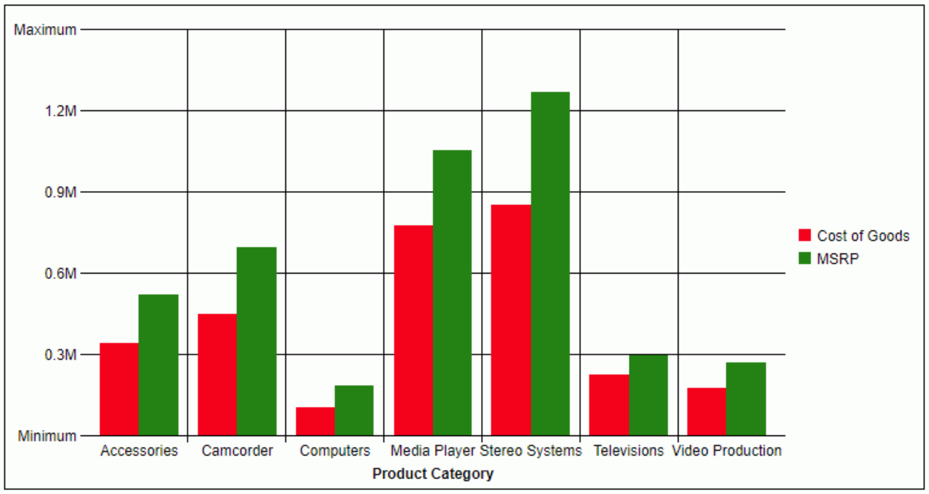
The output is shown in the following image.



The following version of the request replaces the minimum y-axis label with the word Minimum and the maximum y-axis label with the word Maximum.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=n1, bucket=x-axis,$
type=data, column=n2, bucket=y-axis,$
type=data, column=n3, bucket=y-axis,$
*GRAPH_JS
"axisList": {
"y1":
  {"labels":
    {
     "replaceMin": "Minimum",
     "replaceMax": "Maximum"
     }
  }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Including Zero on a Numeric Axis for a Bar Chart

In a bar or stacked bar chart, if the data does not include any values that are close to zero, the numeric axis may not include zero. This can make it difficult to interpret the output correctly. You can force the inclusion of zero on the axis using the mustIncludeZero axis property.

Syntax: How to Include Zero on a Numeric Axis for a Bar Chart

```
"axisname": {"mustIncludeZero": boolean}
```

where:

`"axisname"`
Is the name of a numeric axis on the axisList, or xaxis or yaxis.

`"mustIncludeZero": boolean`
Can be one of the following values:

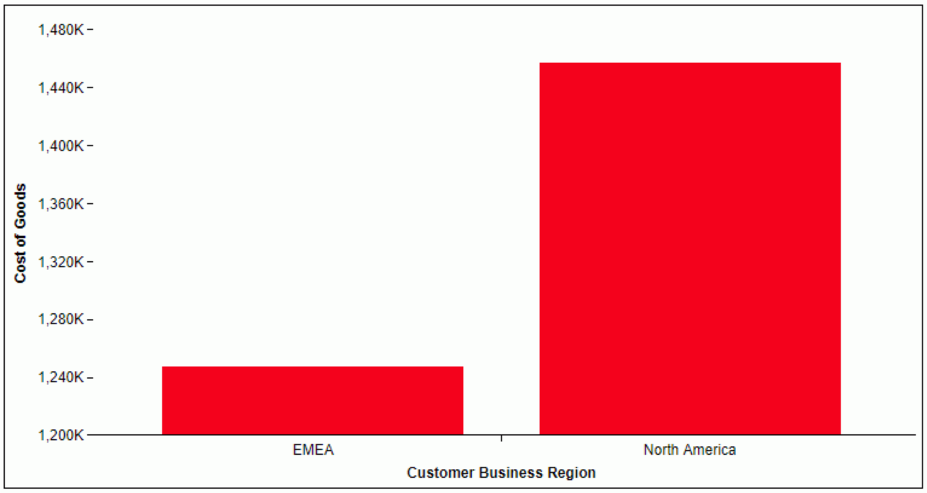
- true, which forces the numeric axis to include the zero value. This is the default value.
- false, which does not force the numeric axis to include the zero value.

Example: Forcing a Numeric Axis to Include Zero

The following request generates a bar chart with the y-axis mustIncludeZero property set to *false*.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania' OR 'South America'
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=y-axis,$
type=data, column= business_region, bucket=x-axis,$
*GRAPH_JS
"axisList": {
"x1": {
    "majorGrid": {"visible": false}
},
"y1": {
    "majorGrid": {"visible": false},
    "mustIncludeZero": false
}
}
END
```

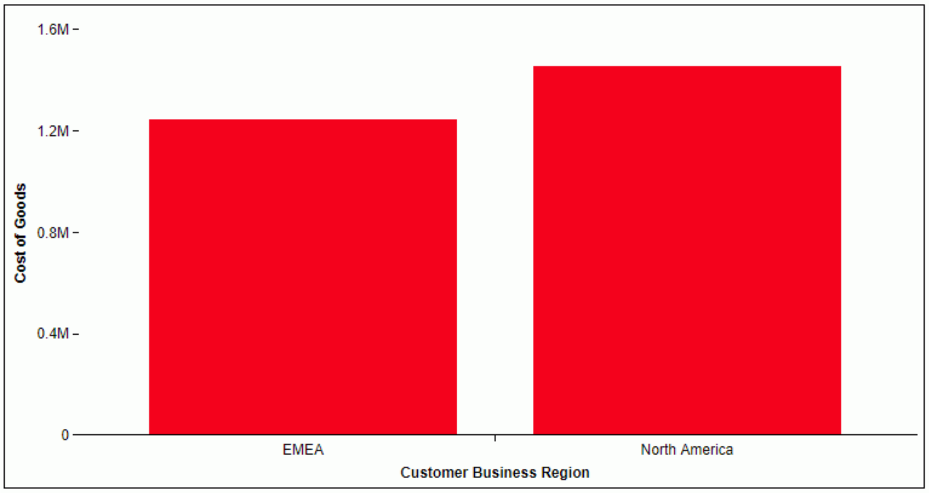
The output is shown in the following image. The minimum value on the y-axis is above zero, as shown in the following image.



The following version of the request changes the y-axis mustIncludeZero property to *true*.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
WHERE BUSINESS_REGION NE 'Oceania' OR 'South America'
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=y-axis,$
type=data, column= business_region, bucket=x-axis,$
*GRAPH_JS
"axisList": {
"x1": {
    "majorGrid": {"visible": false}
},
"y1": {
    "majorGrid": {"visible": false},
    "mustIncludeZero": true
}
}
END
```

Zero is now included on the y-axis scale, as shown in the following image.



Formatting Minor Grid Lines and Tick Marks

The minorGrid properties control the visibility and format of minor grid lines and tick marks.

Syntax: How to Format Minor Grid Lines and Tick Marks

```
"axisname": {
"minorGrid": {
  "visible": boolean,
  "count": number,
  "lineStyle": {
    "width": number,
    "color": "string",
    "dash": "string"
   },
    "ticks": {
      "length": number,
      "visible": boolean,
      "style": "string",
      "lineStyle": {
        "width": number,
        "color": "string"}
    }
  }
}
```

where;

`"`*`axisname`*`"`

Can be any axis on the axisList, or xaxis, yaxis, or zaxis.

`"minorGrid":`

Defines the properties of the minor grid lines.

`"visible":` *`boolean`*

Controls the visibility of the minor grid lines. Valid values are:

- ❏ true, which draws the minor grid lines.
- ❏ false, which does not draw the minor grid lines. This is the default value.

`"count":` *`number`*

Is the number of minor grid lines to draw between each major grid line. The default value is *undefined*.

`"lineStyle":`

Defines the line style of the minor grid lines.

`"width":` *`number`*

Is a number that defines the width of minor grid lines in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of minor grid lines using a color name or numeric specification string. The default value is "black".

For information about defining colors, see *Colors and Gradients* on page 85.

`"dash": "`*`string`*`"`

Is a string that defines the dash style. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

`"ticks":`

Defines the properties of the minor grid line tick marks.

`"length":` *`number`*

Is a number that defines the length of tick marks in pixels. The default value is 5.

`"visible": boolean`

Controls the visibility of tick marks. Valid values are

- ❑ true, which draws the tick marks.
- ❑ false, which does not draw the tick marks. This is the default value.

`"style": "string"`

Is a string that defines the tick mark style. Valid values are:

- ❑ "inner", which places the tick marks inside the minor grid lines. This is the default value.
- ❑ "outer", which places the tick marks outside of the minor grid lines. This is the default value.
- ❑ "span", which places half of each tick mark outside the minor grid line, and half of the tick mark inside the minor grid line.

`"lineStyle":`

Defines the line style of the minor grid line tick marks.

`"width": number`

Is a number that defines the width of the tick marks in pixels. The default value is 1.

`"color": "string"`

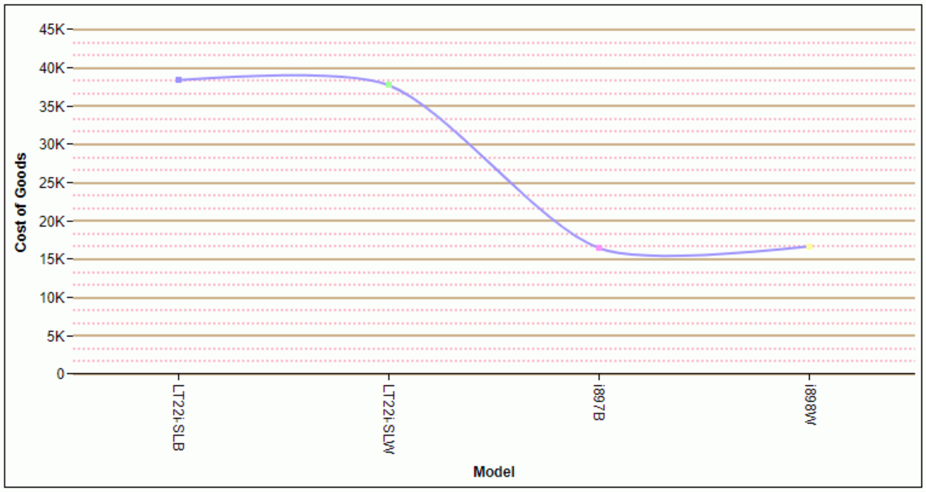
Is a string that defines the color of tick marks using a color name or numeric specification string. The default value is "black".

Example: Formatting Minor Grid Lines and Tick Marks

The following request generates y1-axis minor grid lines that are solid tan lines and minor grid lines that are dashed pink lines:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY MODEL
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"lineConnection": "curved"},
"axisList": {
"y1": {
    "majorGrid": {"visible": true, "lineStyle": {
        "width": 2, "color": "tan"}},
    "minorGrid": {"visible": true, "lineStyle": {
        "width": 2, "color": "pink", "dash": "2 2"}
    }}}
*END
ENDSTYLE
END
```

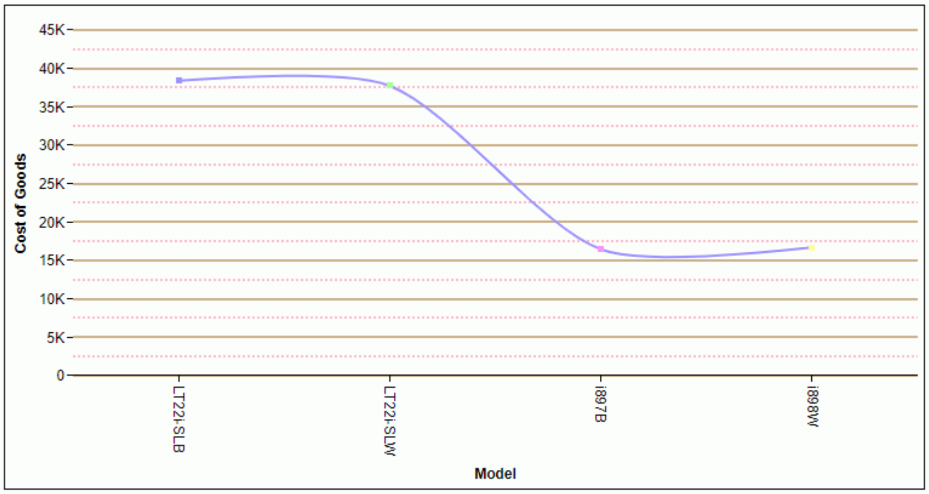
The output is:



By default, two minor grid lines are being drawn between each major grid line. To change that number to one minor grid line between major grid lines, change the y1-axis properties to the following:

```
"y1": {
"majorGrid": {"visible": true, "lineStyle": {
        "width": 2, "color": "tan"}},
"minorGrid": {"visible": true, "count": 1, "lineStyle": {
        "width": 2, "color": "pink", "dash": "2 2"}}}
```

The output is:



Controlling the X-Axis Label Layout

the xaxis:labelLayout properties control whether the x-axis can scroll and whether labels can be staggered, skipped, and truncated.

Syntax: How to Control the X-Axis Label Layout

```
"axisname": {
         "labelLayout": {
             "stagger": boolean,
             "skip": value,
             "truncate": boolean                }
      }
```

where:

`"axisname"`

Is an x-axis on the axisList, or xaxis.

`"labelLayout":`

Defines the x-axis label layout.

`"stagger":` *`boolean`*

Controls whether labels can be staggered.

Note: Generally, only x-axis labels can be staggered.

Valid values are:

- "auto", which leaves the decision up to the chart engine. This is the default value.
- true, which enables staggered labels, if necessary for the spacing of the labels.
- false, which disables staggered labels.

`"skip":` *`boolean`*`, "skip":` *`number`*

Controls whether labels can be skipped, Valid values are:

- "auto", which leaves the decision up to the chart engine. This is the default value.
- true, which enables skipped labels, if necessary for the spacing of the labels.
- false, which disables skipped labels.
- a number greater than zero which skips that number of labels.

`"truncate":` *`boolean`*

Controls whether labels can be truncated, Valid values are:

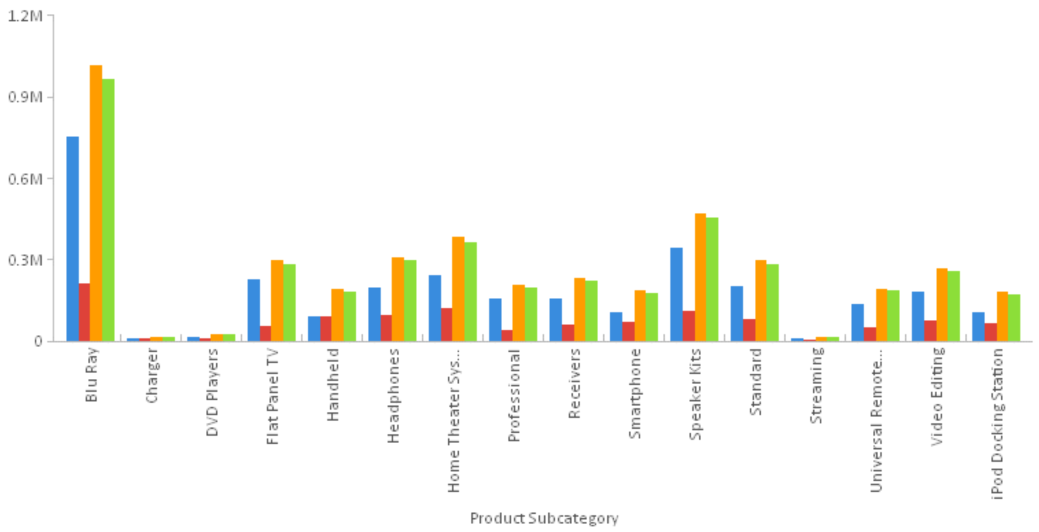
- "auto", which leaves the decision up to the chart engine. This is the default value.
- true, which enables truncated labels, if necessary for the spacing of the labels.
- false, which disables truncated labels.

Example: Controlling the Layout of X-Axis Labels

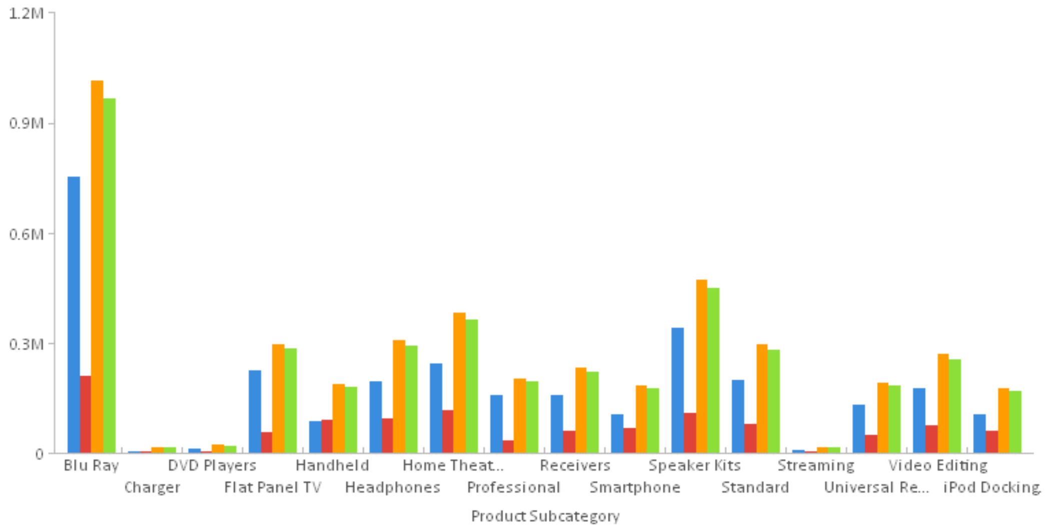
The following request uses the default x-axis label layout:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US REVENUE_US
BY PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"xaxis": {"labelLayout": {
    "stagger": "auto",
    "skip": "auto",
    "truncate": "auto"
    }}
*END
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N4, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N5, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown in the following image:



Changing the stagger property to true generates the following chart:



Formatting a Scrolling X-Axis

The xaxis:groupFit property enables or disables x-axis scrolling. Even if scrolling is enabled, it will only be implemented when needed. You have a choice between two forms of scroll bar:

- A standard scroll bar.
- A minichart, in which the scroll bar background is a compressed version of the main chart. As you scroll the bar to focus on a portion of the minichart, that portion of the main chart scrolls into the chart frame, if necessary. This type of chart is called a *focus-context chart*.

Syntax: **How to Generate a Scrolling X-Axis**

```
"axisname": {
         "groupFit": {"rule": "maxCount", "value": number},
   "scroll": {
     "style": "string",
     "miniChartProperties": {
          "height": number,
          "dataSelection": {
            "selectionRect": {"fill": "rgba(r, g, b, a)" }
                          }
                        }
        }
     }
```

where:

"*axisname*"

Is the name of an x-axis on the axisList, or xaxis.

`"groupFit": {"rule": "maxCount", "value":` *number*`}`

Specifies the maximum number of groups to show at once, unless fewer are available. The default value is *undefined*.

`"scroll":`

Defines the properties of a scrolling x-axis.

`"style": "`*string*`"`

Specifies the type of scroll bar. Valid values are:

- "simple", which generates a standard scrollbar. This is the default value.
- "miniChart", which generates a scrollbar whose background is a compressed version of the main chart.

`"miniChartProperties":`

Defines the properties of a scroll bar visualized as a minichart.

`"height":` *number*

Is the height of the minichart in pixels, or a percent string that represents a percentage of the height of the main chart. The selection rectangle on the scrollable minichart is always the same height as the minichart itself. By default, the minichart height is 20% of the height of the main chart.

`"dataSelection": "selectionRect": "fill": "rgba(r, g, b, a)"`

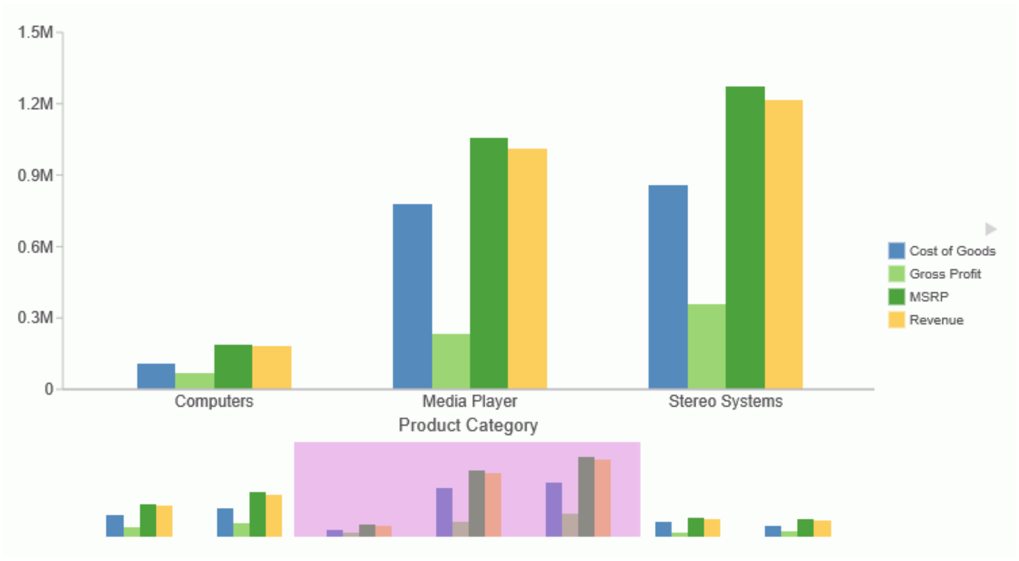
Is a color definition and transparency for the selection rectangle. If you use a color definition without a transparency value, the selection rectangle will be fully opaque and block the view of the minichart group on which it is positioned.

Example: Generating a Minichart Scrollbar on the X-Axis

The following request generates a minichart scrollbar on the x-axis. The selection rectangle is orchid and 45% opaque:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS
"axisList": {
"x1": {
    "groupFit": {"rule": "maxCount", "value": 3},
    "scroll": {
        "style": "miniChart",
        "miniChartProperties": {
            "height": 70,
            "dataSelection": {"selectionRect": {"fill": "rgba(218, 112, 214, 0.45)"}}
        }}}
}
*END
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=MSRP_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

The output is shown on the following image, with the selection rectangle (which spans a maximum of three groups) scrolled to the second block of groups:



Setting the Axis Mode

The mode property sets the axis mode as ordinal, numeric, or time. The axis mode is typically set to *undefined* which allows the charting engine to automatically set the mode based on the data.

Syntax: How to Set the Axis Mode

```
"axisname": {
  "mode": "string"
  }
```

where:

`"axisname"`

Can be any axis on the axisList, or xaxis, yaxis, or zaxis.

`"mode": "string"`

Is a string that specifies the axis mode. If the value is *undefined*, the mode is automatically chosen based on the data. Other supported values are:

- "ordinal"

- "numeric"
- "count"

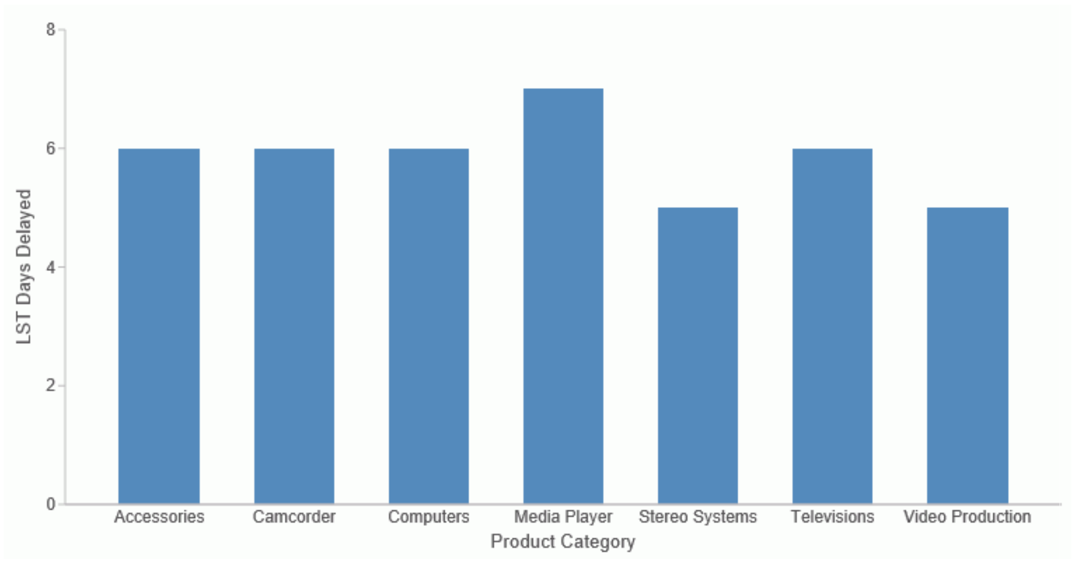
  This mode is supported for a numeric axis in order to guarantee an integer scale when the axis is assigned to a measure that has discrete, rather than continuous, values. This mode is useful for displaying histograms and other charts with a discrete measure.
- "time"

Example: Using Count Mode for a Numeric Axis

The following request charts the number of days delayed by product category. The y-axis mode is set to "count", ensuring that the y-axis only shows integers.

```
GRAPH FILE wf_retail_lite
SUM LST.DAYSDELAYED
BY PRODUCT_CATEGORY
WHERE DAYSDELAYED GT 0
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET AUTOFIT ON
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
*GRAPH_JS
 "axisList:"{
     "y1": {"mode": "count"}
            },
*END
ENDSTYLE
END
```

The output is shown in the following image.



Controlling the Format of Numeric Axis Labels

On a numeric axis, the numberFormat property controls the format of numeric axis labels. The number format can be specified as a JSON object, a format string, or a user-defined function.

For information and examples about defining number formats, see *Formatting Numbers* on page 108.

Reversing the Default Axis Locations

The swapChartSide property controls the location of the axis body line and labels. In the default configuration, the y-axis body line and labels appear on the bottom of a horizontal chart and the left side of a vertical chart. The y2-axis body line and labels appear on the top of a horizontal chart and the right side of a vertical chart. In Bubble and Scatter charts, the x-axis normally appears on the bottom of the chart and the y-axis on the left side of the chart. This property can be used to reverse these default locations.

Syntax: How to Reverse Default Axis Locations

```
"axisname": {
  "swapChartSide": boolean  }
```

where:

"*axisname*"

Can be an axis on the axisList, or xaxis, yaxis, or y2axis.

"swapChartSide": *boolean*

Can be:

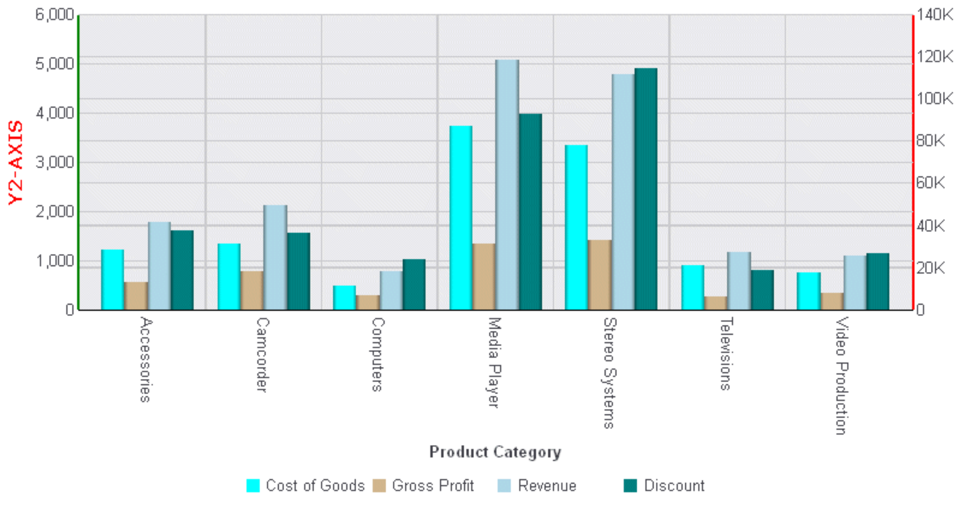
- true, which swaps axis label and body line locations.
- false, which uses the default axis locations. This is the default value.

Example: Reversing the Default Axis Positions

The following request assigns series 3 to the y2-axis and swaps the y2-axis with the y-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"yaxis": {
    "title": {
        "visible": true,"color": "red", "font": "bold 10pt Verdana",
        "text": "Y-AXIS"},
    "bodyLineStyle": {"width": 2, "color": "red"},
    "swapChartSide": true},
"y2axis": {
    "title": {
        "visible": true, "color": "red", "font": "bold 10pt Verdana",
        "text": "Y2-AXIS"},
    "bodyLineStyle": {"width": 2, "color": "green"}},
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "tan"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "teal","yAxisAssignment": 2}]
*END
ENDSTYLE
END
```

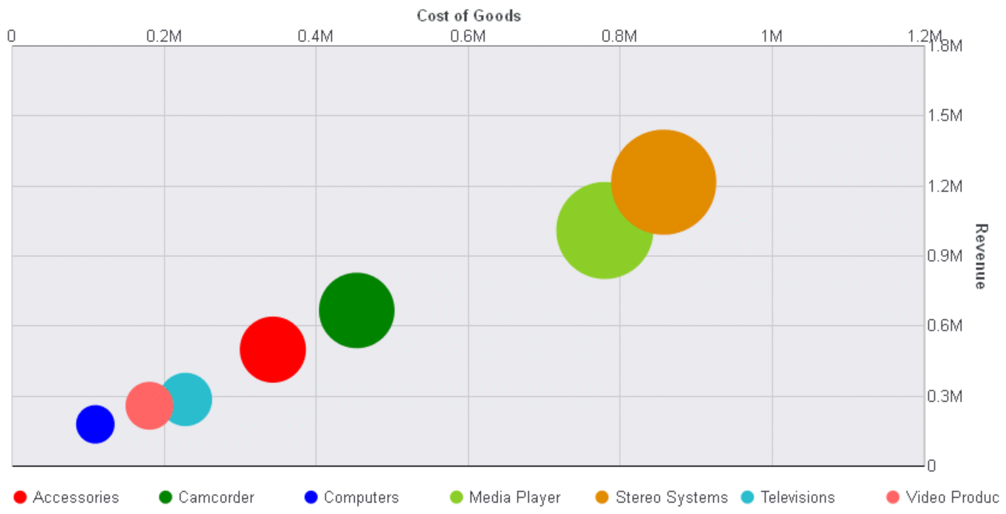
On the output, the y-axis has been moved to the right side of the chart, and the y2-axis is on the left:



The following request generates a bubble chart with the y-axis on the right, and the x-axis on the top:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"axisList": {
"y1": {"swapChartSide": true},
"x1": {"swapChartSide": true}
  },
"series": [
    {"series": "all", "marker": {"shape": "circle"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "green"},
    {"series": 2, "color": "blue"}]
*END
ENDSTYLE
END
```

The output is:



Formatting an Axis Title

The title properties control the content, visibility, and format of an axis title.

Syntax: How to Format an Axis Title

```
"axisname": {
  "title": {
    "text": "string",
    "visible": boolean,
    "font": "string",
    "color": "string",
    "position": "orthogonal"
}
}
```

where:

`"axisname"`

Can be an axis on the axisList or xaxis, yaxis, y2axis, or zaxis.

`"text": "string"`

Is a string that defines axis title text. The default value for xaxis is the sort field name or title, or the name assigned with an AS phrase.

`"visible": boolean`

Controls the visibility of the axis title. Valid values are:

- ❏ true, which makes the axis title visible.
- ❏ false, which makes the axis title not visible. This is the default value.

`"font": "string"`

Is a string that defines the size, style, and typeface of the axis title. The default value for xaxis is 'bold 9pt Sans-Serif'.

`"color": "string"`

Is a string that defines the color of the axis title using a color name or numeric specification string. The default value is "black".

For information about defining colors, see *Colors and Gradients* on page 85.

`"position": "orthogonal"`

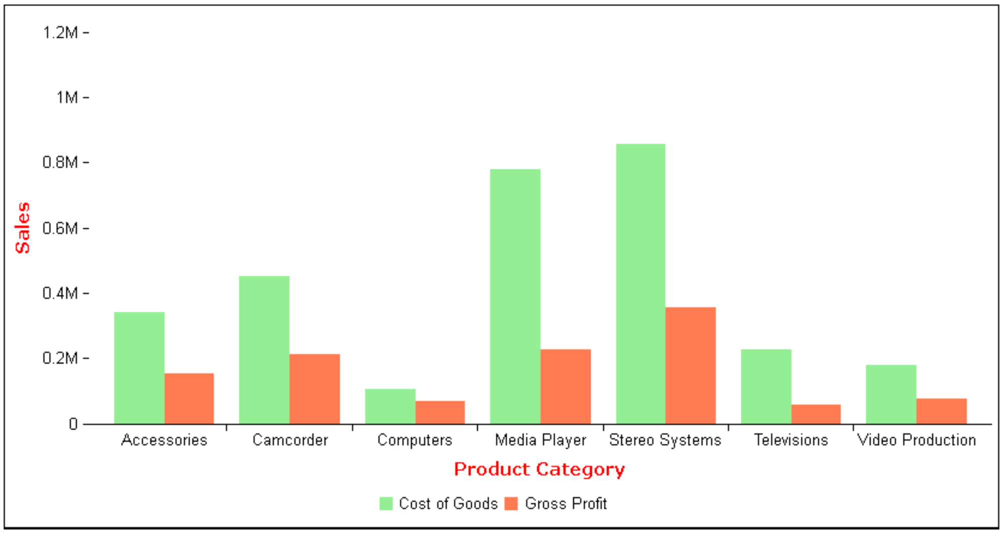
Applies to the y-axis and y2-axis. Places the title above and centered over the axis body line.

Example: Formatting the Axis Titles

The following request makes the axis titles visible and red. It changes the font for the axis titles to bold 10pt Verdana:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"xaxis": {"title": {
    "visible": true, "color": "red", "font": "bold 10pt Verdana"}},
"yaxis": {"title": {
    "visible": true, "color": "red", "font": "bold 10pt Verdana", "text":
"Sales"}},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"}]
*END
ENDSTYLE
END
```

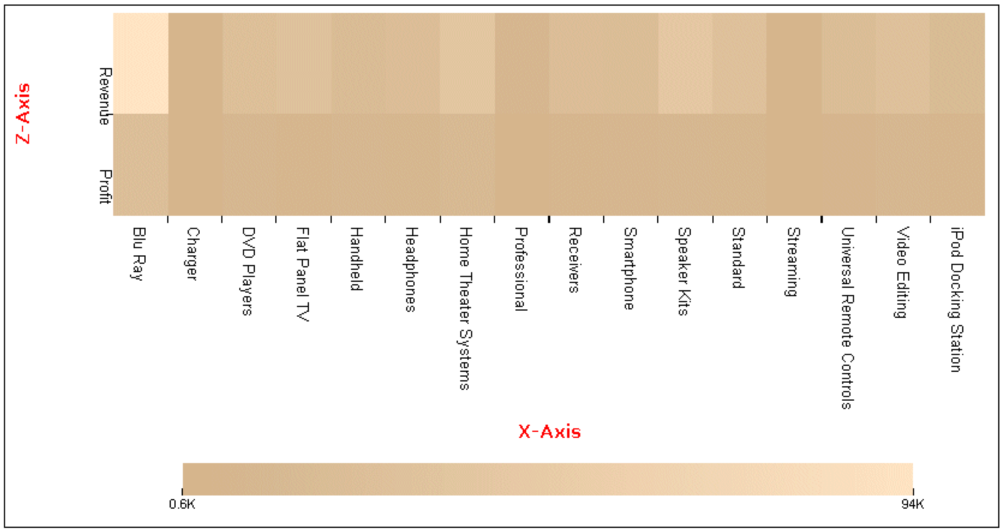
The output is:



The following request generates a heatmap with x- and z-axis titles in red, with font bold 10pt Verdana:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US AS 'Revenue'
        GROSS_PROFIT_US AS 'Profit'
BY PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SPECTRAL
ON GRAPH SET STYLE *
*GRAPH_JS
"colorScale": {"colors": ["tan", "bisque"]},
"xaxis": {"title": {
    "visible": true, "color": "red", "font": "bold 10pt Verdana", "text": "X-Axis"}},
"zaxis": {"title": {
    "visible": true, "color": "red", "font": "bold 10pt Verdana", "text": "Z-Axis"}}
*END
ENDSTYLE
END
```

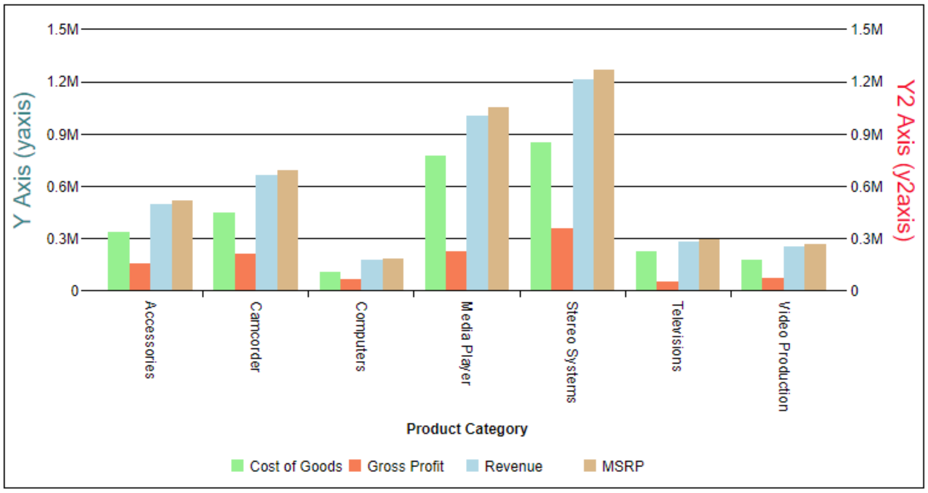
The output is:



The following request assigns two series to the y-axis and two to the y2-axis and formats the titles for those axes:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"axisList": {
"y1": {"title": {
    "visible": true, "text": "Y Axis (yaxis)", "font": "14pt Sans-Serif", "color":
"teal"}},
"y2": {"title": {
    "visible": true, "text": "Y2 Axis (y2axis)", "font": "14pt Sans-Serif", "color":
"red"}}
},
"series": [
    {"series": 0, "color": "lightgreen", "yAxisAssignment": 1},
    {"series": 1, "color": "coral", "yAxisAssignment": 2},
    {"series": 2, "color": "lightblue", "yAxisAssignment": 1},
    {"series": 3, "color": "burlywood", "yAxisAssignment": 2}]
*END
ENDSTYLE
END
```

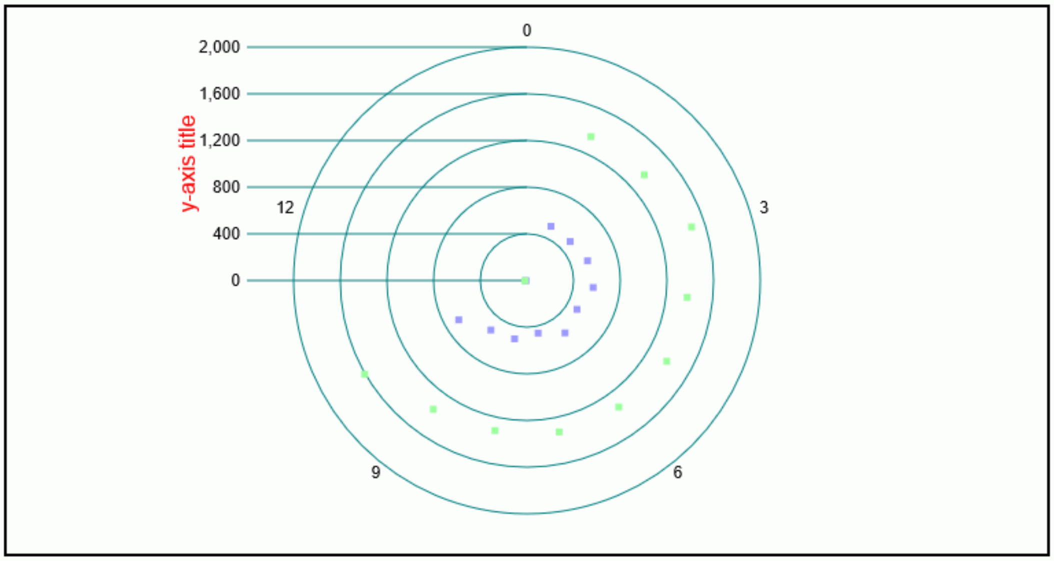
The output is:



The following request generates a polar chart and formats the y-axis label:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
ACROSS TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH POLAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"polarProperties": {"straightGridLines": false, "extrudeAxisLabels": true},
"yaxis": {
    "title": {"visible": true,
        "text": "y-axis title", "font": "12pt Sans-Serif", "color": "red"},
    "majorGrid": {"visible": true, "lineStyle": {"width": 1, "color":
"teal"}
        }}
*END
ENDSTYLE
END
```

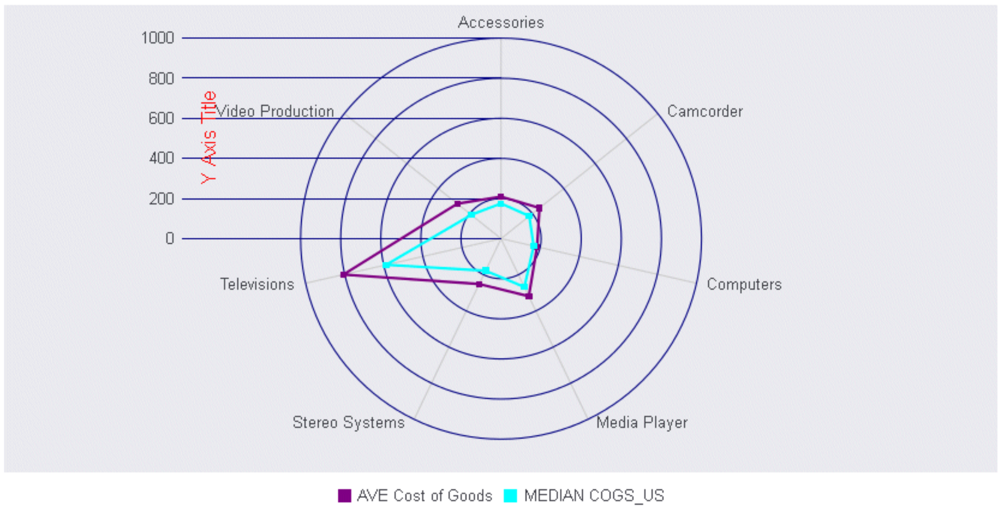
The output is:



The following request generates a radar chart with a y-axis title:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH RADARL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"polarProperties": {"extrudeAxisLabels": true},
"yaxis": {
    "title": {"visible": true, "text": "Y Axis Title",
        "font": "12pt Sans-Serif", "color": "red"},
    "majorGrid": {"lineStyle": {"width": 1, "color": "navy"}}},
"series": [
    {"series": 0, "color": "purple", "border": {"width": 2}},
    {"series": 1, "color": "cyan", "border": {"width": 2}}]
*END
END
```

The output is:



Defining the Position of the Y-Axis or Y2-Axis Title

The "position": "orthogonal" property places the y-axis or y2-axis title above and centered over the y-axis or y2-axis body line.

Syntax: How to Position the Y-Axis Title Over the Y-Axis Body Line

```
"axisname": {
   "title":{
     "visible": true,
     "position": "orthogonal"
}
}
```

where:

"*axisname*"

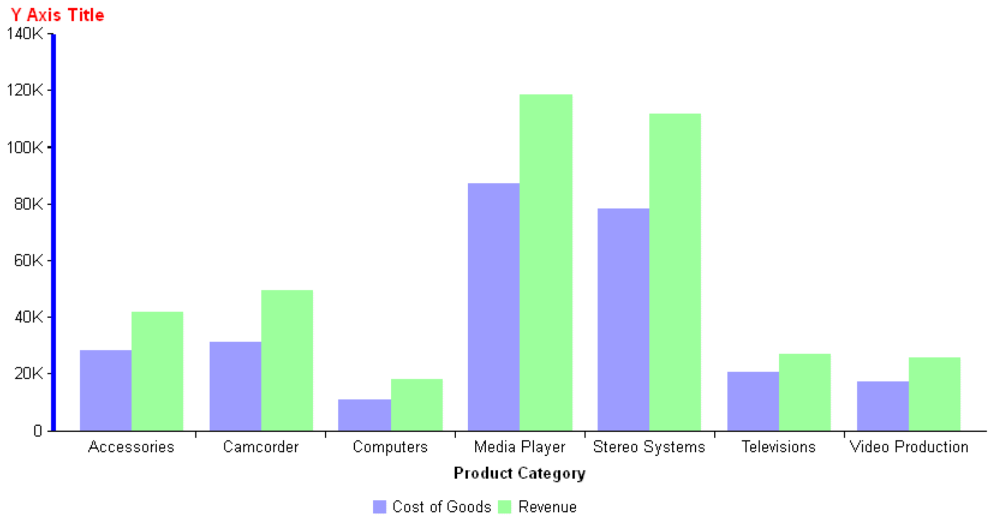
Can be a y-axis on the axisList, or yaxis or y2axis.

Example: Positioning the Y-Axis Title Over the Y-Axis Body Line

The following request makes the y-axis title red and bold 10pt Arial, and places it over the y-axis body line:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 0},
"axisList": {
 "y1": {
     "title": {"visible": true, "position": "orthogonal",
        "text": "Y Axis Title", "font": "bold 10pt Arial", "color": "red"},
     "bodyLineStyle": {"width": 4, "color": "blue"}}
}
*END
ENDSTYLE
END
```

The output is:



Chapter 9

Series-Specific Properties

This chapter describes how to control the appearance of individual series and groups.

In this chapter:

Series-Specific Properties Overview

These properties can be used to control the appearance of individual series and groups.

Note: the data label properties are now series-specific properties, and the showDataValues property has been deprecated and is no longer needed.

The following syntax shows the series properties.

```
series: [
{
   "series": number,
   "group": number,
   "visible": true,
   "label": "string",
   "color": "string",
   "riserShape": "string",
   "connectMarkers": false,
   "border": {
      {"cornerRadius":
           {"x": undefined,
            "y": undefined},
      }
      "width": number,
      "color": "string",
      "dash": "string"
   },
   "marker": {
      "visible": boolean,
      "color": "string",
      "size": number,
      "shape": "string",
      "rotation": number,
      "position": "string",
      "fillEffect": "string",
      "border": {
         "width": number,
         "color": "string",
         "dash": "string"
      }
   },

"dataLabels": {
   "visible": false,
   "displayMode": "x",
   "position": "top",
   "font": "7.5pt Sans-Serif",
   "color": "black",
   "useNegativeColor": false,
   "negativeColor": "red",
   "numberFormat": "auto",
   "formatCallback": undefined
   "feelerLine": {
      "visible": true,
      "width": 1,
      "color": "black",
      "dash": ""}
},
   // Per-series trendline
   "trendline": {
      "enabled": false,
      "mode": undefined,
      "order": 3,
      "lineStyle": {
         "width": 1,
         "color": undefined,
         "dash": ""
      },
      "equationLabel": {
         "visible": false,
         "font": "8pt Sans-Serif",
         "color": undefined,
         "mode": "equation"
      }
   },
   "explodeSlice": number,
   "deleteSlice": boolean,
   "yAxisAssignment": number,
   "tooltip": tool_string_or_function
}
]
```

The following synax segment shows the default settings:

```
"series": [
    {"series": "all", "color": "blue",
        "showDataValues": false, "border": {"width": 2},
        "marker": {"size": 8, "border": {
            "width": 1, "color": "black"}}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "green"},
    {"series": 2, "color": "orange"}]
]
```

The following syntax segment shows the named fill pattern properties:

```
 "color":{
               "type": "pattern",
                "color": undefined,
                   "backgroundColor": undefined,
                "lineWidth": 1,
                   "antialias": true,
                "shape": undefined,
                   "size": 20,
                "pad": 1
               }
```

Selecting Specific Series

A series selection is a zero-based number or a string. If the series does not exist in the chart, the property is ignored.

Syntax: How to Select Specific Series

```
"series": [
{"series": number, "property":value, ..., property:value},
]
```

where:

`"series": number`

Selects a series for the specified properties. Valid values are:

- ❏ A zero-based series number. If the series does not exist in the chart, the property is ignored.
- ❏ "all", to apply the properties to all series.
- ❏ "reset", to remove default properties before assigning the specified properties.

`"property":value`

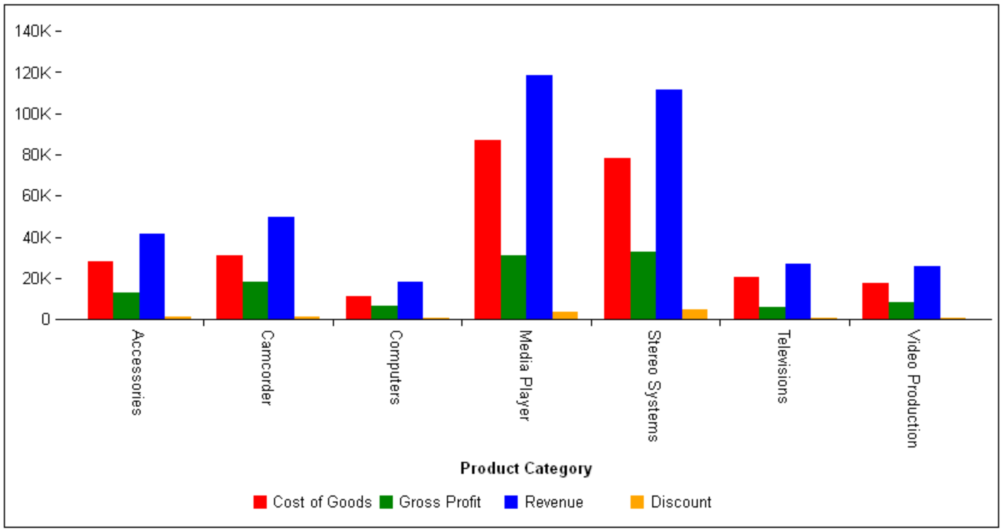
Is a supported series property and its value.

Example: Selecting Specific Series

The following request generates a vertical bar chart with four series and specifies colors for 10 series:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "red"},
    {"series": 1, "color": "green"},
    {"series": 2, "color": "blue"},
    {"series": 3, "color": "orange"},
    {"series": 4, "color": "lightgreen"},
    {"series": 5, "color": "yellow"},
    {"series": 6, "color": "slateblue"},
    {"series": 7, "color": "lavender"},
    {"series": 8, "color": "limegreen"},
    {"series": 9, "color": "red"}
    ]
*END
ENDSTYLE
END
```

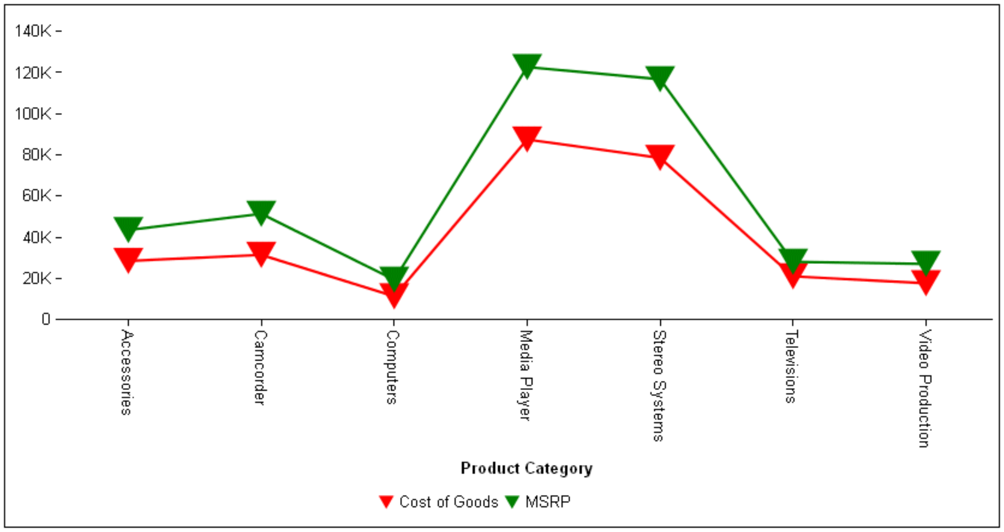
The colors for series that are not in the chart are ignored. The output is:



The following request generates a vertical line chart and sets the markers for all series to the shape 'triangle' and the size 20:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": "all", "marker": {"shape": "triangle", "size": 20}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "green"}]
*END
ENDSTYLE
END
```

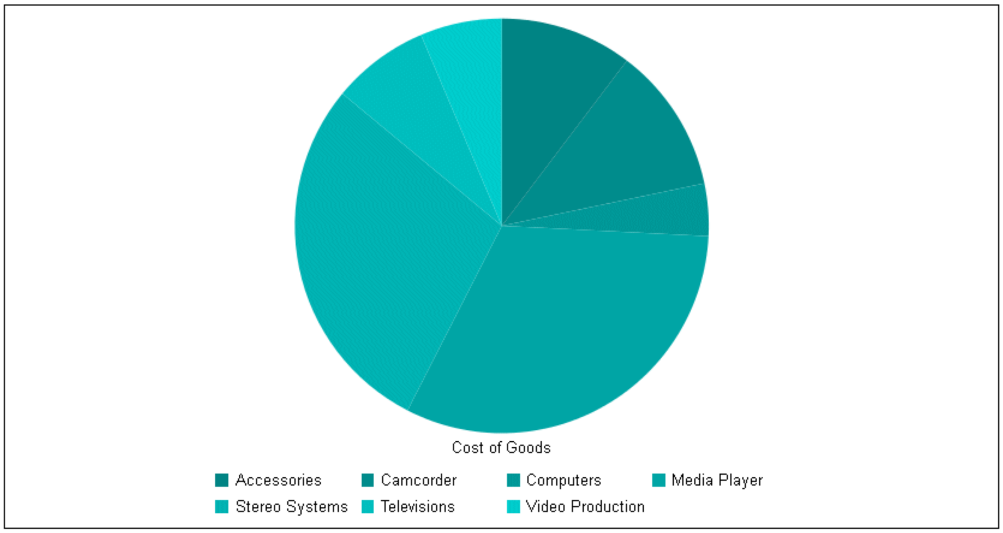
The output is:



The following request sets the riserCycleEndLightness property, which cycles through defined colors and then, if more series exist, uses a lightening or darkening effect for the rest of the series. However, in order to apply this effect, the default colors must be removed, or they will be used instead of the lightening effect:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"riserCycleEndLightness": 0.4,
"series": [
    {"series": "reset", "color": undefined},
    {"series": 0, "color": "teal"}]
*END
ENDSTYLE
END
```

The output is:



Selecting Specific Groups

A group number is a zero-based number that represents a group in the request.

Syntax: How to Select Specific Groups

```
{"series": number, "group": number,
   "property": value, ..., "property": value}
```

where:

`"series": number`

Defines the series for the selected group.

`"group": number`

Is a zero-based number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the properties are applied to all risers in the series.

`"property": value`

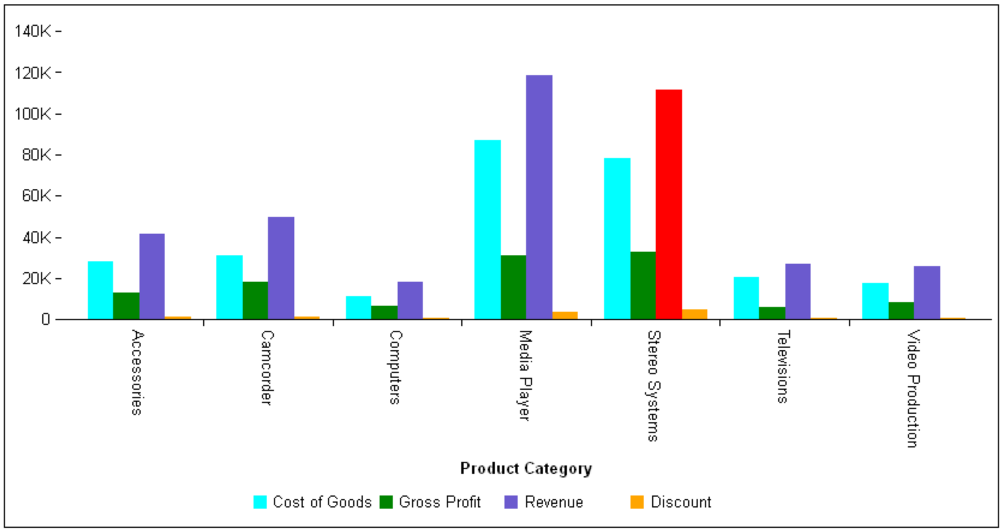
Is a supported series property and its value.

Example: Selecting a Specific Group

The following request generates a vertical bar chart and defines a color for each series. It defines a different color (red) for group 4 in series 2:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "green"},
    {"series": 2, "group": 4, "color": "red"},
    {"series": 2, "color": "slateblue"},
    {"series": 3, "color": "orange"}
    ]
*END
ENDSTYLE
END
```

On the output, the series 2 risers are all slate blue except for the red one in group 4:



Defining a Border for Series Risers

The border properties define a border for all risers, for all risers in an individual series, or for an individual riser identified by a series and group number. It can be used for area risers, bar risers, funnel segments, gauge needles, and pie slices.

Syntax: How to Define a Border for Series Risers

```
"series": [
   {
      "series": number,
      "group": number, // optional
      "border": {
         "width": number,
         "color": "string",
         "dash": "string"
      },
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"group": *number*`

Is an optional zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the border is applied to all risers in the series.

`"border":`

Defines the properties of the riser border.

`"width": *number*`

Is a number that defines the width of the border in pixels.

`"color": "*string*"`

Is a color defined by a color name or numeric specification string.

For information about defining colors, see *Colors and Gradients* on page 85.

`"dash": "*string*"`

Is a string that defines the border dash style. You can use strings of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels. You can also use the following named line styles:

Pattern Name	**Description**	**Sample**	**Corresponding Pixel Properties**
"dash"	The line consists of dashes.		"dash": "5 4"
"dash_dot"	The line consists of alternating dashes and dots.		"dash": "5 4 2 4"
"dash_dot_dot"	The line consists of a series of dashes followed by two dots.		"dash": "10 4 2 4 2 4"

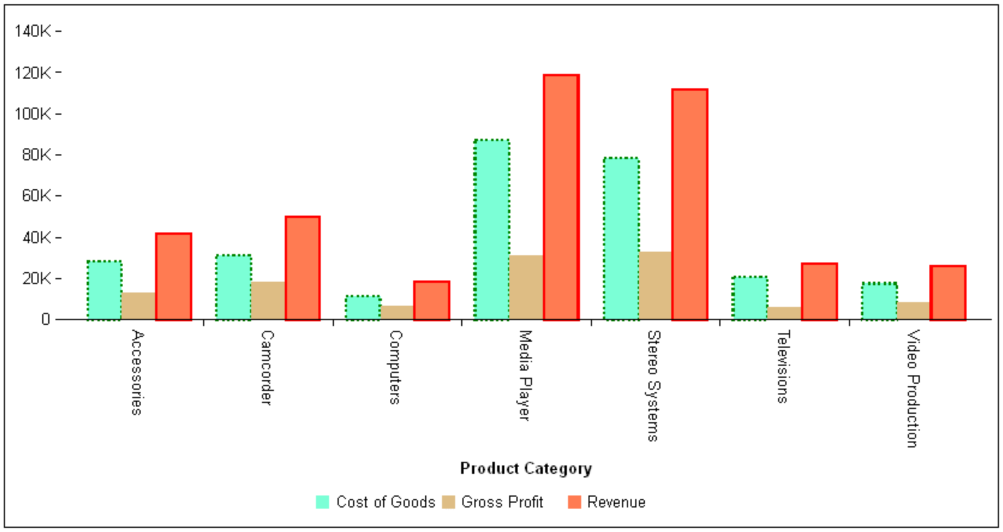
Pattern Name	Description	Sample	Corresponding Pixel Properties
"dot"	The line consists dots.	·········	"dash": "2 4"
"long_dash"	The line consists of long dashes.	— — — —	"dash": "10 4"
"long_dash_dot"	The line consists of alternating long dashes and dots.	— · — · — ·	"dash": "10 4 2 4"
"short_dash"	The line consists of dashes.	--------	"5 2"
"short_dash_dot"	The line consists of alternating dashes and dots.	-·-·-·-·	"dash": "5 2 2 2"
"short_dash_dot_dot"	The line consists of a series of dashes followed by two dots.	-··-··-··	"dash": "5 2 2 2 2 2"
"short_dot"	The line consists dots.	············	"dash": "2 2"
"solid"	The line is solid.	———	"dash": ""

Example: Defining Borders for Series Risers

The following request generates a bar chart with three series. The risers for series 0 have a green dashed border, and the risers for series 2 have a red solid border:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "aquamarine", "border": {
        "width": 2, "color": "green", "dash": "2 2"}},
    {"series": 1, "color": "burlywood"},
    {"series": 2, "color": "coral", "border": {
        "width": 2, "color": "red"}}
    ]
*END
ENDSTYLE
END
```

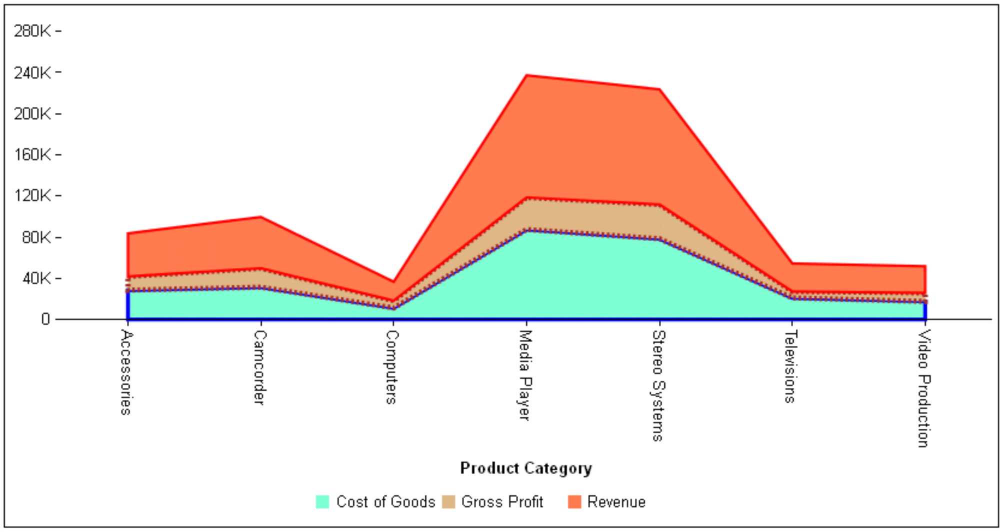
The output is:



The following request generates a stacked area chart. The riser for series 0 has a blue solid border, the riser for series 1 has a brown dashed border, and the riser for series 2 has a red solid border:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VAREASTK
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "aquamarine", "border": {
        "width": 3, "color": "blue"}},
    {"series": 1, "color": "burlywood", "border": {
        "width": 4, "color": "brown", "dash": "2 2"}},
    {"series": 2, "color": "coral", "border": {
        "width": 2,"color": "red"}}
    ]
*END
ENDSTYLE
END
```

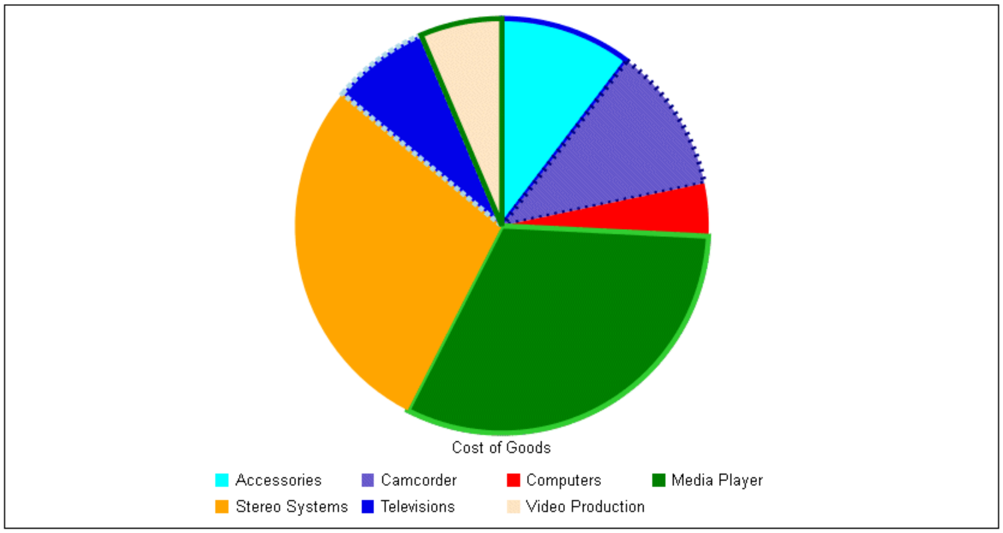
The output is:



The following request generates a pie chart and defines borders for some of the slices:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "cyan", "border": {
        "width": 4, "color": "blue"}},
    {"series": 1, "color": "slateblue", "border": {
        "width": 4, "color": "navy", "dash": "2 4"}},
    {"series": 2, "color": "red"},
    {"series": 3, "color": "green", "border": {
        "width": 4, "color": "limegreen"}},
    {"series": 4, "color": "orange"},
    {"series": 5, "color": "blue", "border": {
        "width": 4, "color": "lightblue", "dash": "4 2"}},
    {"series": 6, "color": "bisque","border": {
        "width": 4, "color": "green"}}
    ]
*END
ENDSTYLE
END
```

The output is:

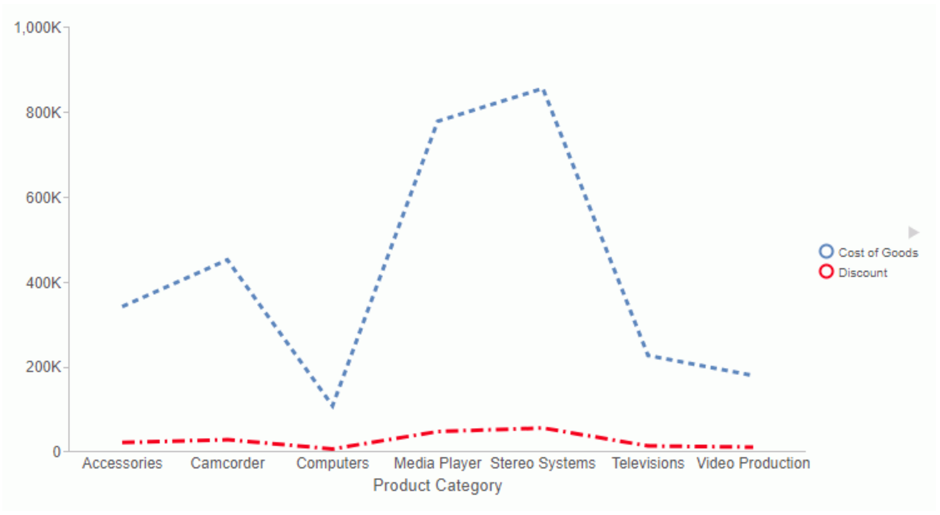


Example: Using Named Constants for Line Styles

The following request uses the "dash" style for series 0 and the "long_dash_dot" style for series 1.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis,$
TYPE=DATA, COLUMN=DISCOUNT_US, BUCKET=y-axis,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS
"series":[
 {"series":0, "border": {"dash": "dash"}},
 {"series":1, "color": "red", "border": {"dash": "long_dash_dot"}}
 ]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Defining Colors for Series Risers

The color properties define a color for all risers, for all risers in an individual series, or for an individual riser identified by a series and group number. If a group number is not specified, the color is also applied to the series icon in the legend area.

Syntax: How to Define Colors for Series Risers

```
"series": [
   {
      "series": number,
      "group": number, // Optional
      "color": "string"
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"group": number`

Is an optional zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the color is applied to all risers in the series.

`"color": "string"`

Is a color defined by a color name or numeric specification string, or a gradient defined by a string.

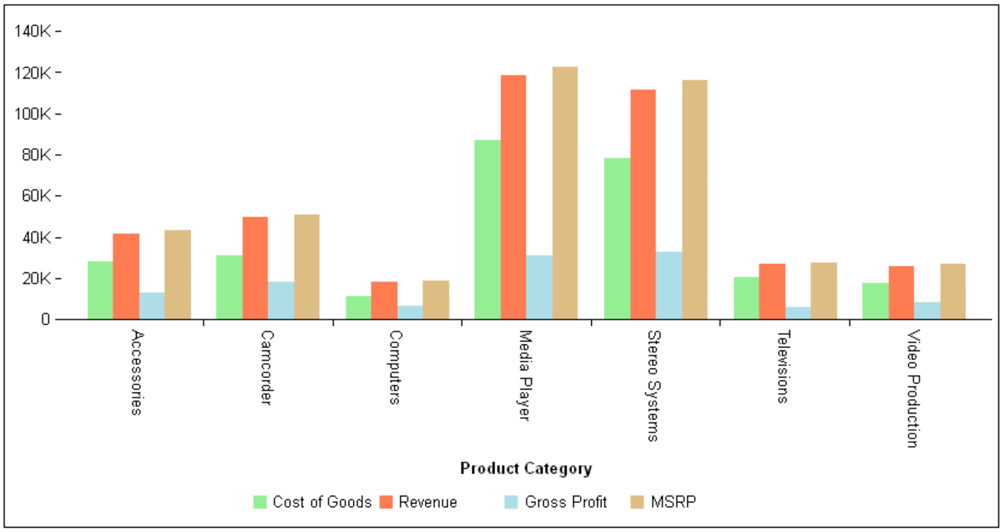
For information about defining colors and gradients, see *Colors and Gradients* on page 85.

Example: Defining Colors for Series Risers

The following request generates a vertical bar chart and defines colors for each series:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US GROSS_PROFIT_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "coral"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "burlywood"}]
*END
ENDSTYLE
END
```

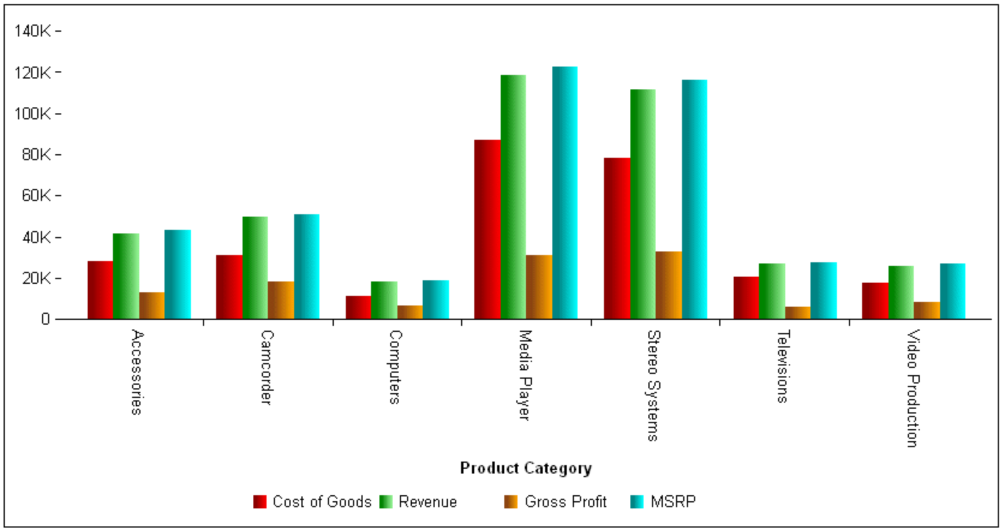
The output is:



The following version of the request applies linear gradients to each series:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US GROSS_PROFIT_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color":
        "linear-gradient(0%,0%,100%,0%, 20% darkred, 95% red)"},
    {"series": 1, "color":
        "linear-gradient(0%,0%,100%,0%, 20% green, 95% lightgreen)"},
    {"series": 2, "color":
        "linear-gradient(0%,0%,100%,0%, 20% saddlebrown, 95% orange)"},
    {"series": 3, "color":
        "linear-gradient(0%,0%,100%,0%, 20% teal, 95% cyan)"}]
*END
ENDSTYLE
END
```

The output is:

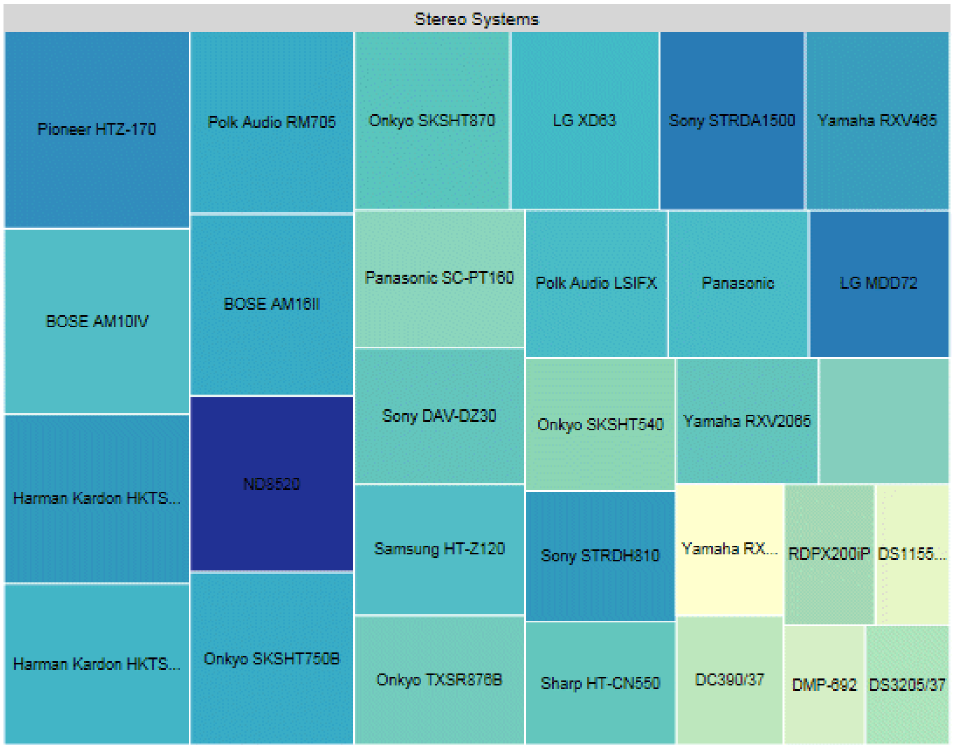


Showing and Formatting Data Text Labels

The dataLabels properties control the visibility and format of data text labels for all or specific series.

When series:dataLabels:position is set to 'outside', the feelerLine property controls the appearance of feeler lines between a pie chart and its data text labels.

If a data label in a Treemap does not fit into its rectangle, it will be truncated and appended with ellipsis (...), as in the following image.



Some properties for data labels are not part of a series object. A standalone dataLabels object controls these properties. To see an example of using the standalone dataLabels object to override default property values set by the WebFOCUS Reporting Server, see *Using the Chart Template Engine to Customize Tooltips, Titles, and Data Labels* on page 130. These properties must be included in a GRAPH_JS_FINAL block in the WebFOCUS StyleSheet.

Syntax: How to Control Display of Data Labels That Do Not Fit Their Containers

A property controlled by the standalone dataLabels object is whether to display data text labels that do not fit inside their containers. By default, these labels are not drawn. The syntax is:

```
"dataLabels": {
    "clipToContainer": boolean
                }
```

where:

`"clipToContainer": boolean`

Specifies whether to draw data text labels that do not fit in their containers. Valid values are:

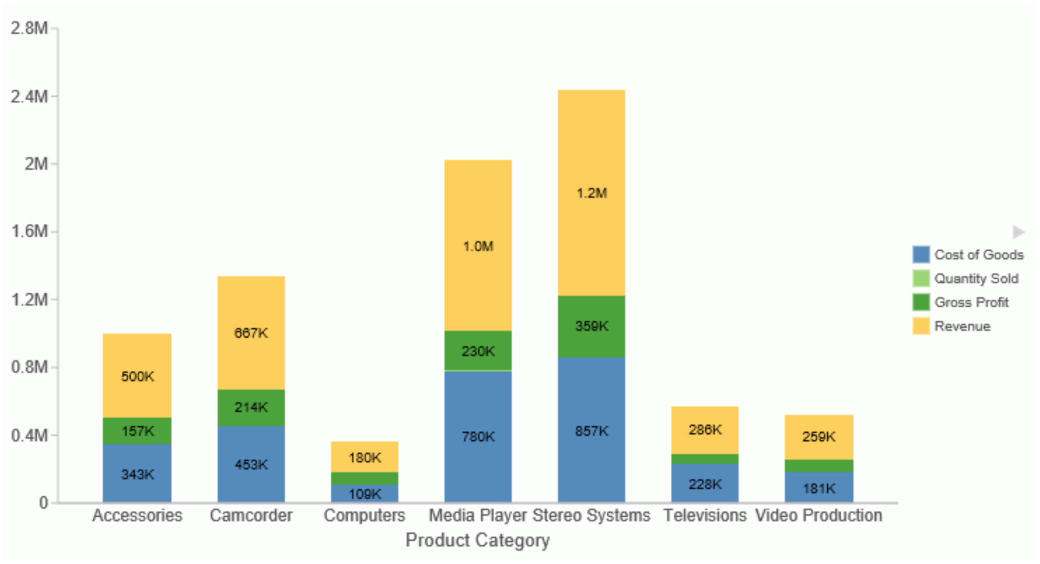
- ❏ true, which does not draw data text labels that do not fit their containers. This is the default value.
- ❏ false, which draws all data text labels.

Example: Controlling the Display of Data Labels That Do Not Fit Their Containers

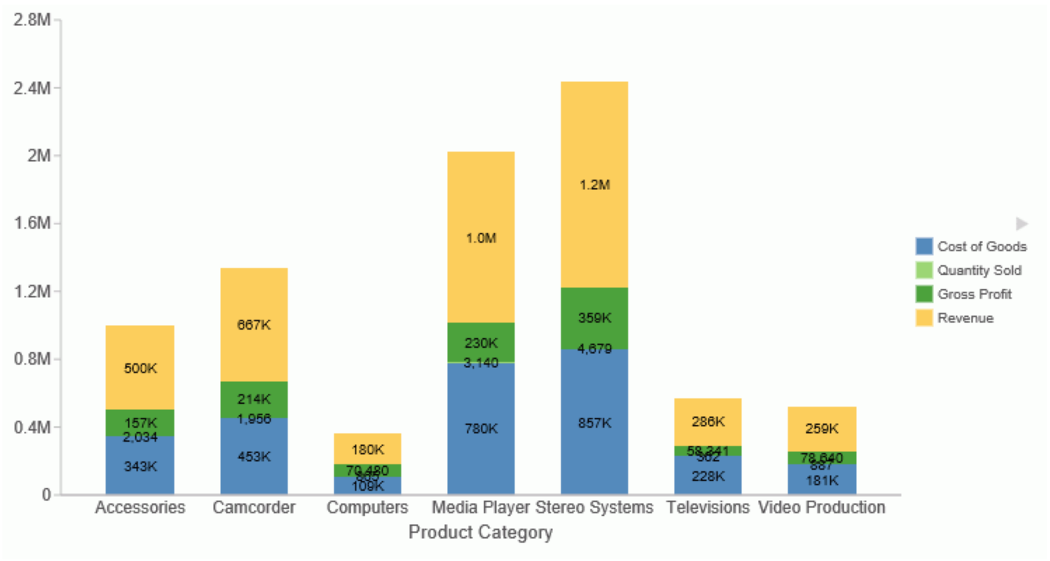
The following request generates a stacked bar chart and sets clipToContainer to *true*, the default.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US QUANTITY_SOLD GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT OFF
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-SERIES-LAYOUT=stacked,$
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N4, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N5, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
*GRAPH_JS
"dataLabels": {"visible":true, "position": "inside",
    "clipToContainer": true}
*END
ENDSTYLE
END
```

On the output, the labels that do not fit are omitted, as shown in the following image.



Changing clipToContainer to *false* shows all data labels, as shown in the following image.



Syntax: How to Show and Format Series Data Text Labels

```
"series":[
{"series": s,
   "dataLabels": {
       "visible": boolean,
       "content": "string",
       "content": function(),
       "position": "string",
       "font": "string",
       "color": "string",
       "useNegativeColor": boolean,
       "negativeColor": "string",
       "numberFormat": "numformat",
       "feelerLine": {
          "visible": boolean,
          "width": number,
          "color": "string",
          "dash": "string"
   }
}
} ...
]
```

where:

s

Is a series identifier, either a number, 'all', or 'reset'.

"visible": *boolean*

Can be:

- true, which makes the data text labels visible. When visible is true, use the series-specific showDataValues property to hide data labels for individual series.
- false, which makes the data text labels not visible. This is the default value.
- "auto", which lets the chart engine determine whether the data labels are visible.

"content": "*string*"

Controls the data text to show. Valid values are "x", "y", "z", "%", "%+", "cumulative", "groupLabel", or "seriesLabel". For choropleth map charts, "name" can be used to display location names. The default value is "x".

- For a bubble chart, use "x", "y", or "z" to identify which value to show (x-Position, y-Position, or Size (z)).
- For a line chart where the data set includes marker size values, use "x" or "y" to show the y-axis value. Use "z" to show the marker size values.

- For a scatter chart, use "x" or "y" to identify which value to show (x-Position, y-Position).
- For a pie chart, use "%" to show the percentage value of a pie slice (as a decimal) or "value" (the default for pie charts) to show the measure value for the slice.

  For % mode in pie charts, you must set the numberFormat property to display the value as a percentage (for example, "numberFormat": "[>9]#,#%;[<0]-#.#%;[<=9]#.##%;[=0]0"). For information on formatting numbers, see *Formatting Numbers* on page 108.
- For stacked bar, line, or area charts ("blaProperties": {"seriesLayout": "stacked"}), use "cumulative" to show the stack total, "%" to show the stack value as a percentage, or "%+" to show percentage and cumulative values.
- For a Leaflet choropleth map chart, use "name" to show the location names. For an example, see *Map Support* on page 817.
- You can also use the content property to define a function to display multiple values.

`"content": function()`

Identifies a function to call for each data label. The default value is *undefined*. This function can use up to four arguments (data value, series ID, group ID, and data object) and should return a string to be displayed. For more information on callback functions, see *Defining a Callback Function to Display Values in Labels* on page 124.

`"position": "`*`string`*`"`

Is a string that defines the position of data labels relative to the risers. Valid values are "bottom", "center", "insideBottom", "insideTop", "left", "outside", "right", "top", or "auto". The default value is "top".

- The "insideBottom" setting is for bar risers only and draws the data labels inside the bottom of the riser.
- The "insideTop" setting is for bar risers and pie slices only and draws the data labels inside the top edge of the riser or slice.
- For pie charts, the only valid settings are "center" (on pie slices), "insideTop", and "outside" (outside pie slices with feeler lines).
- For bar risers, positions are orientation aware (for example, "top" corresponds to "right" in a horizontal bar).
- For bullet charts, also see the series-specific marker position property that defines the position of the marker relative to data text.

- ❏ The "auto" setting lets the chart engine decide where to position the labels.

Note: Position *On top edge* is not supported in JSCHART format.

`"font": "`*`string`*`"`

Is a string that defines the size, style, and typeface of the data text labels. The default value is "font": "7.5pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color by name, numeric specification string, or gradient definition string, of the data text labels. The default value is "black".

`"useNegativeColor":` *`boolean`*

Controls whether to use a different color for data text labels for negative risers. Can be:

- ❏ true, which makes the data text labels for negative risers the color defined by the negativeColor property.
- ❏ false, which uses the same color for all data text labels. This is the default value.

`"negativeColor": "`*`string`*`"`

Is a string that defines the color for data text labels for negative risers by name, numeric specification string, or gradient definition string, of the data text labels. The default value is "red".

`"numberFormat": "`*`numformat`*`"`

Is the number format for the data labels. The number format can be specified as a JSON object, a format string, or a user-defined function. The default value is "auto". You can also use autoNumberFormats to apply a number format to all data text labels. For information on number formats, see *Formatting Numbers* on page 108.

`"feelerLine":`

Defines the properties of the feeler lines.

`"visible":` *`boolean`*

Valid values are:

- ❏ true, to make the feeler lines visible. This is the default value.
- ❏ false, to make the feeler lines not visible.

`"width":` *`number`*

Is a number of pixels that defines the width of the feeler lines. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the feeler lines, using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"dash": "`*`string`*`"`

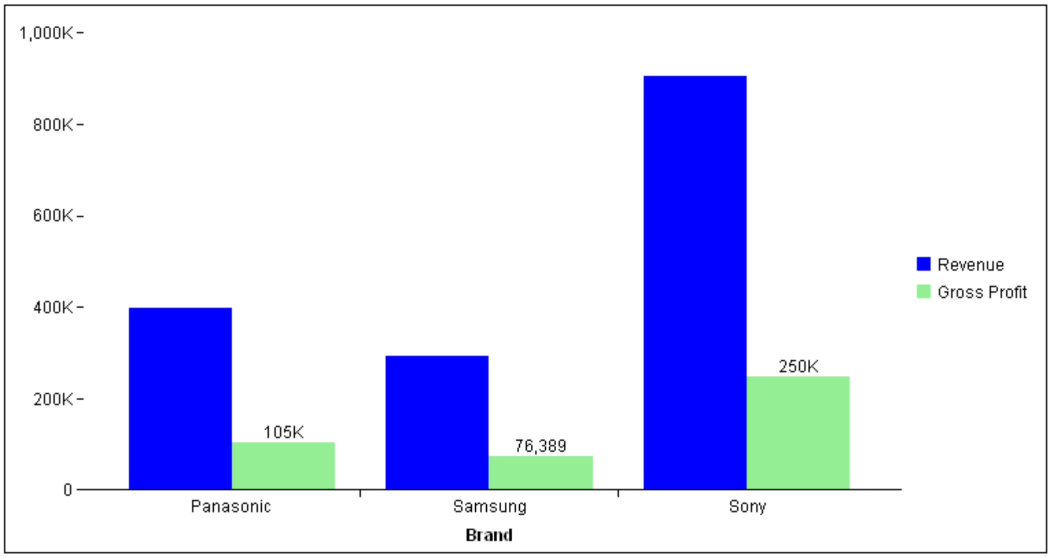
Is a string that defines the dash style. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Showing Data Text Labels for Specific Series

The following request generates data text labels for series 1, but not for series 0:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY BRAND
WHERE BRAND EQ 'Sony' OR 'Samsung' OR 'Panasonic'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=brand, bucket=x-axis,$
type=data, column=revenue_us, bucket=y-axis,$
type=data, column=gross_profit_us, bucket=y-axis,$
*GRAPH_JS
"xaxis": {"majorGrid": {"visible": false}},
"yaxis": {"majorGrid": {"visible": false}},
"series": [
    {"series": 0, "color": "blue"},
    {"series": 1, "color": "lightgreen", "dataLabels": {"visible": true}}]
*END
ENDSTYLE
END
```

The output is shown in the following image:

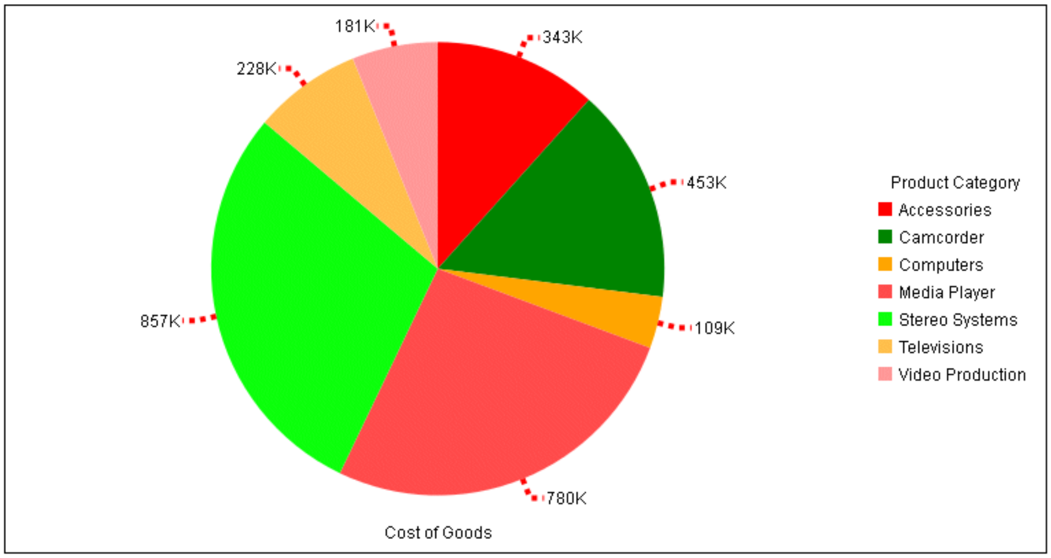


Example: Controlling the Appearance of Feeler Lines in a Pie Chart

The following request makes the feeler lines red and dashed, with a width of 4. In order to generate feeler lines, the data labels are placed outside the pie chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=measure,$
type=data, column=product_category, bucket=color,$
*GRAPH_JS
"series": [{"series": "all", "dataLabels": {
    "position": "outside", "visible": true, "feelerLine": {
        "visible": true, "width": 4, "color": "red", "dash": "4 4"}
    }}]
*END
ENDSTYLE
END
```

The output is shown in the following image:

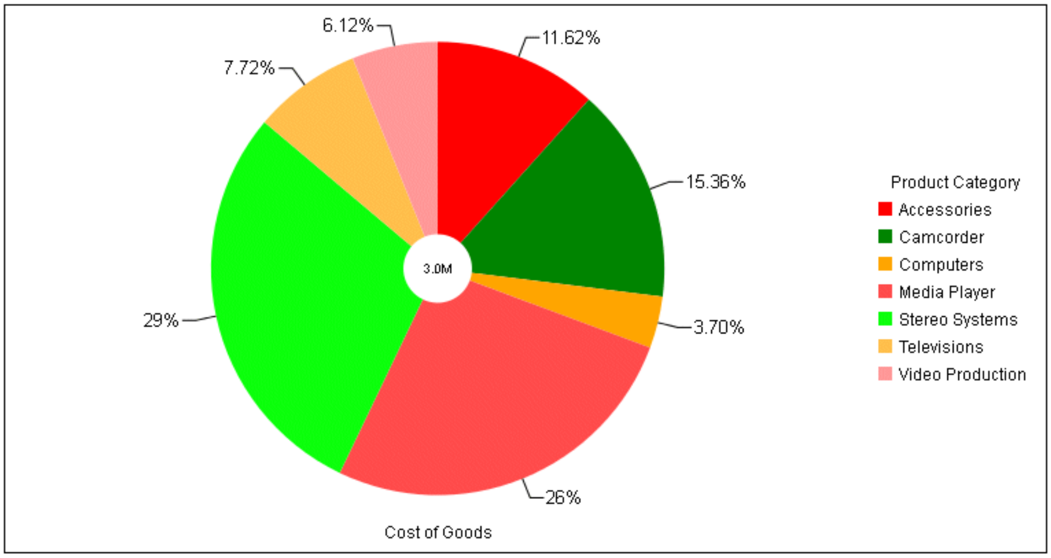


Example: Formatting Data Labels in a Pie Chart

The following request generates a pie chart. The data labels are made visible, are displayed in percent mode in 10pt Sans-Serif font, are positioned outside of the slices, and are formatted differently depending on the data value.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=measure,$
type=data, column=product_category,bucket=color,$
*GRAPH_JS
"pieProperties": {"totalLabel": {"visible": true}, "holeSize": 25},
"series": [{"series": "all", "dataLabels": {
    "visible": true,
    "position": "outside",
    "feelerLine": {"visible": true, "width": 1, "color": "black"},
    "content": "%",
    "font": "10pt Sans-Serif",
    "numberFormat": "[>.2]#,#%;[<0]-#.#%;[<=.2]#.##%;[=0]0"}}]
*END
ENDSTYLE
END
```

On the chart output, the data labels are outside of the pie with feeler lines:

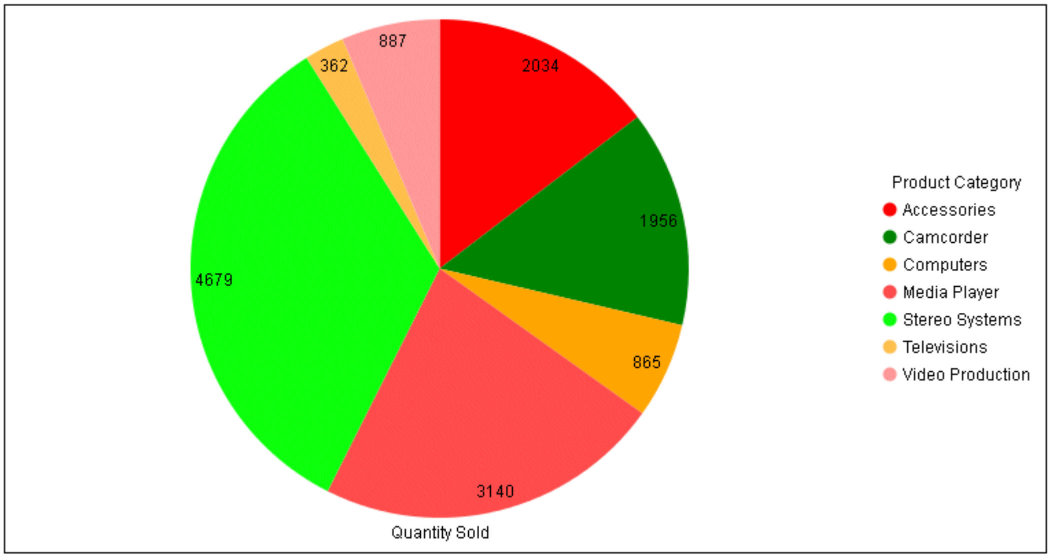


Example: Placing Data Labels Inside the Top Edge of Pie Slices

The following request generates a pie chart with the data labels inside the top edge of the slices.

```
GRAPH FILE WF_RETAIL_LITE
SUM QUANTITY_SOLD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=product_category, BUCKET=color, $
TYPE=DATA, COLUMN=quantity_sold, BUCKET=measure, $
*GRAPH_JS
"dataLabels": {"visible": true, "position": "insideTop"}
*END
ENDSTYLE
END
```

The output is shown in the following image:

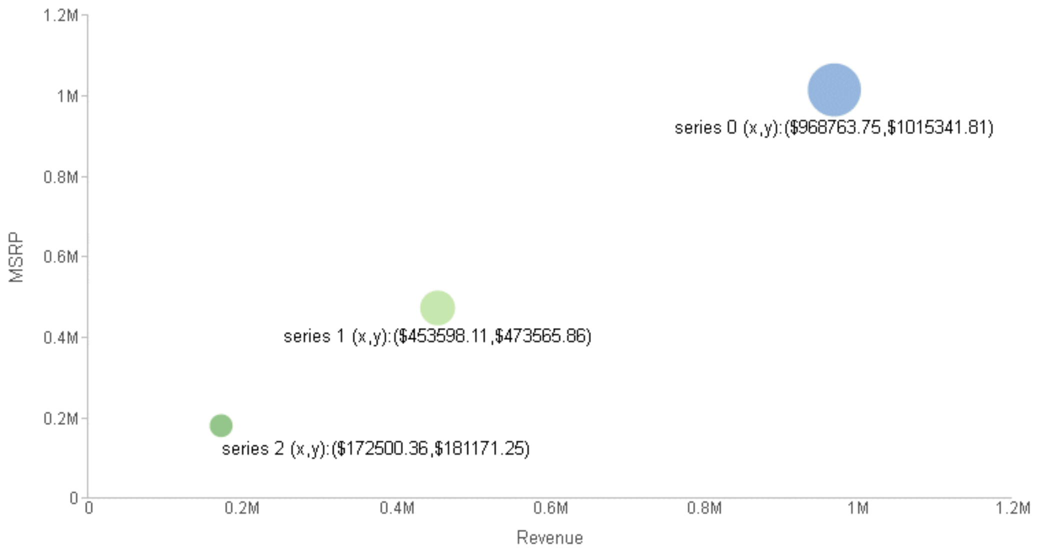


Example: Using a Callback Function to Format Data Labels

The following request makes the data labels visible, places them below the bubbles, and uses a function to return the label. The function takes the data values and series number as arguments and returns a label that has the series number and x,y values of the data, each prefixed with appropriate text:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_SUBCATEG
WHERE PRODUCT_SUBCATEG EQ 'iPod Docking Station' OR 'Blu Ray' OR 'Speaker Kits'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
type=data, column=revenue_us, bucket=x-axis,$
type=data, column=msrp_us, bucket=y-axis,$
type=data, column=discount_us, bucket=size,$
type=data, column=product_subcateg, bucket=color,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"legend": {"visible": "false"},
"series": [{"series": "all", "dataLabels": {
    "visible": true,
    "position": "bottom",
    "font": "10pt Sans-Serif",
    "content": function(d, s)
        {return "series " + s+" (x,y):" +"("+"$"+d.x+","+"$"+d.y+")";
    }}}]
*END
ENDSTYLE
END
```

The output is:

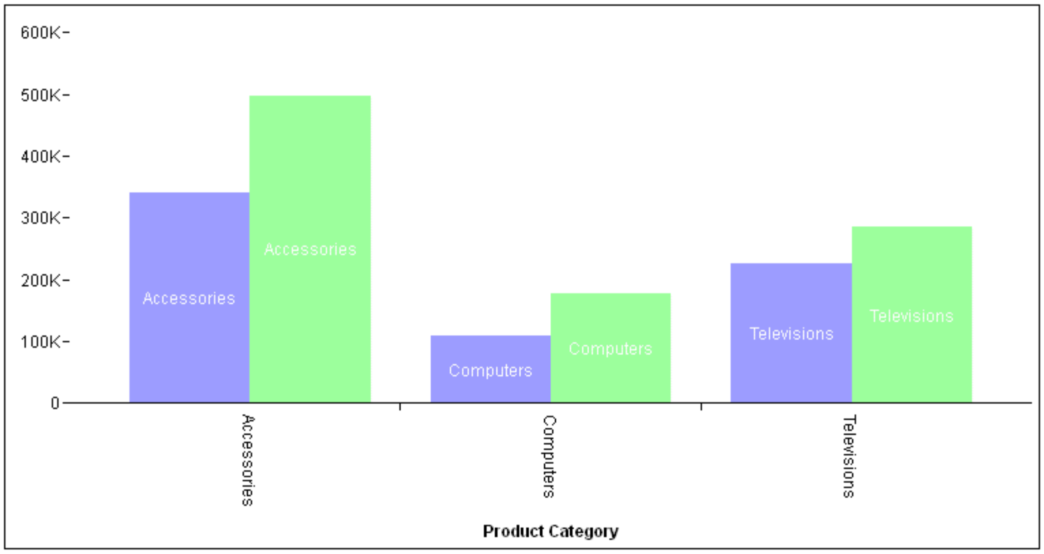


Example: **Using Group or Series Labels as Data Labels**

The following request uses the group labels as the data text and places them inside the risers:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
ACROSS PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Computers' OR 'Televisions'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"series": [{"series": "all", "dataLabels": {
    "visible": true, "color": "white",
    "position": "inside", "content": "groupLabel"}
    }]
*END
ENDSTYLE
END
```

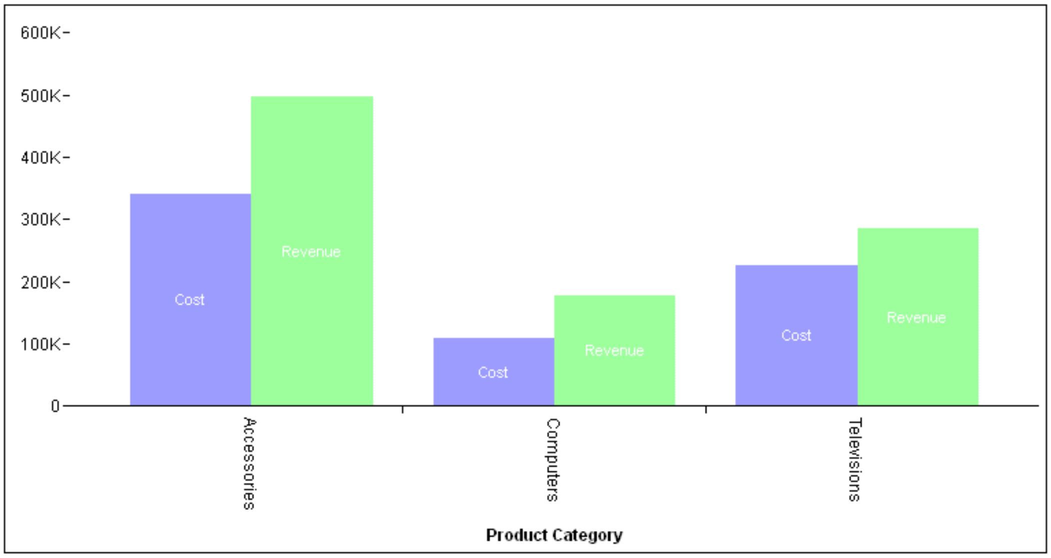
On the output, the group (sort field) label (product category) is used as the data label:



The following request uses the series labels as the data text. The series are generated from the display fields in the request. In this request, the default column headings are replaced with AS names:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US AS Cost
REVENUE_US AS Revenue
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Computers' OR 'Televisions'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"series": [{"series": "all", "dataLabels": {
    "visible": true, "color": "white", "font": "Bold 8pt Sans-Serif",
    "position": "inside", "content": "seriesLabel"}}]
*END
ENDSTYLE
END
```

On the output, the series (display field) AS name is used as the data label:

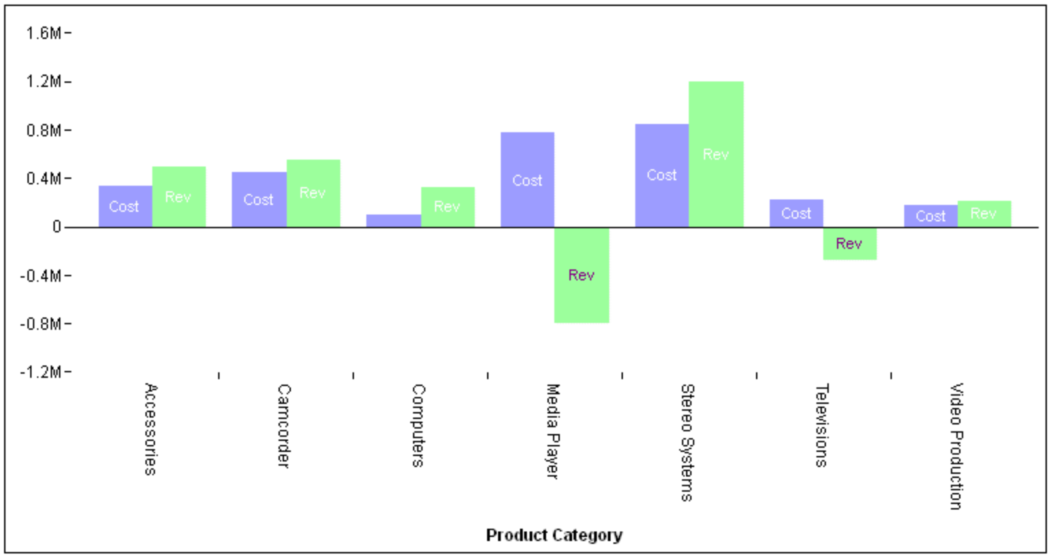


Example: Using a Different Color for Data Labels for Negative Risers

The following request uses the series labels as the data text and makes the data text for negative risers purple. The series are generated from the display fields in the request. In this request, the default column headings are replaced with AS names:

```
DEFINE FILE WF_RETAIL_LITE
REV= IF (COGS_US * 1.3) GT REVENUE_US THEN (REVENUE_US * -3) ELSE
(REVENUE_US *2) ;
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US AS Cost
REV AS Rev
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"series": [{"series": "all", "dataLabels": {
    "visible": true, "color": "white",
    "useNegativeColor": true, "negativeColor": "purple",
    "font": "Bold 8pt Sans-Serif", "position": "inside", "content":
"seriesLabel"}}]
*END
ENDSTYLE
END
```

The output is shown in the following image:



Deleting a Slice From a Pie Chart

The deleteSlice property deletes a slice from a pie chart (effectively, assigns the slice a transparent color).

Syntax: How to Delete a Slice From a Pie Chart

```
"series":
[
    {
      "series": number,
      "group": number,
      "deleteSlice": boolean,
   }
]
```

where:

`"series": number`

Is a zero-based series number or "all" to delete all slices. If the series does not exist in the chart, the property is ignored.

`"group": number`

For multi-pie charts, is an optional zero-based group (pie) number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the slice is deleted from all pies.

`"deleteSlice": boolean`

Can be:

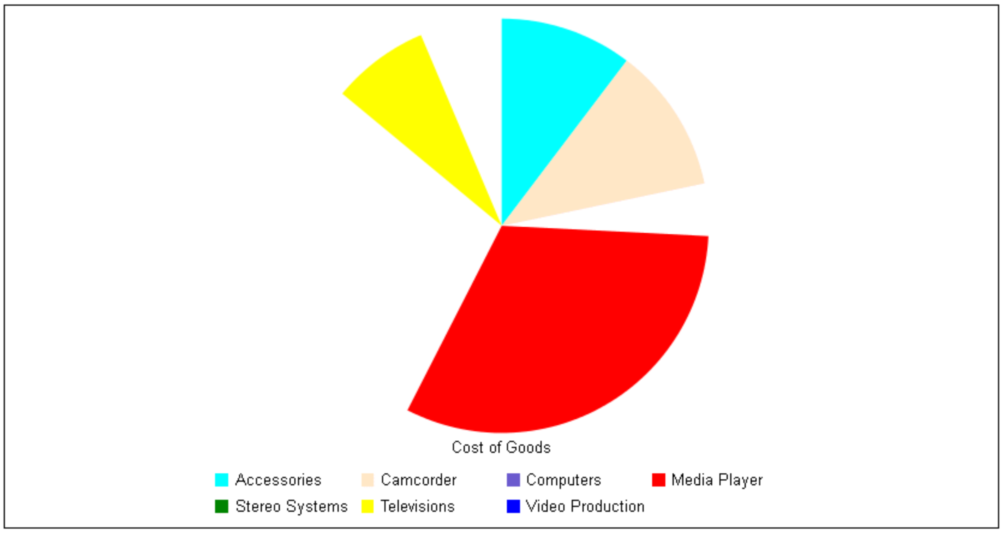
- ❑ true, to delete the slice.
- ❑ false, to restore the slice. This is the default value.

Example: Deleting Pie Slices

The following request generates a pie chart, but deletes the slices for series 2, 4, and 6:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "bisque"},
    {"series": 2, "color": "slateblue", "deleteSlice": true},
    {"series": 3, "color": "red"},
    {"series": 4, "color": "green", "deleteSlice": true},
    {"series": 5, "color": "yellow"},
    {"series": 6, "color": "blue", "deleteSlice": true}]
*END
ENDSTYLE
END
```

The output is;



Pushing a Slice Away From a Pie Chart

The explodeSlice property defines the distance (in pixels) to push a slice away from the pie chart.

Syntax: How to Push a Slice Away From a Pie Chart

```
series:
[
    {
      "series": number,
      "group": number,
      "explodeSlice": "value"    }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"group": number`

For multi-pie charts, is an optional zero-based group (pie) number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the property is applied to all pies.

`"explodeSlice": "*value*"`

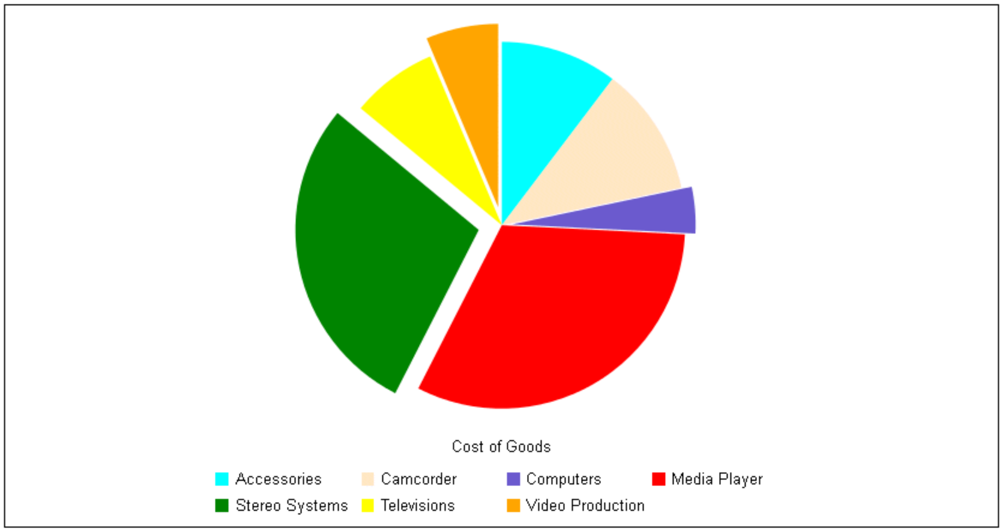
Is a number that defines the distance (in pixels) to push a slice away from the pie, or a percentage string enclosed in double quotation marks and including a percent symbol (for example, "25%") to explode a slice as a percentage of the radius of the pie. For example, if the pie is 200 pixels in diameter (100 pixels radius), "explodeSlice": "50%" will push the slice out 50 pixels from the center of the pie.

Example: Pushing Pie Slices Away From the Pie

The following request generates a pie chart and explodes the slices for series 2, 4, and 6:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "cyan"},
    {"series": 1, "color": "bisque"},
    {"series": 2, "color": "slateblue", "explodeSlice": 8},
    {"series": 3, "color": "red"},
    {"series": 4,"color": "green", "explodeSlice": 18},
    {"series": 5, "color": "yellow"},
    {"series": 6, "color": "orange", "explodeSlice": 14}]
*END
ENDSTYLE
END
```

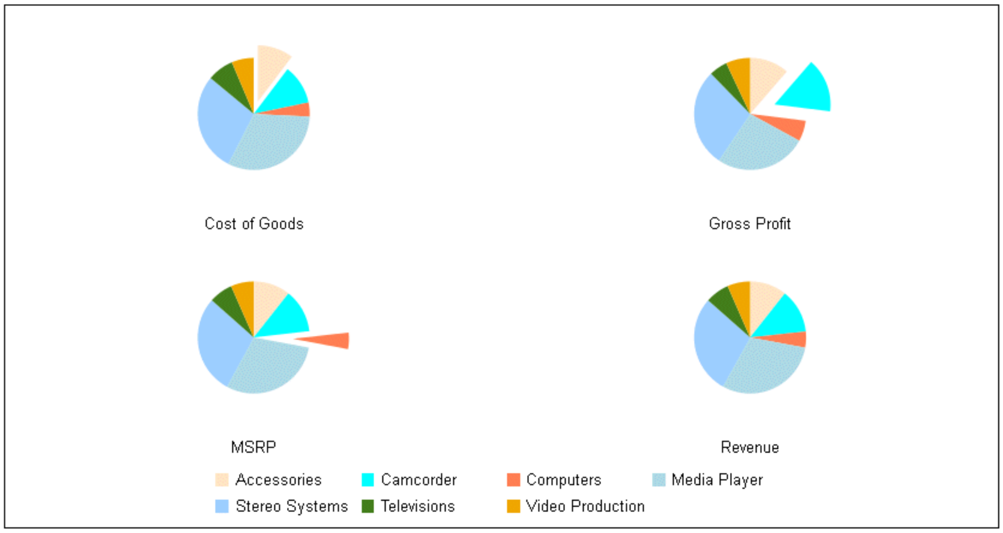
The output is:



The following request generates multiple pies and explodes the slices for series 0 in pie 0, for series 1 in pie 1, and for series 2 in pie 2:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US REVENUE_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "bisque"},
    {"series": 0, "group": 0, "explodeSlice": 10},
    {"series": 1, "color": "cyan"},
    {"series": 1, "group": 1, "explodeSlice": 20},
    {"series": 2, "color": "coral"},
    {"series": 2, "group": 2, "explodeSlice": 30},
    {"series": 3, "color": "lightblue"}]
*END
ENDSTYLE
END
```

The output is:



Assigning Labels to Individual Series

The label property assigns a label to an individual series that will be shown in the chart legend area. You can use the legend:labels property to define the format and color of the series label.

Syntax: How to Assign a Series Label

```
"series": [
   {
      "series": number,
      "label": "string",
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"label": "string"`

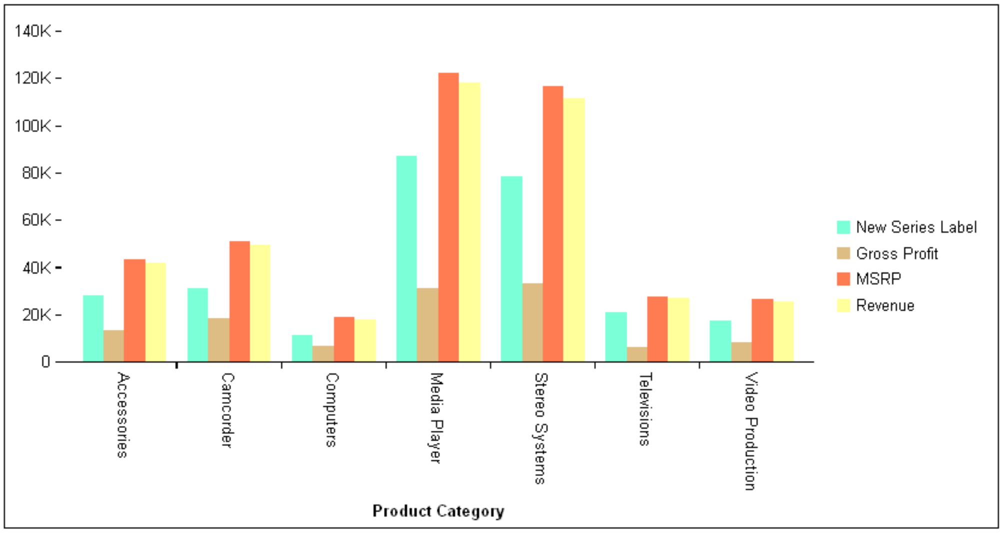
is a string that defines the series label.

Example: Assigning Series Labels

The following request generates a vertical bar chart and assigns a new label to series 0. You can see the new label in the legend, which is positioned to the right of the chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US MSRP_US REVENUE_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"position": "right"},
"series": [
    {"series": 0, "color": "aquamarine", "label": "New Series Label"},
    {"series": 1, "color": "burlywood"},
    {"series": 2, "color": "coral"}]
*END
ENDSTYLE
END
```

The output is:



Defining the Size, Border, Color, Shape, and Rotation of Series Markers

The marker properties define the size, border color, shape, and rotation of series markers.

Syntax: How to Define the Size, Border, Color, Shape and Rotation of Series Markers

```
"series": [
   {
      "series": number,  "color": "string",
      "group": number, // optional
      "marker": {
         "visible": boolean,  // Line Charts Only
         "size": number,
         "shape": "string",
         "rotation": number,
         "position": "string",
         "fillEffect": "string",
         "border": {"width": number, "color": "string", "dash": "string"}
      }
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"group": number`

Is an optional zero-based group number. If the group does not exist in the chart, the property is ignored. If a group number is not specified, the marker definition is applied to all risers in the series.

`"visible": boolean`

Controls the visibility of markers in line charts only. Valid values are:

- true, which makes the markers visible. This is the default value.
- false, which makes the markers not visible.

`"color": "string"`

Is a color for the marker, defined by a color name or numeric specification string, or a gradient defined by a string.

For information about defining colors and gradients, see *Colors and Gradients* on page 85.

`"size": number or "size": "n%"`

Is a number that defines the size of the marker in pixels or a percentage string. Percentages are relative to the minimum of the inner chart frame width and height.

`"shape": "`*`string`*`"`

Is a string that defines the shape of markers for a series. Valid values are:

- "arrow".
- "bar".
- "circle".
- "circleMinus" (a circle with a minus sign inside it).
- "circlePlus" (a circle with a plus sign inside it).
- "cross".
- "diamond".
- "fill" (supported for matrix marker charts. Fills each cell with the marker color, making it look similar to a heatmap).
- "fiveStar".
- "hexagon".
- "hourglass".
- "house".
- "pie" (for bubble and scatter charts only).
- "pin".
- "pirateCross".
- "plus".
- "rectangle".
- "rectangleThin".
- "ring".
- "sixStar".
- "square". This is the default value.
- "thinPlus".
- "tick".

- "triangle".

Note that bar, circlePlus, circleMinus, cross, and tick markers require a border width and color.

`"rotation": `*`number`*

Is a number between 0 and 360 that defines the angle (in degrees) of the marker.

`"position": "`*`string`*`"`

For bullet charts only, defines the position of the marker relative to data text. Valid values are:

- "bottom", which draws the data text label above the marker.
- "middle", which draws the data text label according to the general property dataLabels:position.
- "top", which draws the data text label below the marker.

`"fillEffect": "`*`string`*`"`

Defines the fill effect for markers. Valid values are:

- A percentage string that defines the transparency of the marker fill color. The default value is "100%" (fully opaque).
- undefined, which uses the existing marker border color and fill color.
- "seriesHollow", for which the marker fill area is hollow (transparent), and the marker border is the color set by the series:color property. You must define a non-zero border width.
- "seriesFill", for which the marker draws with no border, and the fill is the same as the series:color property.
- "seriesWhite", for which the marker fill area is white, and the marker border is the color set by the series:color property. You must define a non-zero border width.
- "seriesAuto", which causes the chart engine to fill the markers with the following fill:

  "seriesHollow" for scatter and radar charts.

  "seriesLighten" for bubble, map, and polar charts.

  "seriesWhite" for line charts.

  "seriesFill" for other types of charts, which uses the series color as the fill.

`"border":`

Defines the properties of the marker border.

`"width":` *`number`*

Is a number that defines the width of the border in pixels.

`"color": "`*`string`*`"`

Is a color for the marker border defined by a color name or numeric specification string.

`"dash": "`*`string`*`"`

Is a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Note:

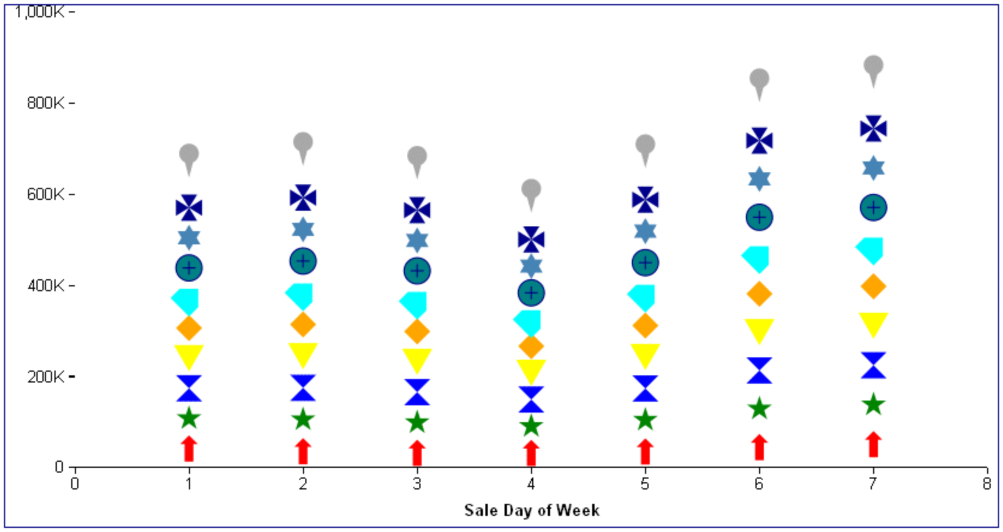
- The bar, cross, circlePlus, circleMinus, and tick marker shapes require a border width and color.
- The special pie marker shape is for bubble and scatter charts only and requires the use of "series": "all", {"marker": {"shape": "pie"}}. For details and examples, see bubble and scatter chart properties in *Chart-Specific Properties* on page 491.
- Markers in a Scatter Chart cannot be connected.
- The ring marker can be any size, but the thickness of the ring is hard-coded to an aesthetically pleasing percentage The center is transparent, but marker fill properties for the ring are supported.
- The rectangleThin marker is like a tick mark whose size can be varied and supports the marker fill and rotation properties (the marker can be changed from horizontal to vertical using rotation: 90).

Example: Defining Series Markers

The following request generates a scatter chart with different marker shapes:

```
DEFINE FILE WF_RETAIL_LITE
COGS1 = COGS_US;
COGS2 = COGS1 +500;
COGS3 = COGS1 +1000;
COGS4 = COGS1 +1500;
COGS5 = COGS1 +2000;
COGS6 = COGS1 +2500;
COGS7 = COGS1 +3000;
COGS8 = COGS1 +3500;
COGS9 = COGS1 +4000;
COGS10 = COGS1 +4500;
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US COGS2 COGS3 COGS4 COGS5 COGS6 COGS7 COGS8 COGS9 COGS10
ACROSS TIME_DAYOFWEEK
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERS
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"color": "navy"},
"legend": {"visible": false},
"xaxis": {"labels": {"rotation": 0}},
"series": [
    {"series": 0, "color": "red", "marker": {"shape": "arrow", "size": 20}},
    {"series": 1, "color": "green", "marker": {"shape": "fiveStar", "size":
20}},
    {"series": 2, "color": "blue", "marker": {"shape": "hourglass", "size":
20}},
    {"series": 3, "color": "yellow", "marker": {"shape": "triangle",
"size": 20}},
    {"series": 4, "color": "orange", "marker": {"shape": "diamond", "size":
20}},
    {"series": 5, "color": "cyan", "marker": {"shape": "house", "size":
20,"rotation": 45}},
    {"series": 6, "color": "teal", "marker": {
        "shape": "circlePlus", "size": 20, "border": {"width": 1, "color":
"navy"}}},
    {"series": 7, "color": "steelblue", "marker": {"shape": "sixStar",
"size": 20}},
    {"series": 8, "color": "darkblue", "marker": {"shape": "pirateCross",
"size": 20}},
    {"series": 9, "color": "darkgrey", "marker": {"shape": "pin", "size":
50}}]
*END
ENDSTYLE
END
```

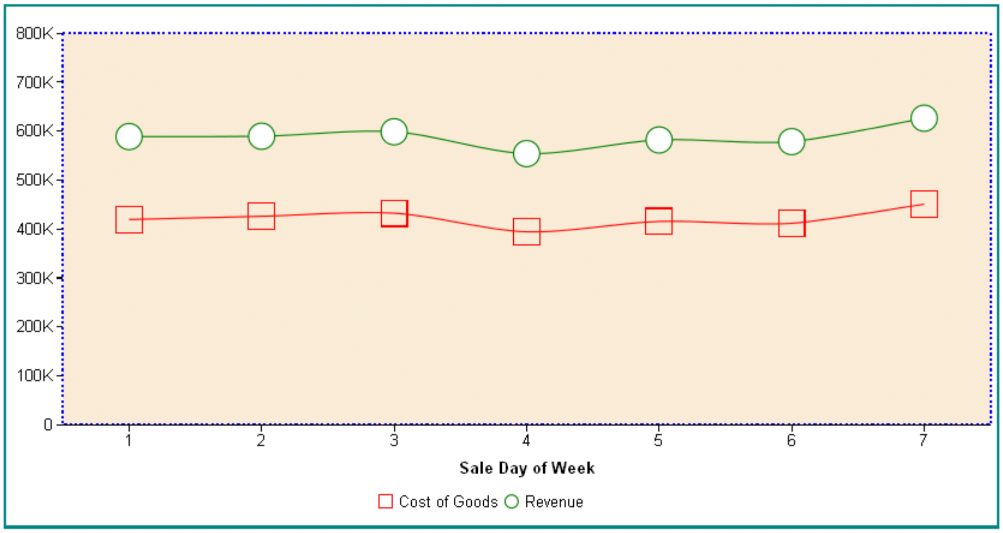
On the output, note that the house marker shape is rotated 45 degrees, and that the circlePlus marker shape has a border width and color, as required:



In the following request, series 0 (zero) has the fill effect seriesHollow, and series 1 has the fill effect seriesWhite. In each case a border is required:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
ACROSS TIME_DAYOFWEEK
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"xaxis": {"labels": {"rotation": 0}},
"border": {"width": 2, "color": "teal"},
"chartFrame": {"fill": {"color": "antiquewhite"},
    "border": {"width": 2, "color": "blue", "dash": "2 2"}},
"blaProperties": {"lineConnection": "curved"},
"series": [
    {"series": 0, "color": "red", "marker": {
        "border": {"color": "red", "width": 1},
        "shape": "square", "size": 20, "fillEffect": "seriesHollow"}},
    {"series": 1, "color": "green", "marker": {
        "border": {"color": "green", "width": 1},
        "shape": "circle", "size": 20, "fillEffect": "seriesWhite"}}]
*END
ENDSTYLE
END
```

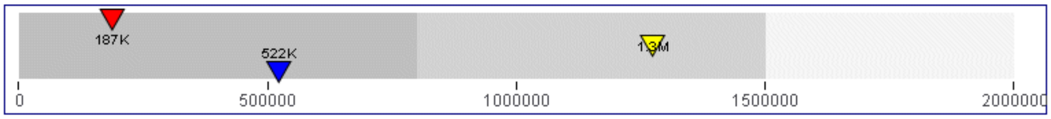
The output is:



The following request generates a bullet chart and sets the positions and shapes of the markers:

```
GRAPH FILE WF_RETAIL_LITE
SUM MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ  'Accessories' OR 'Computers' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET VAXIS 80
ON GRAPH SET LOOKGRAPH CUSTOM
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"chartType": "bullet",
"border": {"width": 1, "color": "navy"},
"bulletProperties": {"drawFirstValueAsBar": false},
"dataLabels": {"visible": true, "font": "6pt"},
"yaxis": {"colorBands": [
    {"start": 0, "stop": 800000, "color": "silver"},
    {"start": 800000, "stop": 1500000, "color": "lightgrey"},
    {"start": 1500000, "stop": 2000000, "color": "whitesmoke"}]},
"series": [
    {"series": "all", "showDataValues": true},
    {"series": 0, "group": 0, "color": "blue",
    "marker": {"size": 15, "position": "bottom", "shape": "triangle"}},
    {"series": 0, "group": 1, "color": "red",
    "marker": {"size": 15, "position": "top", "shape": "triangle"}},
    {"series": 0, "group": 2, "color": "yellow",
    "marker": {"size": 15, "position": "middle", "shape": "triangle"}}]
*END
ENDSTYLE
END
```

On the output, the yellow marker is positioned in the middle of the data text, the red marker is above the data text, and the blue marker is below the data text:

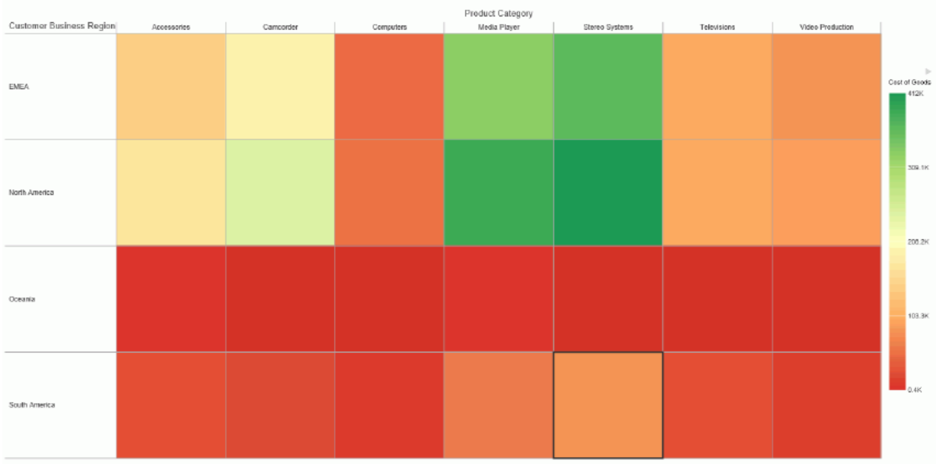


Example: Filling Cells in a Matrix Marker Chart

The following request generates a matrix marker chart with the *fill* marker shape.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH MARKER
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=row, $
TYPE=DATA, COLUMN=N2, BUCKET=column, $
TYPE=DATA, COLUMN=N3, BUCKET=color, $
*GRAPH_JS
"series": [
           {
            "series": "all",
                 "marker": {
                            "shape": "fill"
                           }
            },
         ]
*END
ENDSTYLE
END
```

The output is shown in the following image.

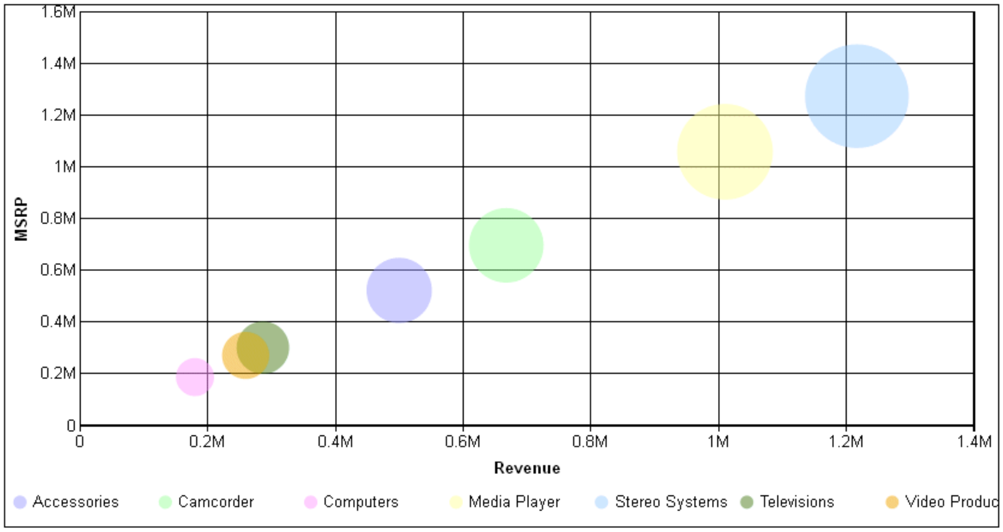


Example: Making Bubble Markers Partially Transparent

In the following request, the "fillEffect":"50%" property makes the bubble markers partially transparent:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": "all", "marker": {"shape": "circle", "fillEffect": "50%"}}]
*END
ENDSTYLE
END
```

On the output, the bubble markers are 50% transparent, as shown on the following image:

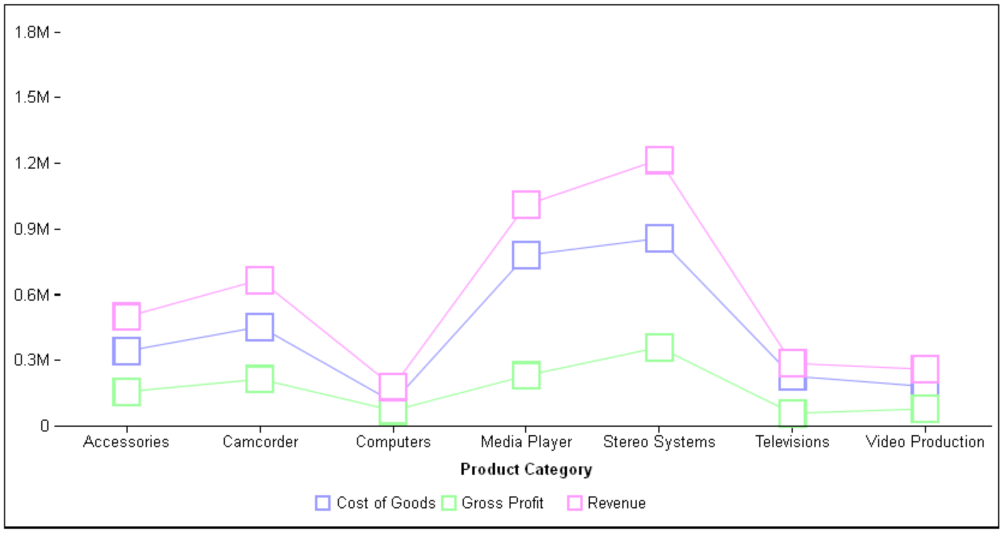


Example: Using Automatic Fill Effect for Markers

The following request generates a line chart with the "seriesAuto" fill effect which, for line charts, is equivalent to "seriesWhite":

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": "all", "marker": {"border": {"width": 2},
        "size": 20, "fillEffect": "seriesAuto"}}]
*END
ENDSTYLE
END
```

The output is shown on the following image:



Defining the Shapes of Risers for a Series in Bar, Line, and Area Charts

The riserShape property is used in conjunction with the comboCharts properties to define a combination chart (that is, a chart that can consist of a combination of bar risers, area risers, and line risers). In bar, line, and area charts, these properties can be used to define the shape of risers (bar, line, or area) for each series. The comboCharts properties control the layout (stacked, absolute, percent, or sideBySide) of the risers for each series.

Syntax: How to Define the Shape of Risers in Bar, Line, and Area Charts

```
"series":
[
    (
      "series": number,
      "riserShape": "string",
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"riserShape": "string"`

Is a string that defines the shape of the riser for the specified series. Valid values are:

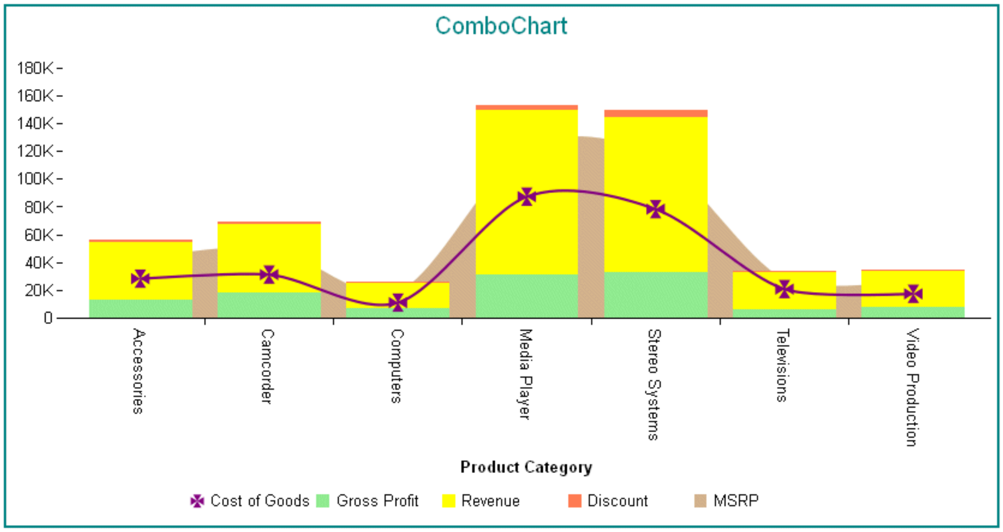
- "bar".
- "line".
- "area".

Example: Defining Riser Shapes in Bar, Line, and Area Charts

The following request generates a vertical bar chart in which the comboCharts properties define the bar series layout as stacked, the line series layout as absolute, and the area series layout as undefined. The properties for each series define the riser shape for that series:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US DISCOUNT_US MSRP_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
"border": {"width": 2, "color": "teal"},
"title": {"visible": true, "text": "ComboChart", "font": "14pt Sans-Serif",
"color": "teal"},
"blaProperties": {
    "lineConnection": "curved",
    "comboCharts": {"barSeriesLayout": "stacked",
        "lineSeriesLayout": "absolute",
        "areaSeriesLayout": "undefined"}},
"series": [
    {"series": 0, "color": "purple", "riserShape": "line", "border":
{"width": 2},
        "marker": {"shape": "pirateCross", "size": 14, "visible": true}},
    {"series": 1, "color": "lightgreen", "riserShape": "bar"},
    {"series": 2, "color": "yellow", "riserShape": "bar"},
    {"series": 3, "color": "coral", "riserShape": "bar"},
    {"series": 4, "color": "tan", "riserShape": "area"}]
*END
ENDSTYLE
END
```

The output is:



Connecting the Markers in Scatter Charts

When analyzing patterns in scatter charts, lines connecting the points can help illustrate the pattern in the data. You can connect the points in a scatter chart using the connectMarkers property when the BY field is assigned to the detail attribute category, not the color category.

Syntax: How to Connect Markers in a Scatter Chart

```
"series":
 [{
  "series": 0,
  "connectMarkers": boolean,
  "border": {
            "width": number,
             "color": "string"
            }
  }
 ]
```

where:

`"connectMarkers": boolean`

Specifies whether the markers should be connected. Valid values are *true*, which connects the markers, and *false*, which does not connect the markers. False is the default value.

Note that the BY field must be assigned to the detail attribute category in order for markers to be connected.

`"border"`

Defines the properties of the line connecting the markers.

`"width": `*`number`*

Specifies the width of the line in pixels.

`"color": "`*`string`*`"`

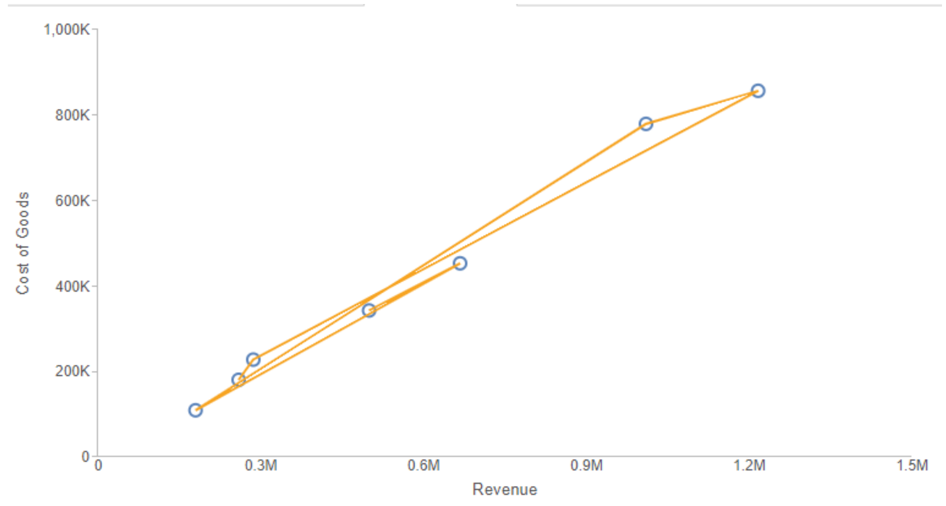
Is a color specification string that defines the color of the line.

Example: **Connecting markers on a Scatter Chart**

The following request generates a scatter chart with a 2-pixel wide orange line connecting the markers.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=detail, $
*GRAPH_JS
"series": [{
    "series": 0,
    "connectMarkers": true,
    "border": {
        "width": 2,
        "color": "orange"
    }
}]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Using Named Fill Patterns for Series Risers and Markers

Commonly used fill patterns for series risers and markers have been assigned names.

Syntax: How to Specify Named Fill Patterns for Risers and Markers

The pattern properties replace the color specification string for a series.

```
{"series": number,
      "color":{
              "type": "pattern",
              "color": "string",
              "backgroundColor": "string",
              "lineWidth": number,
              "antialias": true,
              "shape": "string",
              "size": number,
              "pad": number
              }
}
```

where:

`"color": "string"`

Is the color of the fill pattern. If undefined, the pattern is not visible.

`"backgroundColor": "string"`

Is the background color between the fill characters.

`"lineWidth": number`

Is the width of a stroke in the pattern (if any), in pixels. The default value is 1.

`"antialias": boolean`

Defines whether or not the lines are anti-aliased (smoothed). If undefined, automatically chooses an appropriate setting based on the chosen pattern. Can be one of the following values.

- true, which anti-aliases the pattern. This is the default value.
- false, which does not anti-alias the pattern.
- undefined, which lets the chart engine choose an appropriate setting based on the chosen pattern.

`"shape": "string"`

Can be one of the following named fill patterns.

- **hline.** Repeated horizontal lines.
- **vline.** Repeated vertical lines.
- **hatch.** Repeated hatch marks (#).
- **slash.** Repeated slashes.
- **backslash.** Repeated backslashes.
- **diagonalhatch.** Hatch marks repeated diagonally.
- **dots.** Repeated dots.
- **diagonaldots.** Dots repeated diagonally.
- **solid.** A solid fill of the specified color.
- **empty.** No fill.

`"size": number`

Defines the width and height of the space in which to repeat the pattern, in pixels. The default value is 20.

`"pad": number`

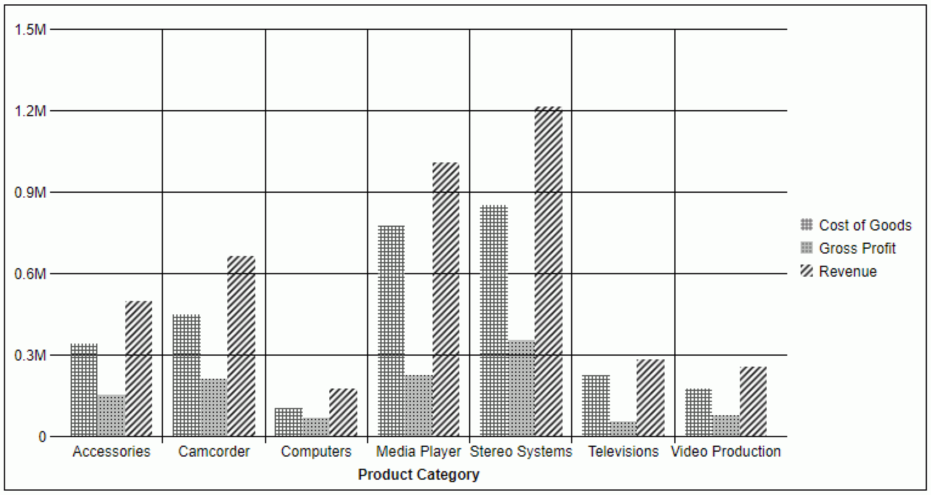
Defines the number of pixels of empty space to add around the repeated pattern. The default value is 1.

Example: Using Named Fill Patterns in Bar Chart Risers

The following request defines series 0 with the hatch fill pattern, series 1 with dots, and series 2 with slashes.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=GROSS_PROFIT_US, bucket=y-axis,$
type=data, column=REVENUE_US, bucket=y-axis,$
type=data, column=PRODUCT_CATEGORY, bucket=x-axis,$
*GRAPH_JS
series:[
{"series": 0, "color": {"type": "pattern", "color": "#555",
              "shape": "hatch", "pad": 2, "lineWidth": 1}} ,
{"series": 1, "color": {"type": "pattern", "color": "black",
              "shape": "dots", "pad": 1.5, "lineWidth": 1.5}} ,
{"series": 2, "color": {"type": "pattern", "color": "black",
              "shape": "slash", "pad": 2.5, "lineWidth": 1.5}}
]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Generating Curved Corners on Bar Chart Risers

By default, bar chart risers have square corners. You can use the series property cornerRadius to generate curved corners for a more modern look.

Note: Legend markers are not curved, even if the chart risers are curved.

Syntax: How to Generate Bar Chart Risers With Curved Corners

```
"series": number, "border": {"cornerRadius": {"x": value, "y": value}}
```

where:

`"series": number`

Specifies a series number for which the corners are being rounded, or "all", to apply the cornerRadius properties to all risers. The corner shape specified for series "all" is applied to any series that does not have its own cornerRadius property.

`"border"`

Defines the properties of the border around the riser.

`"cornerRadius": {"x": value, "y": value}`

Defines the properties of the corners of the riser.

The cornerRadius property can consist of one value or two values. A single zero (0) value generates square corners. This is the default corner shape. Rounded corners can be circular or elliptical. The x and y values denote the size of the circle radius or the semi-major and semi-minor axes of the ellipse, where x and y are orientation-aware and depend on chart orientation. x represents the ordinal axis, and y represents the numeric axis.

Valid values for x and y are:

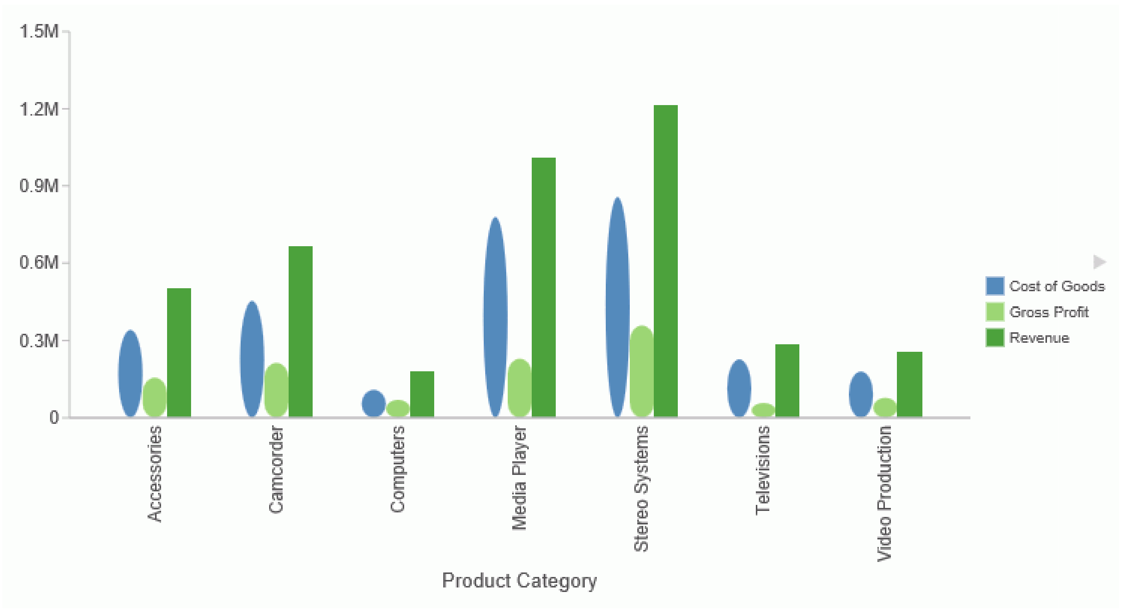
- A number that specifies the radius of the corner in pixels.
- A percent string such as "20%", that represents a percentage of the ordinal or numeric axis of the riser.
- A CSS length string such as "5em" or "10px" that specifies the radius of the corner in pixels. For information about CSS length strings, see *https://developer.mozilla.org/en-US/docs/Web/CSS/length*.
- An object with "x" and "y" properties (where "x" and "y" can be any of the above) to specify the corner radius along the ordinal axis (x) and numeric axis (y) of the riser.

Example: Controlling Corner Shape of Bar Chart Risers

In the following request, series 0 (COGS_US) has elliptical corners, series 2 (REVENUE_US) has square corners, and series 1 (GROSS_PROFIT_US) has the round corners specified for series "all".

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET STYLE *

INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$

TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis, $

*GRAPH_JS
series: [{series: 'all', border: {cornerRadius: 10}},
{series: 0, border: {cornerRadius: {x: "10%", y: "30%"}}},
{series: 2, border: {cornerRadius: 0}}
]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Defining or Hiding a Tooltip for Risers

The tooltip property defines a tooltip for one or more risers. The tooltip string is shown when the mouse hovers over a riser. A tooltip can be assigned to all risers and markers, a single riser and marker (identified by a series and group), or all risers and markers in a series.

You can prevent the tooltip for displaying for specific risers using the *null* property for the series tooltip.

When chart objects, such as risers, are in close proximity, and the tooltip has a menu (such as a drill-down menu), clicking an object to display its tooltip and then hovering the mouse over a nearby object may not change the tooltip being displayed to the one associated with the new object.

You can use the chart template engine to customize tooltips with macros. For information see *Introduction to JSON Properties for HTML5 Charts* on page 83.

You can also use autoNumberFormats to apply a number format to tooltips. For information about number formats, see *Formatting Numbers* on page 108.

Note:

- If you are using chart attribute syntax, you must use a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet in order to override the server-generated tooltips. For information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.
- Chart requests that do not use chart attribute syntax display an older style of tooltips than the new style tooltips generated by requests that use chart attribute syntax.

Syntax: How to Define a Tooltip for Risers

```
"series": [
{
   "series": number,
   "group": number, // Optional
   "tooltip": "string"}
]
```

where:

`"series": number`

Is a zero-based series number, "all" to assign the tooltip to all series, or "reset" to remove default values. If the series does not exist in the chart, the property is ignored.

`"group": number`

Is an optional zero-based group number. If not specified, the property is applied to all groups in the series.

`"tooltip": "string"`

Is a string, or a function that returns a string, to display when the mouse hovers over the riser. You can inhibit the display of the tooltip for specific series using the value *null* for the tooltip object.

If htmlTooltip is enabled, you can use the special string "auto" to automatically populate the tooltip with series labels, group labels, and values. The content will be different for different chart types. See *Generating HTML Tooltips (htmlToolTip)* on page 788 for examples.

A callback function is invoked with three arguments: value, series ID, and group ID. For example:

```
tooltip: function(value, series, group) {
 return this.title.text + ", value: " + value + ", s: " + series + ",
g: " + group;
}
```

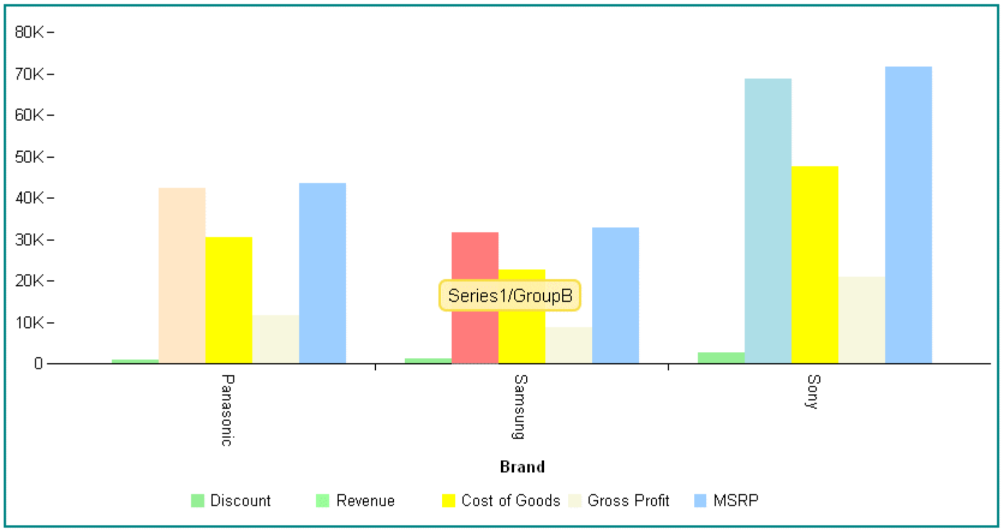
Note that single and double quotation marks are both supported to enclose text in the function definition.

Example: Defining Series Tooltips

The following request assigns a tooltip to series 1, group 1:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US REVENUE_US COGS_US GROSS_PROFIT_US MSRP_US
BY BRAND
WHERE BRAND EQ 'Sony' OR 'Samsung' OR 'Panasonic'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "group": 0, "color": "bisque"},
    {"series": 1, "group": 1, "color": "red", "tooltip": "Series1/GroupB"},
    {"series": 1, "group": 2, "color": "lightblue"},
    {"series": 2, "color": "yellow"},
    {"series": 3, "color": "beige"}]
*END
ENDSTYLE
END
```

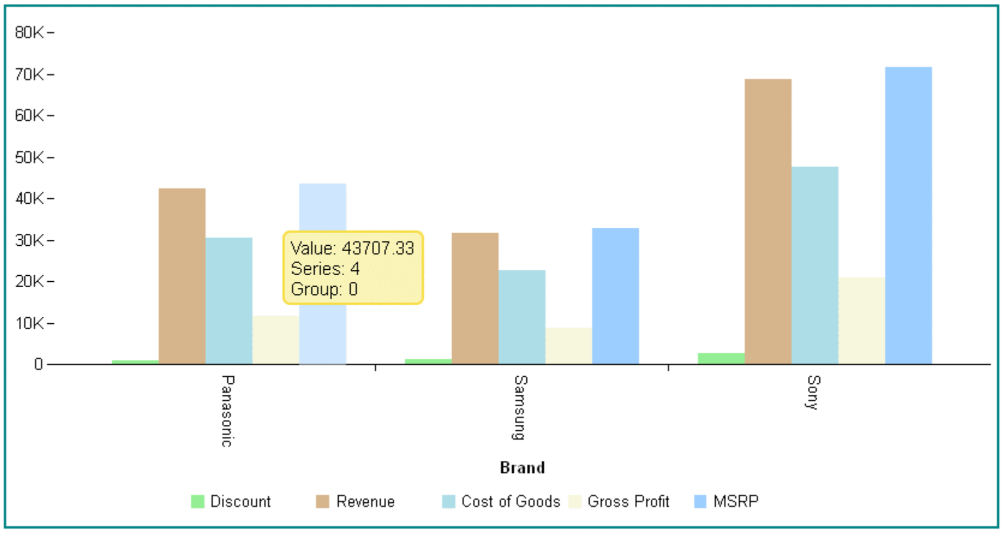
The tooltip displays when the mouse hovers over the specified riser:



The following request uses a callback function to generate the tooltips. In order to remove all default tooltips so that the callback function will be used, you must use series "reset" or set each individual series tooltip to undefined. HTML tags can also be used in the function. In this example, <br> tags are used to generate line breaks in the tooltips:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US REVENUE_US COGS_US GROSS_PROFIT_US MSRP_US
BY BRAND
WHERE BRAND EQ 'Sony' OR 'Samsung' OR 'Panasonic'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"series": [
    {"series": "reset", "tooltip": function(v, s, g) {return "Value: " + v
+"<br>Series: " + s +"<br>Group: " + g;}},
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "tan"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "beige"}]
*END
ENDSTYLE
END
```

The output shows the custom tooltip returned by the callback function:



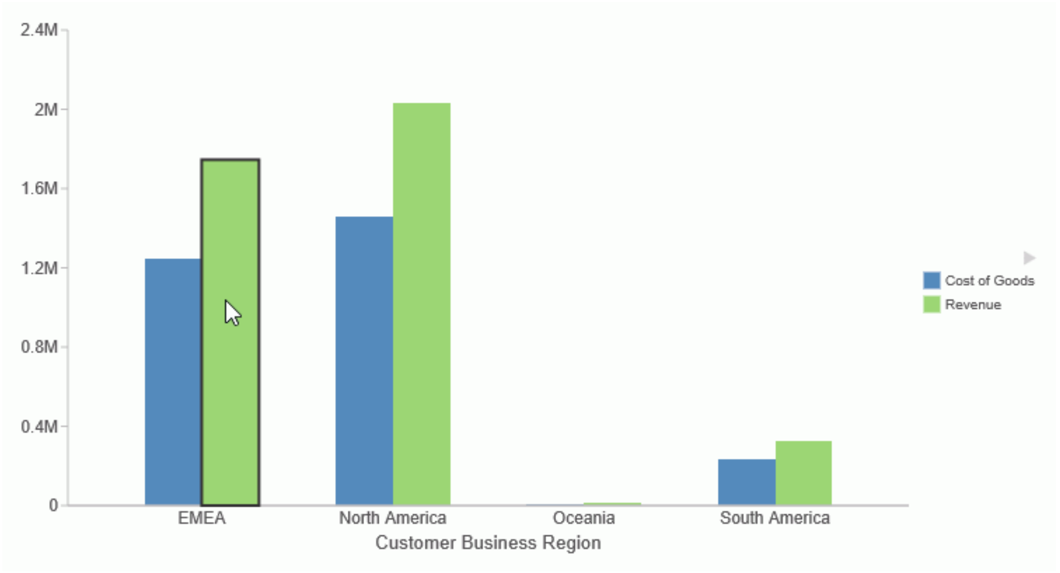
Note: When a tooltip is defined for one or all series, the htmlToolTip property can be used to define HTML-based (div) style tooltips for any chart tooltips. For information on HTML-based tooltips, see *Generating HTML Tooltips (htmlToolTip)* on page 788.

Example: Turning Tooltips Off for a Series

The following request turns tooltips off for series 1.

```
GRAPH FILE wf_retail_lite
SUM COGS_US REVENUE_US
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"series": [ {"series": 1, "tooltip": null} ],
*END
ENDSTYLE
END
```

The output is shown in the following image. Although the mouse is hovering over a riser for series 1, no tooltip displays.



Syntax: How to Define Tooltip Submenus

Tooltips can be defined as arrays of submenus with entries and children. Nested submenus, in which the child entries have their own submenus, are supported. You can use htmlToolTip properties to style these menus, as described in *Generating HTML Tooltips (htmlToolTip)* on page 788.

```
"series":[
 {"series": number,
   "tooltip": [
      {
    "entry": "Submenu1",
    "children": [
                "Child1",
                "Child2"
                ]
      },
   {
    "entry": "Submenu2",
    "children": [
                 {
                 "entry": "Nested Submenu",
                 "children": [
                             "NestedChild1",
                             "NestedChild2"
                             ]
                }
                  ]
    }

       ]
  }]
```

where:

`"series": number`
: Specifies the series ID for the tooltip.

`"entry": "Submenu1", "entry": "Submenu2" ...`
: Defines the text of a tooltip submenu. You can use a combination of tooltip components, such as literal values and template macros.

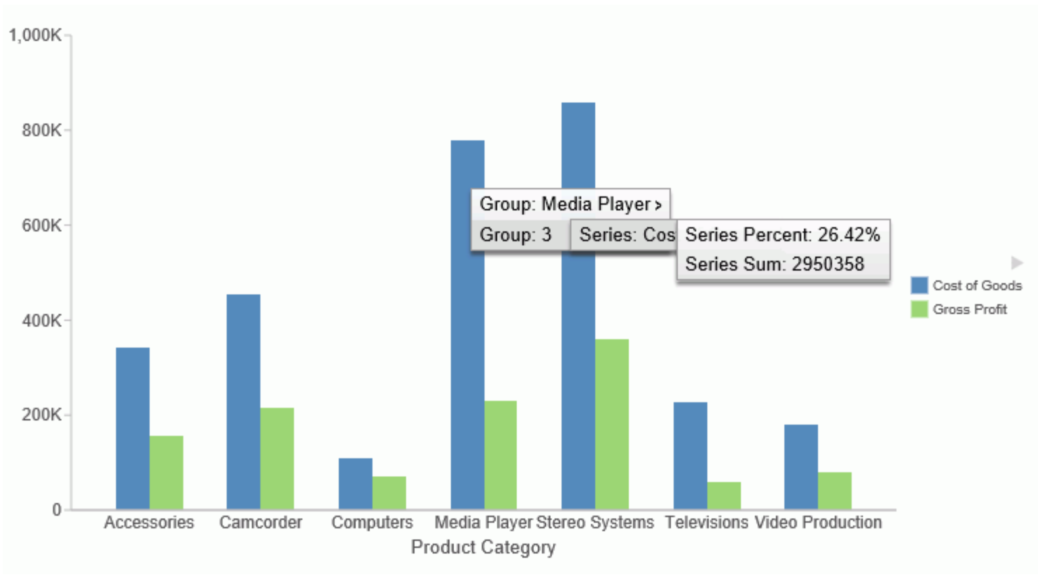
`"children": [ "Child1", "Child2" ... ]`
: Defines the text of submenu children. A child can also have submenus. You can use a combination of tooltip components, such as literal values and template macros.

Example: Defining Tooltip Submenus

In the following request, series 0 has two submenus. The second submenu is a nested submenu.

```
GRAPH FILE wfretail82/wf_retail_lite
SUM COGS_US
GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"series":[
 {"series": "0",
   "tooltip": [
      {
          "entry": "Group: {{group_label}}",
          "children": [
                "Series Label: {{series_label}}",
                "Series ID: {{series_id}}"
              ]
      },
   {
    "entry": "Group: {{group_id}}",
    "children": [
                 {
                 "entry": "Series: {{series_label}}",
                 "children": [
                          "Series Percent: {{series_percent}}",
                    "Series Sum: {{series_sum}}"
                             ]
                }
                  ]
    }

         ]
  }
 ]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Drawing a Series Trendline in Bubble and Scatter Charts

The trendline properties draw a trendline and equation label for individual series in bubble and scatter charts.

Note: You can also draw a trendline for all series using the global trendline property. For information, see *Drawing Trendlines in Bubble and Scatter Charts (trendline)* on page 812.

Syntax: How to Draw a Series Trendline For Series in Bubble and Scatter Charts

```
"series": [
{
  "series": number,
      "trendline": {
         "enabled": boolean,
         "mode": "string",
         "order": number,
         "lineStyle": {
             "width": number,
             "color": "string",
             "dash": "string"
                   },
         "equationLabel": {
            "visible": boolean,
            "font": "string",
            "color": "string",
            "mode": "string"
         }
      }
}
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"enabled": boolean`

Enables or disables the trendline. Valid values are:

- true, which enables the trendline.
- false, which disables the trendline. This is the default value.

`"mode": "string"`

Is string that defines the trendline mode (the type of equation used to generate the trendline). Valid values are:

- "exponential".
- "geometric".
- "hyperbolic",
- "linear". This is the default value if the trendline is visible.
- "logarithmic".

- "logQuadratic".
- "modExponential".
- "modHyperbolic".
- "polynomial".
- "quadratic".
- "rational".
- undefined. This is the default value if the line is not visible. If the trendline is visible but "mode" is undefined, "linear" mode will be used.

`"order":` *number*

Applies to polynomial mode. Specifies the degree of the polynomial. The default value is 3.

`"lineStyle":`

Defines the properties of the trendline.

`"width":` *number*

Defines the width of the trendline in pixels. The default value is 1.

`"color": "`*string*`"`

Is a color specification string that defines the color of the trendline. The default value is undefined, which uses the series color for the trendline color.

`"dash": "`*string*`"`

Is a string that defines the dash style of the trendline. Specify the length of a dash in pixels followed by the number of pixels between dashes (for example, "1 1", which generates a dotted line). The default value is no dash ("").

`"equationLabel"`

Defines the properties of the equation label for the trendline.

`"visible":` *boolean*

Controls the visibility of the equation label. Valid values are:

- true, which shows the equation label.
- false, which does not show the equation label. This is the default value.

`"font": "`*string*`"`

Is a string that defines the size, style, and typeface of the equation label. The default value is "8pt Sans-Serif".

`"color": "`*`string`*`"`

Is a color defined by a color name or numeric specification string. The default value is "black".

`"mode": "`*`string`*`"`

Is a string that defines the equation label mode. Valid values are:

❏ "<u>equation</u>", which displays the equation in the most appropriate form for the type of trendline generated. For a linear trendline, the equation is displayed in slope intercept form. This is the default value.

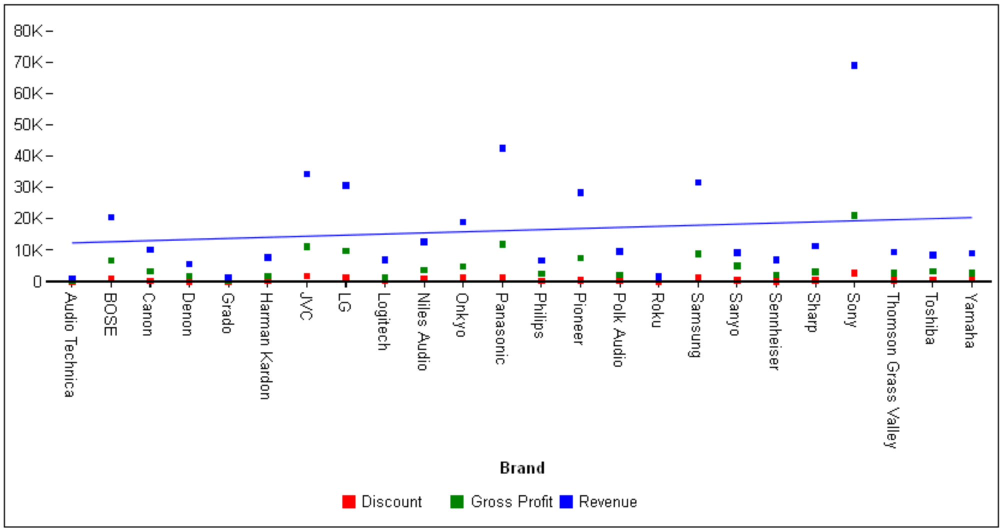
❏ "r", which displays the correlation coefficient.

Example: Drawing a Series Trendline

The following request generates a scatter chart with a linear trendline for series 2:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERS
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "red"},
    {"series": 1, "color": "green"},
    {"series": 2, "color": "blue",
        "trendline": {"enabled": true, "mode": "linear"
        }}]
*END
ENDSTYLE
END
```

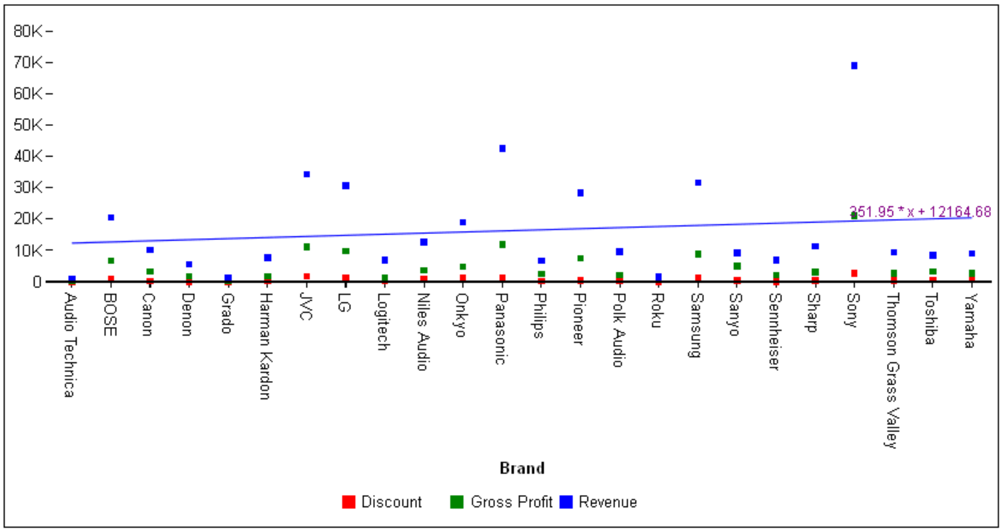
The output is:



The following version of the request adds a purple equation label:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERS
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "red"},
    {"series": 1,"color": "green"},
    {"series": 2, "color": "blue", "trendline": {
        "enabled": true, "mode": "linear", "equationLabel": {
            "visible": true, "color": "purple"}}}]
*END
ENDSTYLE
END
```

The output is:

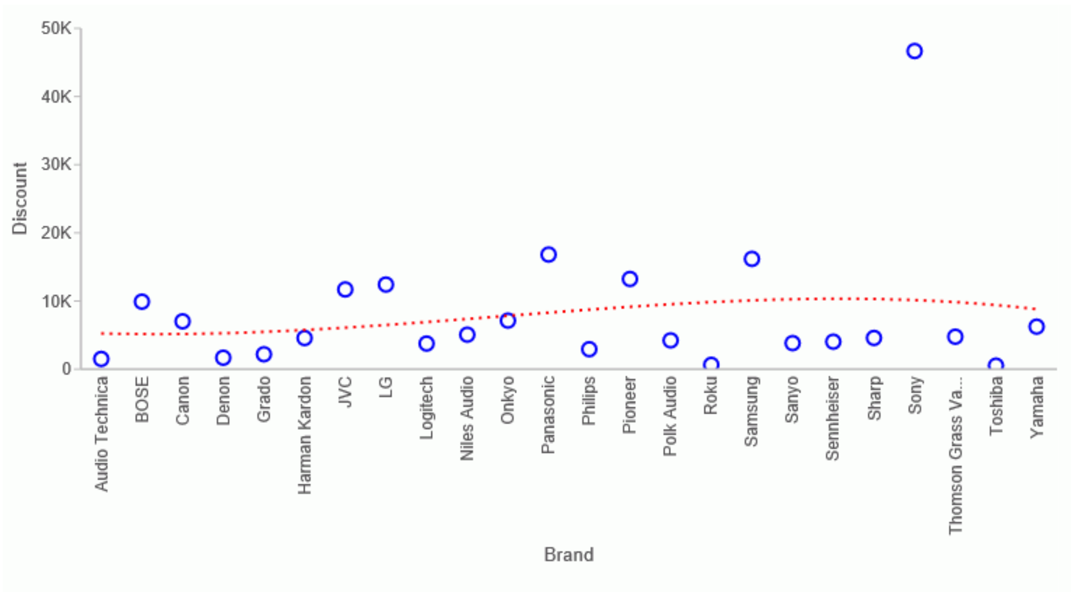


Example: **Styling a Series Trendline**

The following request styles the trendline to be two pixels wide, red, and dashed.

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=BRAND, BUCKET=x-axis,$
TYPE=DATA, COLUMN=DISCOUNT_US , BUCKET=y-axis,$
*GRAPH_JS
"series": [
{"series": 0, "color": "blue", "trendline": {"enabled": true,
   "mode": "polynomial", "order": 3,
   "lineStyle": {"width": 2, "color": "red", "dash": "2 4"},
}}]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Controlling the Visibility of Individual Series

The visible property controls the visibility of individual series.

Syntax: How to Control the Visibility of Individual Series

```
"series":
[
    (
      "series": number,
      "visible": boolean,
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"visible": boolean`

Controls the visibility of the specified series. Valid values are:

❏ true, which makes the series visible. This is the default value.

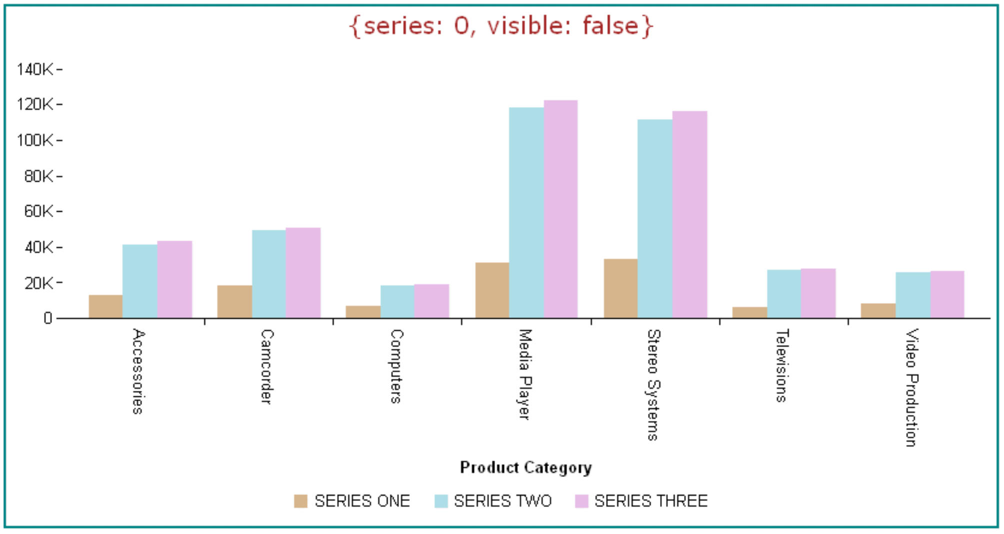
- false, which makes the series not visible.

Example: **Controlling the Visibility of Individual Series**

The following request generates a vertical bar chart and makes series 0 not visible:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"title": {"visible": true, "text": "{series: 0, visible: false}",
    "font": "14pt Verdana", "color": "brown"},
"border": {"width": 2, "color": "teal"},
"series": [
    {"series": 0, "color": "rgb(204,255,255)", "label": "SERIES ZERO",
"visible": false},
    {"series": 1, "color": "tan", "label": "SERIES ONE"},
    {"series": 2, "color": "lightblue", "label": "SERIES TWO"},
    {"series": 3, "color": "rgb(226,185,229)", "label": "SERIES THREE"}]
*END
ENDSTYLE
END
```

On the output, series 0 has no risers and no entry in the legend:



Assigning a Series to an Axis

The yAxisAssignment property assigns a series to an axis. When yAxisAssignment is set to 2, y2-axis labels are added to the chart. For information about properties that control y2axis objects, see *Axis Properties* on page 325.

Syntax: How to Assign a Series to an Axis

```
"series":
[
    {
      "series": number,
      "yaxisAssignment": number,
   }
]
```

where:

`"series": number`

Is a zero-based series number. If the series does not exist in the chart, the property is ignored.

`"yaxisAssignment": number`

Assigns the specified series to the y- or y2-axis. Valid values are:

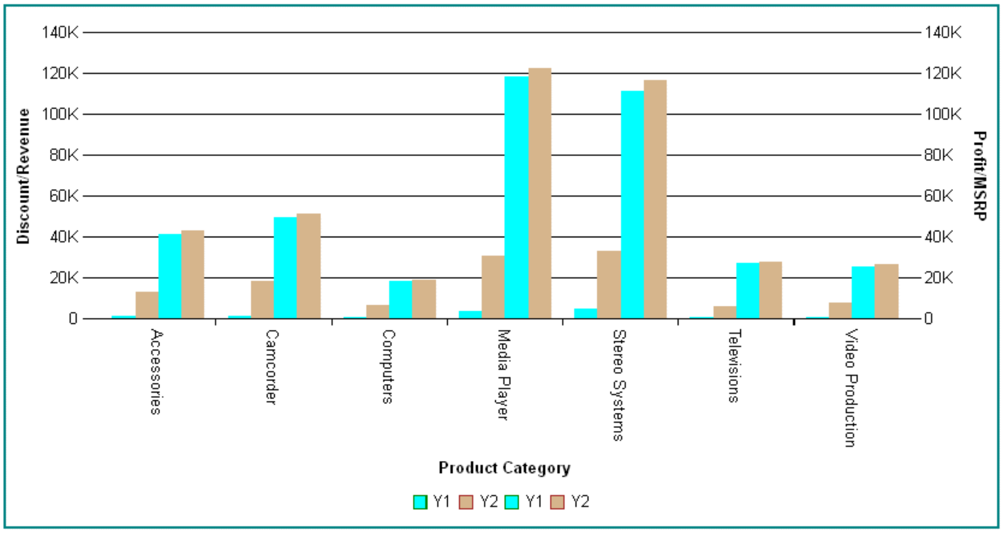
- 1, which assigns the series to the y-axis. This is the default value.
- 2, which assigns the series to the y2-axis.

Example: Assigning a Series to an Axis

The following request generates a vertical bar chart with a y- and y2-axis and assigns series to each axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM DISCOUNT_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"title": {"text": "yAxisAssignment", "font": "14pt Sans-Serif", "color":
"teal"},
"yaxis": {"title": {"visible": true, "text": "Discount/Revenue"}},
"y2axis": {"title": {"visible": true, "text": "Profit/MSRP"}},
"series": [
    {"series": 0, "color": "cyan", "yAxisAssignment": 1, "label": "Y1",
        "marker": {"border": {"width": 1, "color": "green"}}},
    {"series": 1, "color": "tan", "yAxisAssignment": 2, "label": "Y2",
        "marker": {"border": {"width": 1, "color": "brown"}}},
    {"series": 2, "color": "cyan", "yAxisAssignment": 1, "label": "Y1",
        "marker": {"border": {"width": 1, "color": "green"}}},
    {"series": 3, "color": "tan", "yAxisAssignment": 2, "label": "Y2",
        "marker": {"border": {"width": 1, "color": "brown"}}}]
*END
ENDSTYLE
END
```

The output is:



Chapter 10

Chart-Specific Properties

This chapter describes the properties that are specific to each type of chart. Note that the value *undefined* for any property means that the property will be ignored.

In this chapter:

Bar, Line, and Area Chart Properties (blaProperties)

The following properties show the default values for basic attributes in Bar, Line, and Area (BLA) charts. Note that the value *undefined* means that the property will be ignored.

```
"blaProperties": {
"seriesLayout": "sideBySide",
"orientation": "horizontal",
"lineConnection": "linear",
"lineFillEffect": "undefined",
"barGroupGapWidth": 0.2,
"stackTotalLabel": {
    "visible": false,
    "font": "10pt Sans-Serif",
    "color": "black",
    "numberFormat": "auto"
},
"missingDataExtrapolate": true,
"splitY": true,
"comboCharts": {
    "barSeriesLayout": undefined,
    "lineSeriesLayout": undefined,
    "areaSeriesLayout": undefined
    }
}
```

In addition, you can use the axis groupFit properties to control riser width and the axis sort properties to sort stacked bar chart risers in ascending or descending order of the numeric axis values. For information, see *Axis Properties* on page 325.

Controlling the Series Layout in Bar, Line, or Area Charts

The seriesLayout property controls the arrangement of risers in a Bar, Line, or Area chart.

Note: Instead of including the seriesLayout property in the JSON block of the style section, you can use the WebFOCUS LOOKGRAPH parameter, which has a value for each layout. For more information, see *Controlling Chart Type Using LOOKGRAPH* on page 40.

Syntax: How to Control the Series Layout in Bar, Line, or Area Charts

```
"blaProperties": {
   "seriesLayout": "string"
              }
```

where;

`"seriesLayout": "string"`

Defines the arrangement of risers. Valid values are:

- "stacked" to stack the risers on top of each other, with the length of each riser representing its data value.

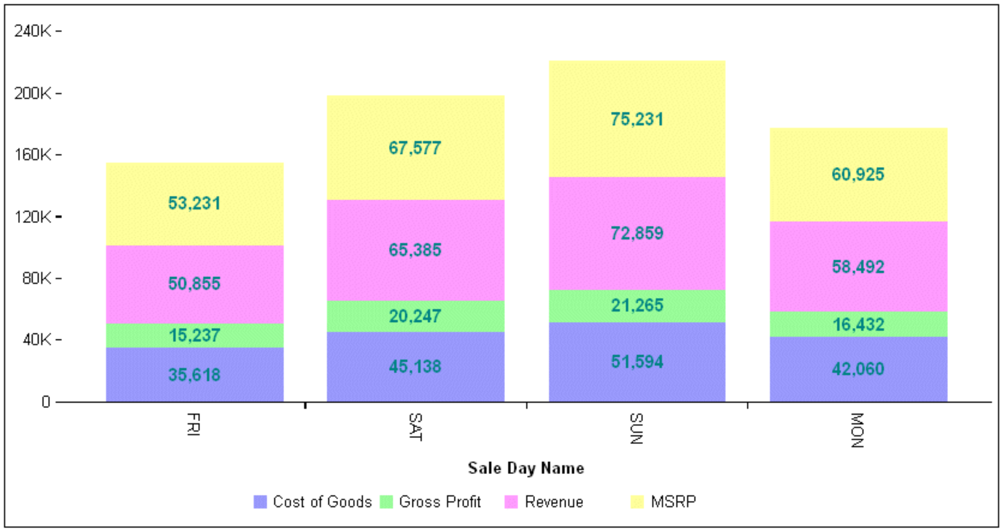
- "absolute" draws each riser at the same x-axis value (for a vertical chart) or y-axis value (for a horizontal chart). If the risers are different widths, they can be distinguished on the chart.
- "sideBySide" (bar charts only) to place the risers next to each other (for vertical bar charts) or one on top of another (for horizontal bar charts). The default value is "sideBySide".
- "percent" to stack the risers, with the length of each riser representing the percentage of the total for that riser, where the total of the stack adds up to 100%.

Example: Setting the seriesLayout Property

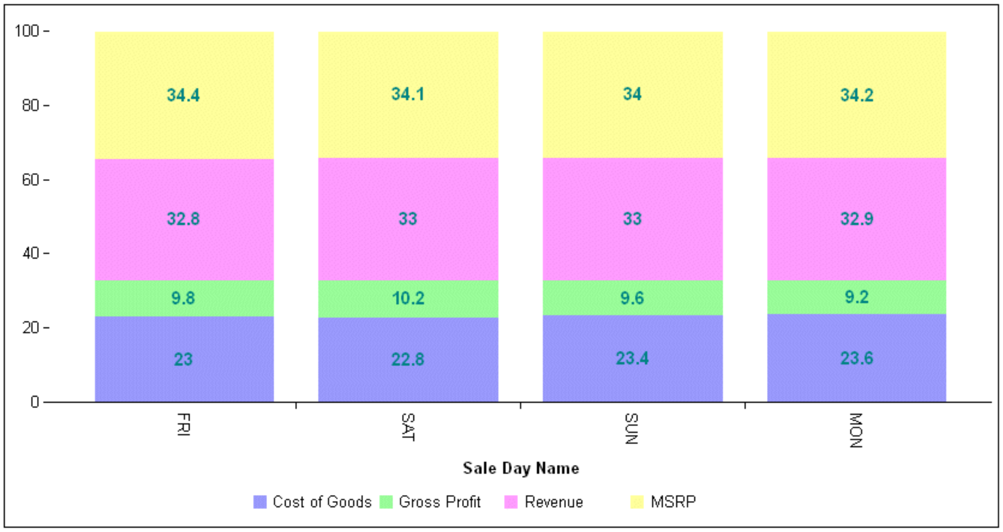
The following request sets the seriesLayout property to "stacked".

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US MSRP_US
ACROSS TIME_DAYNAME COLUMNS 'FRI' AND 'SAT' AND 'SUN' AND 'MON'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"seriesLayout": "stacked"},
"series": [{"series": "all", "showDataValues": true}],
"dataLabels": {"visible": true, "useNegativeColor": true,
    "font": "bold 10pt Sans-Serif", "color": "teal"}
*END
ENDSTYLE
END
```

On the output, the risers are stacked on top of each other, and the value for each one is its data value:



Changing the seriesLayout property to "percent" generates a stacked chart in which the length of each riser represents its percent, and the total for each group is 100:



Specifying the Chart Orientation for Bar, Line, or Area Charts

The orientation property selects the orientation of a Bar, Line, or Area chart.

Note: Instead of including the orientation property in the JSON block of the style section, you can use the WebFOCUS LOOKGRAPH parameter, which has a value for each orientation. For more information, see *Controlling Chart Type Using LOOKGRAPH* on page 40.

Syntax: How to Specify the Chart Orientation

```
"blaProperties": {
   "orientation": "string"
   }
```

where:

`"orientation": "string"`

Valid values are:

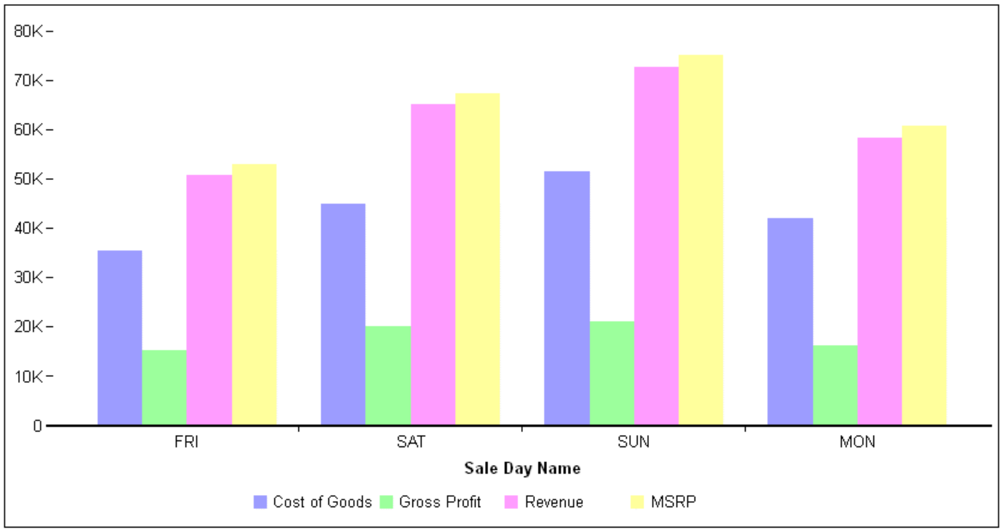
- "horizontal", which draws a horizontal chart (x-axis on left, y-axis on bottom). This is the default value.
- "vertical", which draws a vertical chart (x-axis on bottom, y-axis on left).

Example: Controlling the Chart Orientation in a Bar Chart

By default, the charting engine creates a horizontal bar chart. The following request changes the orientation to vertical, even though the LOOKGRAPH parameter specifies a horizontal bar chart (ON GRAPH SET LOOKGRAPH HBAR):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US MSRP_US
ACROSS TIME_DAYNAME COLUMNS 'FRI' AND 'SAT' AND 'SUN' AND 'MON'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH HBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"orientation": "vertical"}
*END
ENDSTYLE
END
```

The output is:



Controlling How Data Points Are Connected in Bar, Line, or Area Charts

The lineConnection property controls how data points are connected in line and area charts and in charts that use a combination of area and/or line risers using the *Specifying Properties for Combination Charts* property and the series-specific *riserShape* property.

Syntax: How to Set the lineConnection Property in Bar, Line, or Area Charts

```
"blaProperties": {
   "lineConnection": "string"
   }
```

where:

`"lineConnection": "string"`

Defines the line connection. Valid values are:

- "linear", which connects the markers with straight line segments. This is the default value.
- "curved", which connects the markers with curved line segments.
- "stepBefore", which connects the markers with straight line segments that are only horizontal and vertical. The line segments needed to move to the next marker are placed so that the marker is at the end point of the segment at the correct y (for vertical charts) or x (for horizontal charts) coordinate.

- "stepBetween", which connects the markers with straight line segments that are only horizontal and vertical. The line segments needed to move to the next marker are placed so that the marker is at the middle point of the segment at the correct y (for vertical charts) or x (for horizontal charts) coordinate.

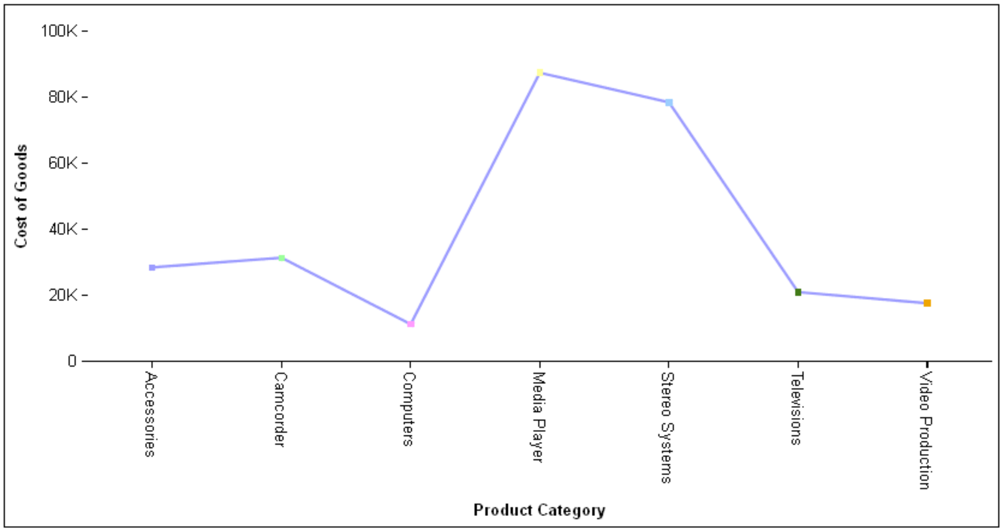
- "stepAfter", which connects the markers with straight line segments that are only horizontal and vertical. The line segments needed to move to the next marker are placed so that the marker is at the beginning point of the segment at the correct y (for vertical charts) or x (for horizontal charts) coordinate.

Example: Setting the lineConnection Property in a Line Chart

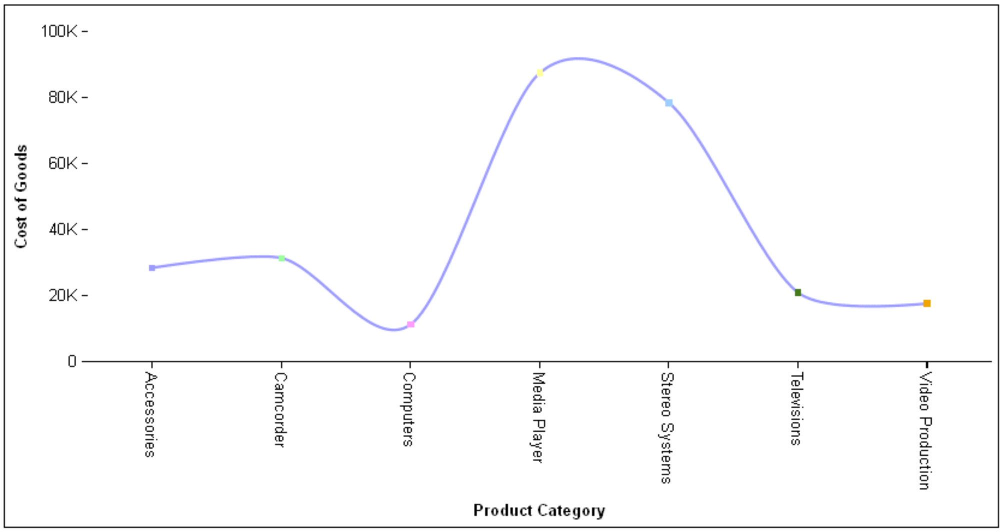
The following request creates a vertical line chart and sets the lineConnection property to the default value, linear:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {
   "lineConnection": "linear"
}
*END
ENDSTYLE
END
```

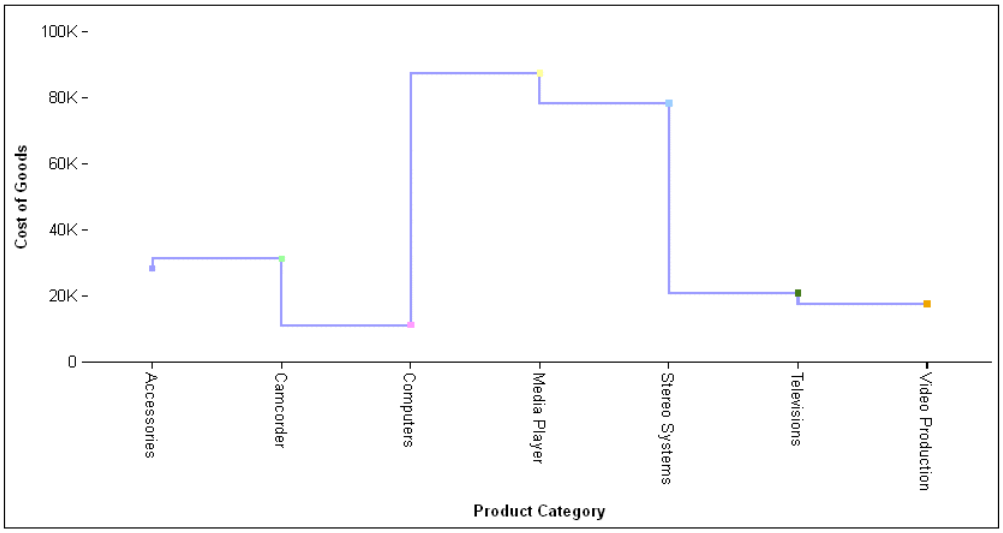
The output is:



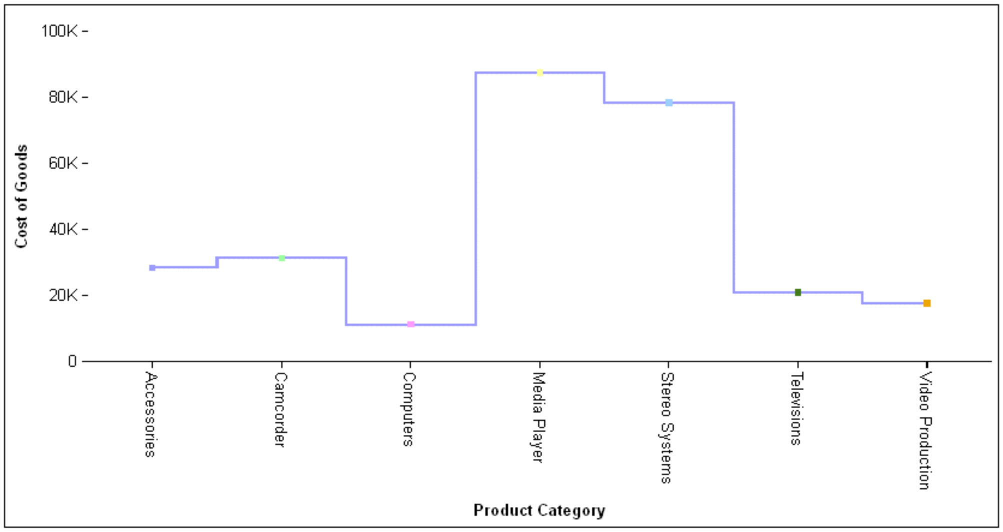
Changing the lineConnection to "curved" produces the following chart:



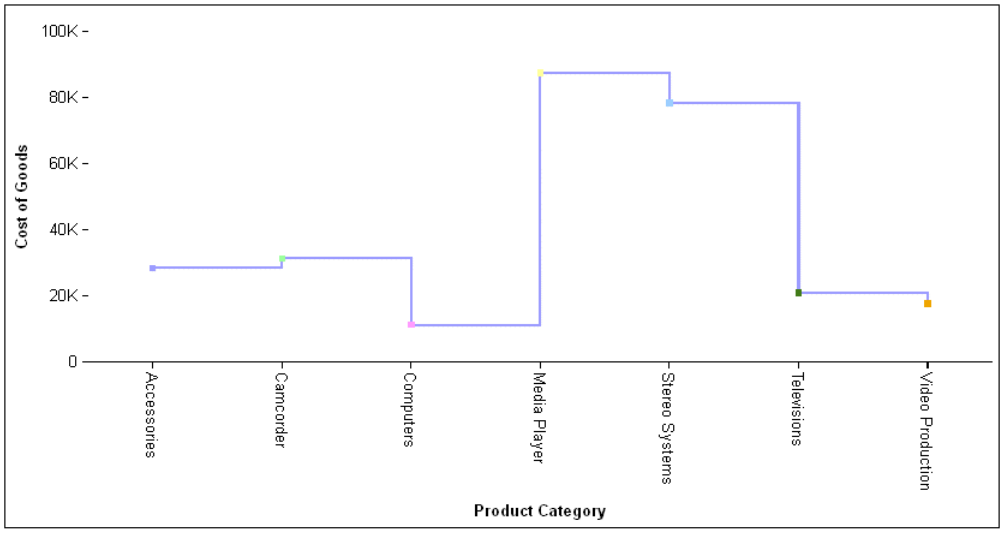
Changing the lineConnection to "stepBefore" produces the following chart, in which all of the line segments are horizontal or vertical, and the markers are at the end of the appropriate line segments:



Changing the lineConnection to "stepBetween" produces the following chart, in which all of the line segments are horizontal or vertical, and the markers are in the center of the appropriate line segments:



Changing the lineConnection to "stepAfter" produces the following chart, in which all of the line segments are horizontal or vertical, and the markers are at the beginning of the appropriate line segments:



Controlling the Line Fill Effect in Bar, Line, or Area Charts

The lineFillEffect property fills the space between line risers and the baseline, or between sibling risers in a stacked or percent line, with the specified color.

Syntax: How to Set the lineFillEffect Property in Bar, Line, or Area Charts

```
"blaProperties": {
   "lineFillEffect": "string"
   }
```

where:

`lineFillEffect: "string"`

Specifies the fill effect color. Valid values are:

- undefined (disabled), which means that no fill effect is used. This is the default value.
- "seriesAuto", which uses a lighter version of the series color for the fill effect.
- "seriesLighten", which uses a lighter version of the series color for the fill effect.
- "seriesLightenOpaque".

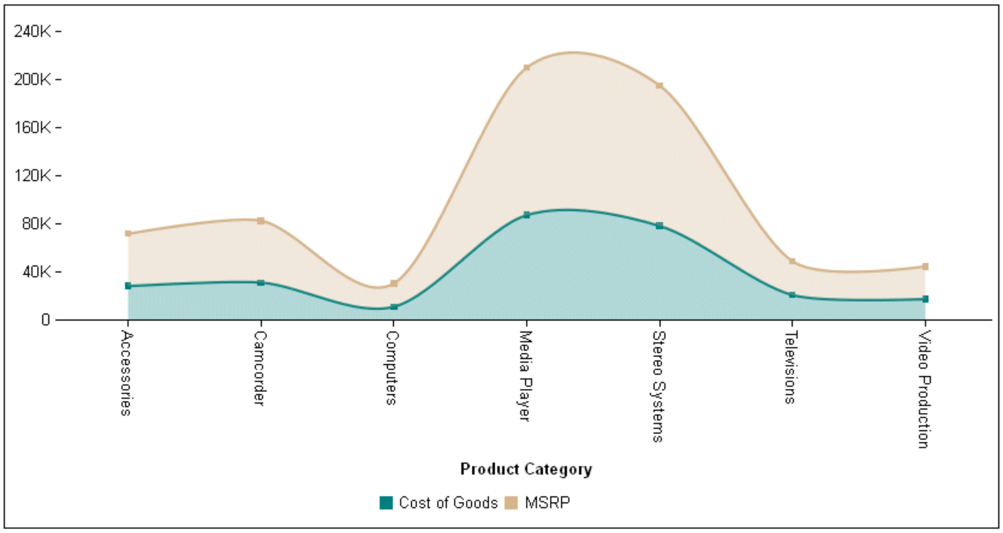
- ❑ A color name or numeric specification string.
- ❑ A gradient defined by a string.
- ❑ a percent string ("-100%"..."100%"). A percent string uses the series color at the specified percentage of opacity.

Example: Setting the lineFillEffect Property in a Line Chart

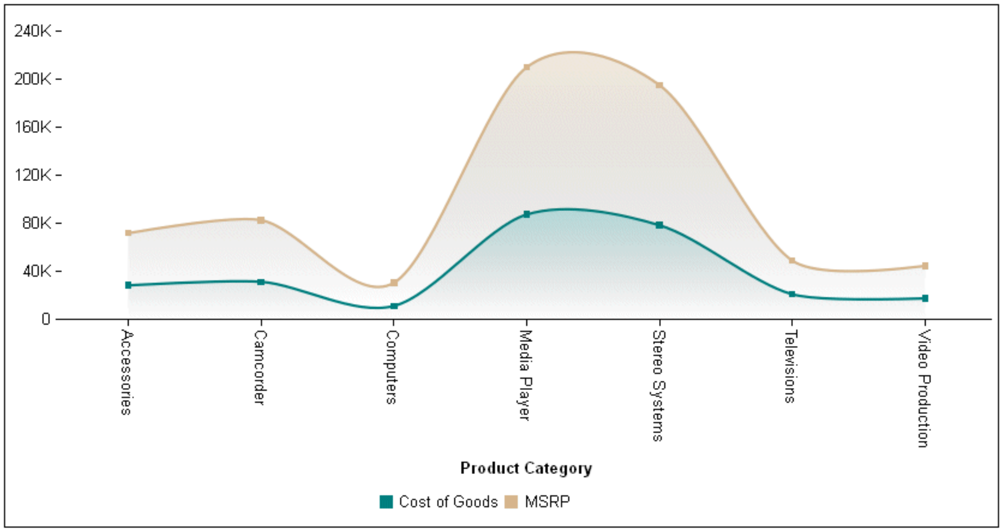
The following request creates a vertical line chart with two series. The line for series 0 is teal, and the line for series1 is tan. The fill effect is seriesAuto:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {
    "seriesLayout": "stacked", "lineConnection": "curved",
    "lineFillEffect": "seriesAuto"},
"series": [
    {"series": 0, "color": "teal"},
    {"series": 1, "color": "tan"}],
"legend": {"visible": true}
*END
ENDSTYLE
END
```

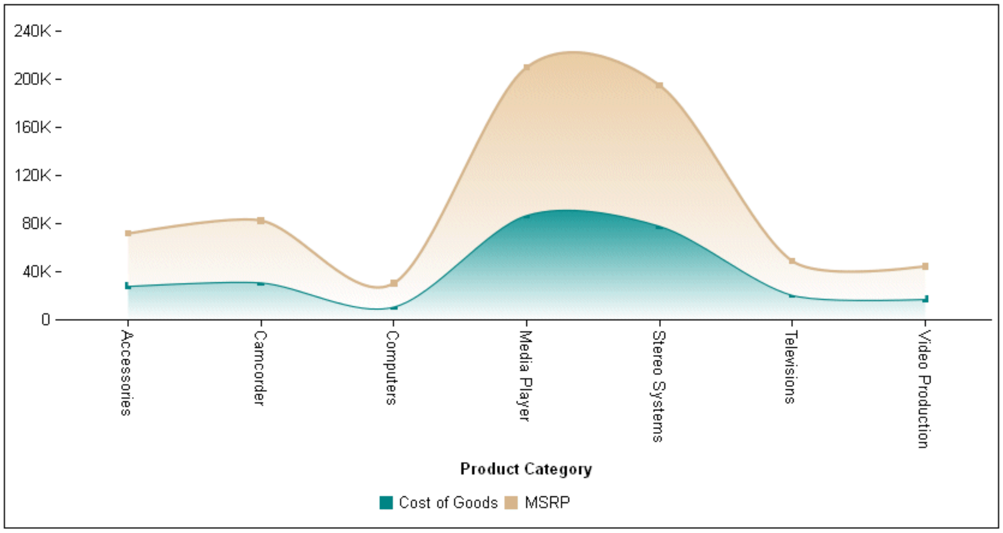
The fill effect is a less opaque version of the series colors, automatically generated by the chart engine:



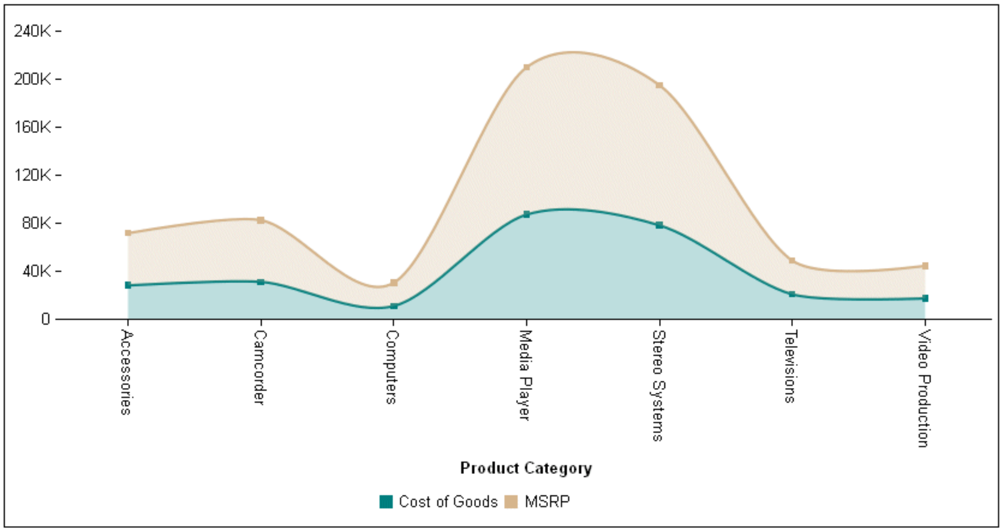
Changing the lineFillEffect to "seriesLighten" produces the following chart, in which the fill effect is lighter than the seriesAuto fill effect:



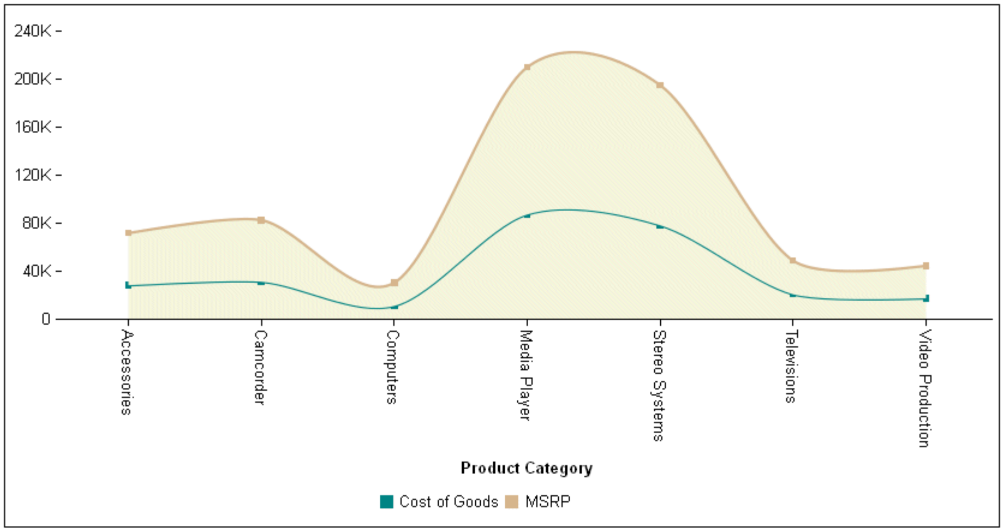
Changing the lineFillEffect to "seriesLightenOpaque" produces the following chart, in which the fill effect is lighter than the seriesAuto fill effect and less transparent than the seriesLighten fill effect:



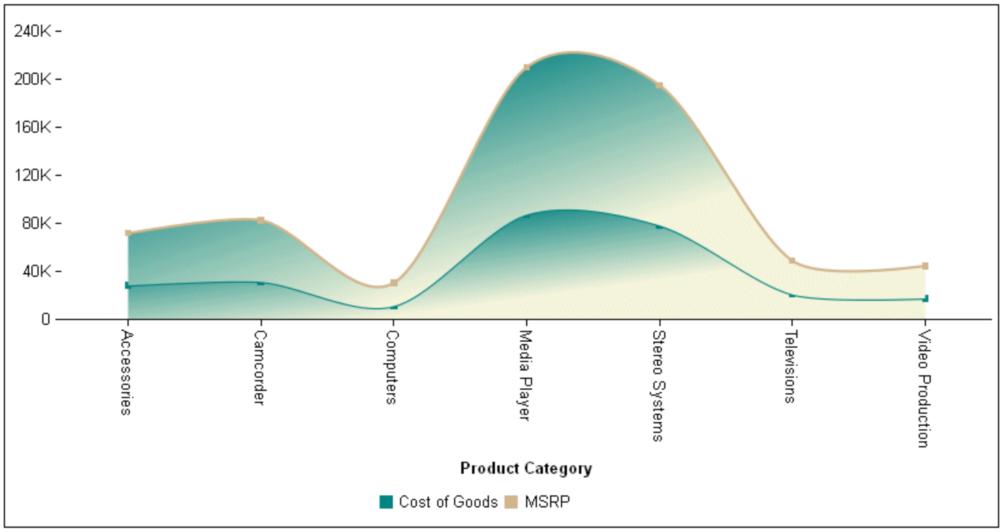
Changing the lineFillEffect to "25%" produces the following chart, in which the fill effect is the series color at 25% opacity:



Changing the lineFillEffect to "beige" produces the following chart, in which the fill effect is the color beige:



Changing the lineFillEffect to "linear-gradient(0,0,100%,100%, 20% teal, 65% beige)" produces the following chart, in which the fill effect is a linear gradient that transitions from teal to beige:



Controlling the Gap Between Groups of Risers in Bar, Line, or Area Charts

The barGroupGapWidth property controls the width of the gap between groups of risers.

Syntax: How to Control the Gap Between Groups of Risers in a Bar, Line, or Area Chart

```
"blaProperties": {
   "barGroupGapWidth": number   }
```

where:

`"barGroupGapWidth":` *`number`*

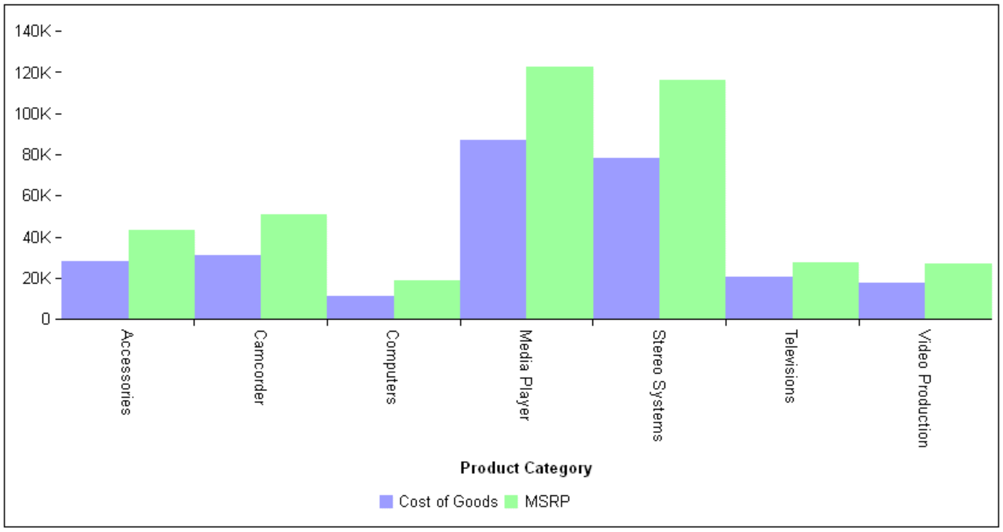
Is a number between 0.0 and 1.0 that defines the width of the gap between groups of risers, as a percent of the available space between the groups. The default value is 0.2.

Example: Specifying the Width Between Risers in a Bar Chart

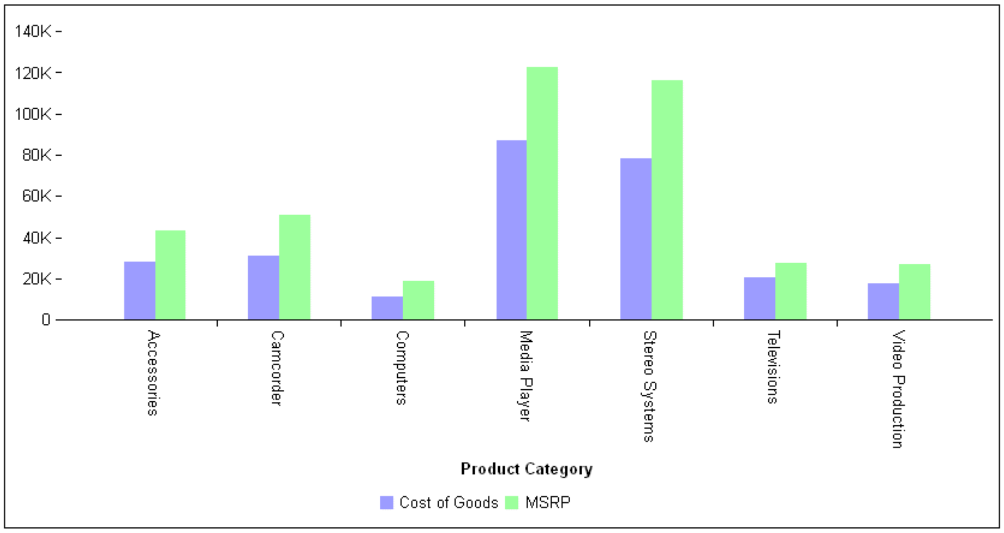
The following request sets the gap between risers to zero (0.0):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"barGroupGapWidth": 0.0}
*END
ENDSTYLE
END
```

On the resulting chart, there is no space between the groups of risers (risers for each product category):



Changing the barGroupGapWidth to 0.5 generates the following chart:



Controlling the Display of Multiple Y-Axes in Bar, Line, or Area Charts

The splitY property controls whether multiple y-axes are displayed as dual axes (one on the left and one on the right) or split axes (stacked on top of each other).

Syntax: How to Control the Display of Multiple Y-Axes in a Bar, Line, or Area Chart

```
"blaProperties": {"splitY": boolean}
```

where:

`"splitY": `*`boolean`*

Determines whether dual or split y-axes are drawn. Valid values are:

- true, which draws split y-axes. This is the default value.
- false, which draws dual y-axes.

Note: To generate split y-axes on which the measures are all rendered with the same color, use the following StyleSheet declaration:

```
type=report, chart-color-measures=off,$
```

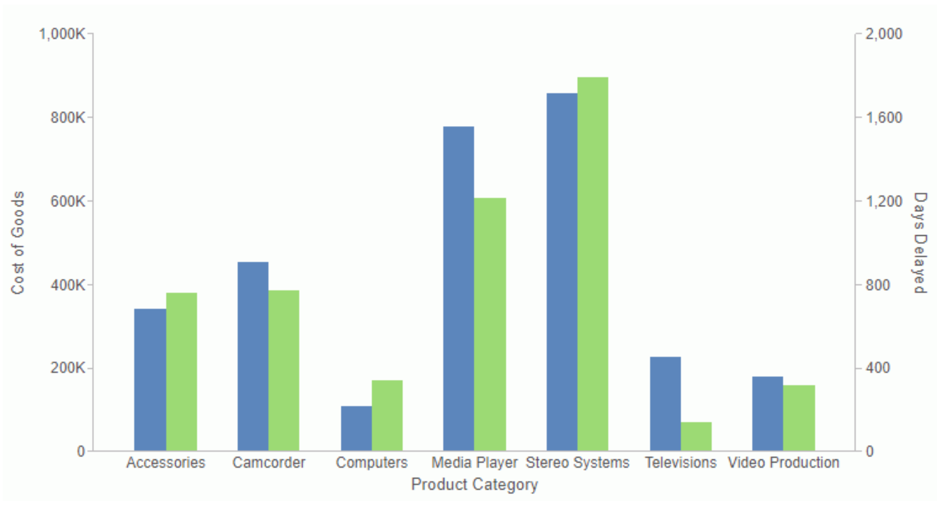
The default value for chart-color-measures is on, which generates a separate color for each measure.

Example: Controlling the Display of Multiple Y-Axes

The following request assigns one measure to the y1-axis and one measure to the y2-axis and displays them as dual y-axes:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DAYSDELAYED
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
type=data, column=product_category, bucket=x-axis,$
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=daysdelayed, bucket=y-axis(2),$
*GRAPH_JS
"blaProperties": {"splitY": false},
"legend": {"visible": false}
*END
ENDSTYLE
END
```

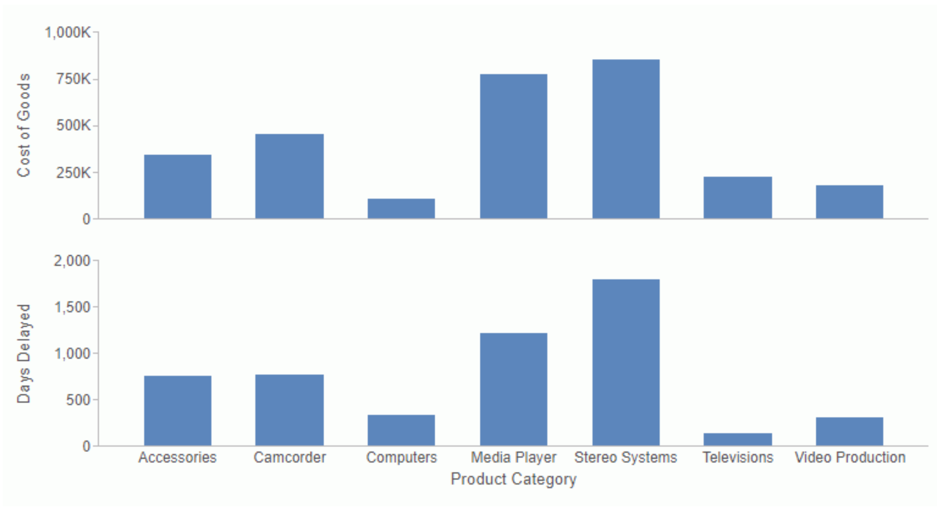
The output is shown in the following image:



The following version of the request produces split y-axes and sets chart-color-measures to off so that both measures are rendered with the same color:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US DAYSDELAYED
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
type=data, column=product_category, bucket=x-axis,$
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=daysdelayed, bucket=y-axis(2),$
type=report, chart-color-measures=off,$
*GRAPH_JS
"blaProperties": {"splitY": true},
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Controlling the Total Label in a Stacked Bar, Line, or Area Chart

The stackTotalLabel properties control the appearance of the total label in stacked bar, line, and area charts (that is, where seriesLayout='stacked').

Syntax: How to Generate a Stack Total

```
"blaProperties": {
   "stackTotalLabel": {
      "visible": boolean,
      "font": "string",
      "color": "string",
      "numberFormat": numformat   },
}
```

where:

`visible: boolean`

Defines the visibility of the stack total label. Valid values are:

- true, which makes the total label visible.
- false, which makes the total label not visible. This is the default value.

`font: "string"`

Is a string that defines the size, style, and typeface of the stack total label. The default value is "10pt Sans-Serif".

`color: "`*`string`*`"`

Is a string that defines the color of the stack total label. The default value is "black".

`numberFormat:` *`numformat`*

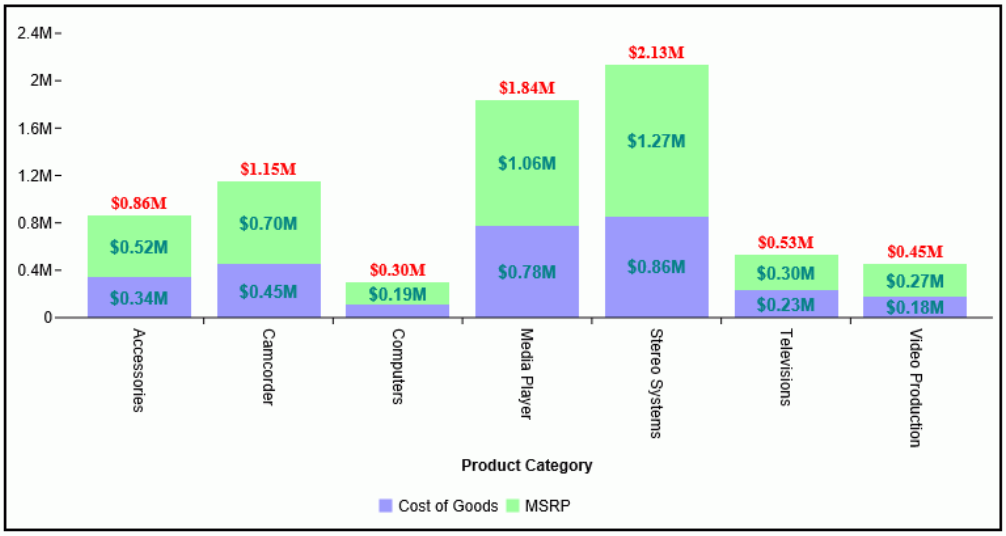
Defines the number format for the label. It can be specified as a JSON object, a format string, or a user-defined function. The default value is "auto". You can also use autoNumberFormats to apply a number format to all stack total labels. For information on number formats, see *Formatting Numbers* on page 108.

Example: Generating a Total Label in a Stacked Bar Chart

The following request creates a stacked vertical bar chart with a stack total in red on top of each stacked bar. The series object enables data labels, and the dataLabels object makes the data text visible:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"blaProperties": {"seriesLayout": "stacked", "stackTotalLabel": {
    "visible": true, "font": "10pt Serif, bold", "color": "red",
    "numberFormat": {"mode": "currency", "grouping": "M"}}},
"series": [{"series": "all", "showDataValues": true}],
"dataLabels": {"visible": true, "font": "bold 10pt Sans-Serif", "color":
"teal",
    "numberFormat": {"mode": "currency", "grouping": "M"}},
"legend": {"visible": true}
*END
ENDSTYLE
END
```

On the output, the data labels are within the bars, and the stack total is on top:



Hiding an Extrapolated Line That Starts Before Actual Values

By default, if missing values are shown on a line chart, and if they start before the start of actual values in the data source, the line is extrapolated to include those missing values. The missingDataExtrapolate:false property omits those extrapolated values from the chart output.

Syntax: How to Control Display of an Extrapolated Line for Missing Values

```
"blaProperties": {
 "missingDataExtrapolate": boolean}
```

where:

`"missingDataExtrapolate": `*`boolean`*

Controls display of an extrapolated line that starts before actual values. Valid values are

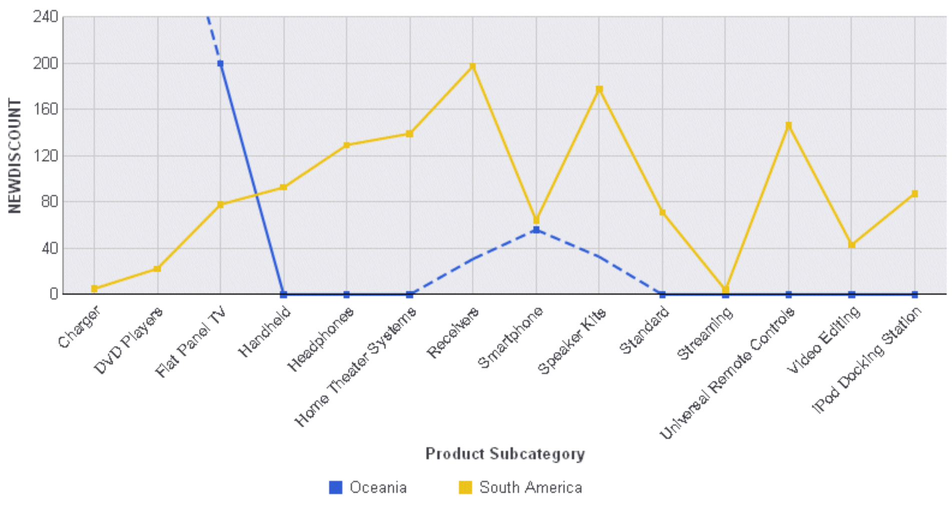
- true, which displays the extrapolated line. This is the default value.
- false, which hides the extrapolated line.

Example: Hiding an Extrapolated Line That Starts Before Actual Values

The following request generates a vertical line chart with missing values that start before the actual values:

```
GRAPH FILE WF_RETAIL_LITE
SUM COMPUTE NEWDISCOUNT/D12.2=IF BUSINESS_REGION EQ 'South America'
   THEN DISCOUNT_US *.1
   ELSE DISCOUNT_US ;
BY BUSINESS_REGION
ACROSS PRODUCT_SUBCATEG
WHERE BUSINESS_REGION EQ 'Oceania' OR 'South America';
WHERE PRODUCT_SUBCATEG NE 'Blu Ray';
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET VZERO OFF
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_SCRIPT
setFillMissingData(2);
*END
ENDSTYLE
END
```

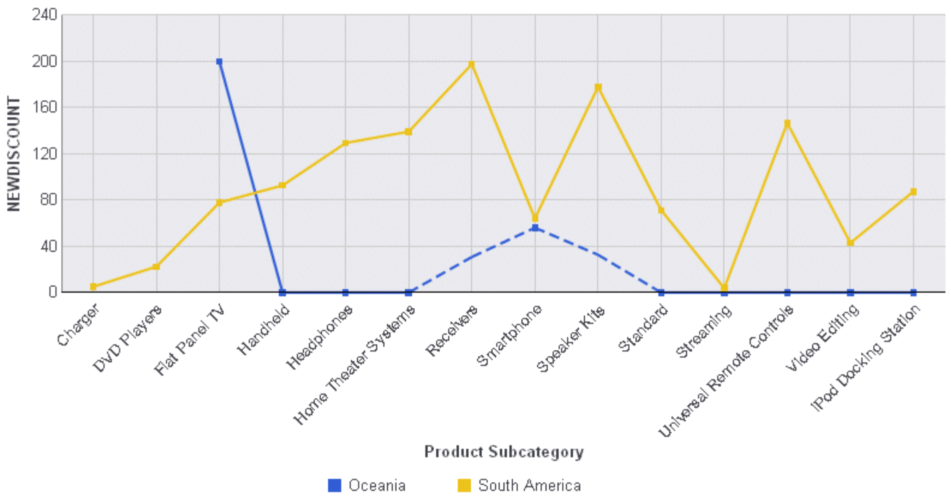
On the chart output, the line is extrapolated to display those missing values, as shown on the following image:



Running the request with the following *GRAPH_JS block removes the extrapolated line from the chart output:

```
*GRAPH_JS
"blaProperties": {"missingDataExtrapolate": false}
*END
```

The chart output is shown on the following image:



Specifying Properties for Combination Charts

The comboCharts properties are used in conjunction with the series#:*riserShape* property to define a combination chart (that is, a chart that can consist of a combination of bar risers, area risers, and line risers). When you generate a bar, line, or area chart, the series-specific *riserShape* property can be used to define the shape of the riser for each series as a bar, line, or area. (For more information, see *Defining the Shapes of Risers for a Series in Bar, Line, and Area Charts* on page 463.) These properties control the layout (stacked, absolute, percent, or sideBySide) of the risers for each series.

Syntax: How to Specify Combinations of Riser Shapes

```
"blaProperties": {
   "comboCharts": {
      "barSeriesLayout": "string",
      "lineSeriesLayout": "string",
      "areaSeriesLayout": "string",
   }
}
```

where:

`"barSeriesLayout": "`*`string`*`"`

Defines the riser layout for series assigned bar risers (series#:markerShape: "bar"). Valid values are "absolute", "percent", "sideBySide", or "stacked". The default value is undefined. For more information about series layout, see *Controlling the Series Layout in Bar, Line, or Area Charts*.

`"lineSeriesLayout": "`*`string`*`"`

Defines the riser layout for series assigned line risers (series#:markerShape: "line"). The default value is undefined. Valid values are "absolute", "percent", or "stacked". For more information about series layouts, see *Controlling the Series Layout in Bar, Line, or Area Charts* on page 492.

`"areaSeriesLayout": "`*`string`*`"`

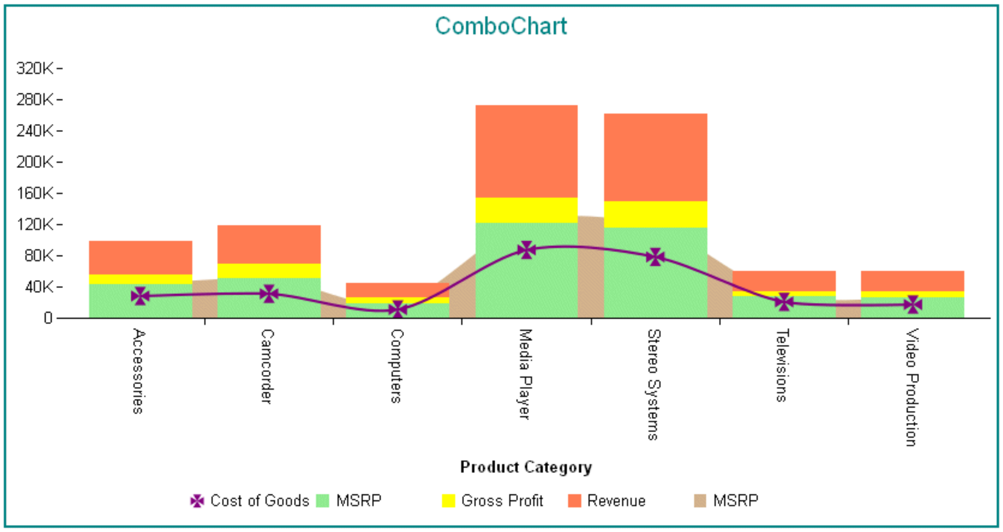
Defines the riser layout for series assigned area risers (series#:markerShape: "area"). The default value is undefined. Valid values are "absolute", "percent", or "stacked". For more information about series layouts, see *Controlling the Series Layout in Bar, Line, or Area Charts* on page 492.

Example: Controlling the Series Layouts in Combo Charts

The following request creates a combo chart with one line riser, whose marker shape is a pirate cross, one area riser, and three stacked bar risers. The series object assigns each series a type of riser:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US MSRP_US GROSS_PROFIT_US REVENUE_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
"border": {"width": 2, "color": "teal"},
"title": {"visible": true, "text": "ComboChart", "font": "14pt Sans-Serif",
"color": "teal"},
"blaProperties": {
    "lineConnection": "curved", "comboCharts": {
        "barSeriesLayout": "stacked",
        "lineSeriesLayout": "absolute",
        "areaSeriesLayout": "undefined"}},
"series": [
    {"series": 0, "color": "purple", "riserShape": "line", "border":
{"width": 2},
        "marker": {"shape": "pirateCross", "size": 14, "visible": true}},
    {"series": 1, "color": "lightgreen", "riserShape": "bar"},
    {"series": 2, "color": "yellow", "riserShape": "bar"},
    {"series": 3, "color": "coral", "riserShape": "bar"},
    {"series": 4, "color": "tan", "riserShape": "area"}]
*END
ENDSTYLE
END
```

The output is:



Note: When a combination chart is defined with both the comboChart properties and the series#:*riserShape* property, area risers are drawn first in the back, then bars, then lines in the front.

Box Plot Properties (boxPlotProperties)

These properties control the general appearance of a box plot.

The following code segment shows the properties with their default values:

```
"boxPlotProperties": {
   "hatWidth": "50%",
   "drawHatAsBox": false,
   "medianLine": {
      "width": 1,
      "color": "black",
      "dash": ""
   },
         "connectorLine": {
                        "width": 1,
                        "color": "black",
                        "dash": ""
   }
},
```

Specifying Box Plot Hat Width

The hatWidth property controls the width of the hat that draws at the top and base of a box plot.

Syntax: How to Set the Width of the Box Plot Hats

```
boxPlotProperties: {
   "hatWidth": width
```

where:

`"hatWidth":`*`width`*

Can be a number that defines the width of the hat in pixel or a string that includes a percent symbol, enclosed in double quotation marks (") that expresses a percentage value ("0%"..."100%"). The default value is "50%".

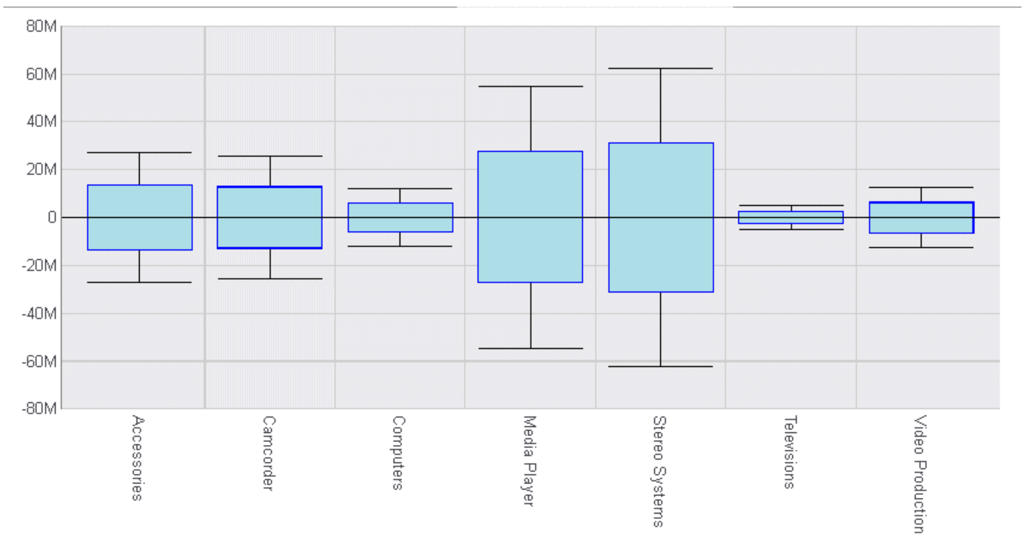
Note: The series and group border properties control the color and format of the hats when they are drawn as boxes. When they are not drawn as boxes, the connectorLine properties control the color and format of the hats. You can set the border width to zero to remove the hats entirely (the same as hatWidth: "0%" or hatWidth: 0).

Example: Specifying the Box Plot Hat Width

The following request sets the hat width to 100% of the box width:

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US  -100000;
DIFF2 = COGS_US  -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US  DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"boxPlotProperties": {"hatWidth": "100%"},
"series": [
    {"series": 0, "color": "lightblue", "border": {
        "width": 1, "color": "blue"}
        }]
*END
ENDSTYLE
END
```

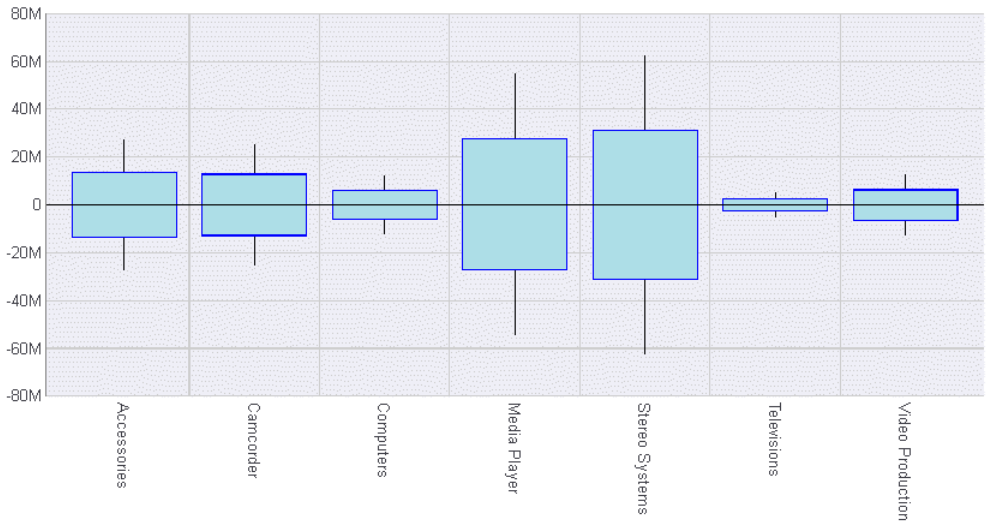
The output is:



Setting the hatwidth to zero (0), removes the hats, but leaves the connector lines:

```
"boxPlotProperties": {
   "hatWidth": 0},
```

The output is:



Controlling the Appearance of Box Plot Hats

The drawHatAsBox property controls the appearance of the box plot hats (upper and lower).

Syntax: How to Control the Appearance of Box Plot Hats

```
"boxPlotProperties": {
   "drawHatAsBox": boolean   },
```

where:

`"drawHatAsBox": boolean`

Valid values are:

- true, which draws box plot hats as boxes.
- false, which draws normal hats. This is the default value.

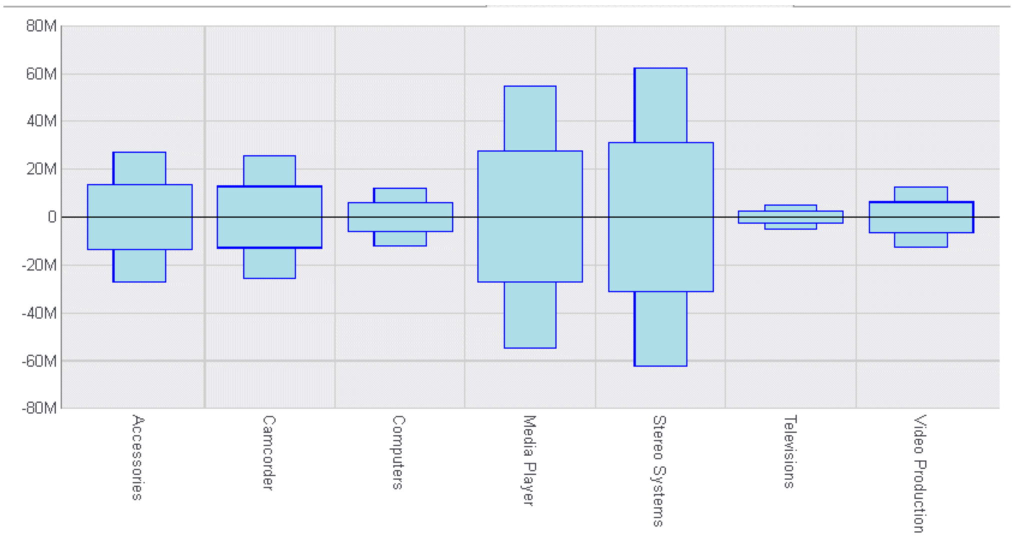
Note: The series and group border properties control the color and format of the hats when they are drawn as boxes. When they are not drawn as boxes, the connectorLine properties control the color and format of the hats.

Example: Drawing the Box Plot Hats as Boxes

The following request draws the hats in the box plot as boxes:

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US  -100000;
DIFF2 = COGS_US  -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US  DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"boxPlotProperties": {"drawHatAsBox": true},
"series": [
    {"series": 0, "color": "lightblue", "border": {
        "width": 1, "color": "blue"}
        }]
*END
ENDSTYLE
END
```

The output is:



Controlling the Appearance of the Box Plot Median Line

The medianLine properties control the appearance of the box plot median line.

Syntax: How to Control the Appearance of the Box Plot Median Line

```
"boxPlotProperties": {
            "medianLine": {
                        "width": number,
      "color": "string",
      "dash": "string"
   }
},
```

where:

`"width": number`

Is the width of the median line in pixels. The default value is 1.

`"color": "string"`

Is the color of the median line defined by a keyword or numeric specification string. The default value is 'black'.

`"dash": "string"`

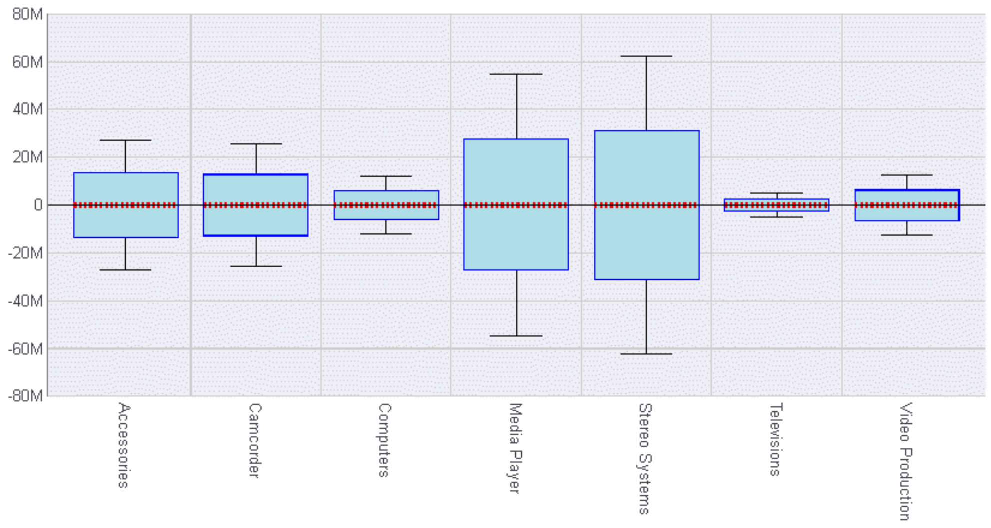
Is a string that defines the dash style of the median line. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Controlling the Box Plot Median Line

The following request generates a red median line with a width of 4 and a dash with a width of 2 and a gap of 2:

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US  -100000;
DIFF2 = COGS_US  -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US  DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"boxPlotProperties": {"medianLine": {
    "width": 4, "color": "red", "dash": "2 2"}},
"series": [
    {"series": 0, "color": "lightblue", "border": {"width": 1, "color": "blue"}
        }]
*END
ENDSTYLE
END
```

The output is:



Controlling the Appearance of the Box Plot Connector Line

The connectorLine properties control the appearance of the box plot connector lines and hats. The connector line is only visible when drawHatAsBox is false.

Syntax: How to Control the Appearance of the Box Plot Connector Line

```
"boxPlotProperties": {
   "connectorLine": {
      "width": number,
      "color": "string",
      "dash": "string"
   }
},
```

where:

`"width": number`

Is the width of the connector line in pixels. The default value is 1.

`"color": "string"`

Is the color of the connector line defined by a keyword or numeric specification string. The default value is "black".

`"dash": "string"`

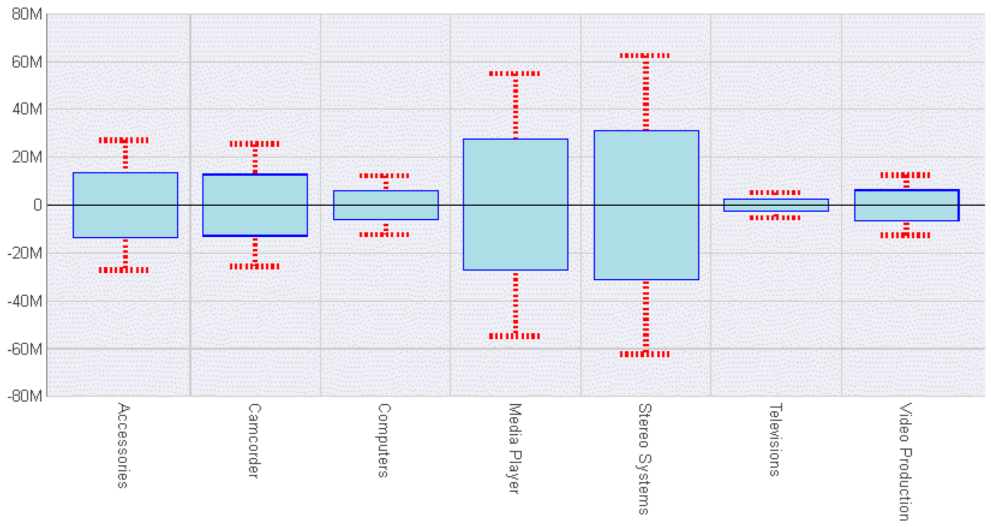
Is a string that defines the dash style of the connector line. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Controlling the Box Plot Connector Line

The following request generates a red connector line with a width of 4 and a dash with a width of 2 and a gap of 2:

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US  -100000;
DIFF2 = COGS_US  -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US  DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"boxPlotProperties": {"connectorLine": {
    "width": 4, "color": "red", "dash": "2 2"}},
"series": [
    {"series": 0, "color": "lightblue", "border": {"width": 1, "color": "blue"}
        }]
*END
ENDSTYLE
END
```

The output is:



Bullet Chart Properties (bulletProperties)

The drawFirstValueAsBar property controls the general appearance of a bullet chart.

Syntax: How to Control the Appearance of a Bullet Chart

```
"bulletProperties": {
   "drawFirstValueAsBar": boolean},
```

where:

`"drawFirstValueAsBar": boolean`

Valid values are:

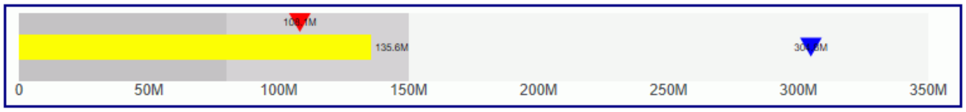
- true, which draws the first data value as a bar. This is the default value.
- false, which draws the first data value as a marker.

Example: Controlling the Appearance of a Bullet Chart

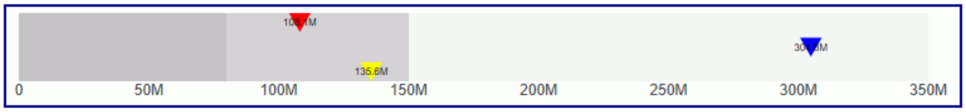
The following request creates a bullet chart using the default value (true) for the drawFirstValueAsBar property

```
GRAPH FILE WF_RETAIL_LITE
SUM MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories' OR 'Computers' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET VAXIS 80
ON GRAPH SET LOOKGRAPH CUSTOM
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"chartType": "bullet",
"border": {"width": 1, "color": "navy"},
"bulletProperties": {"drawFirstValueAsBar": true},
"dataLabels": {"visible": true, "font": "6pt"},
"yaxis": {"colorBands": [
    {"start": 0, "stop": 80000000, "color": "silver"},
    {"start": 80000000, "stop": 150000000, "color": "lightgrey"},
    {"start": 150000000, "stop": 200000000, "color": "whitesmoke"}]},
"series": [
    {"series": "all", "showDataValues": true},
    {"series": 0, "group": 0, "color": "yellow",
        "marker": {"size": 15, "position": "bottom", "shape": "triangle"}},
    {"series": 0, "group": 1, "color": "red",
        "marker": {"size": 15, "position": "top", "shape": "triangle"}},
    {"series": 0, "group": 2, "color": "blue",
        "marker": {"size": 15, "position": "middle", "shape": "triangle"}}]
*END
ENDSTYLE
END
```

On the chart, the first data value is represented by a yellow bar, and the other data values are represented by red and blue triangles:



Changing the value of drawFirstValueAsBar to false generates a chart on which the yellow marker is now a triangle:



Data Grid Properties (dataGridProperties)

A data grid is a tabular representation of data, similar to a tabular report. The grid can have multiple rows, columns, and measures. Unlike the tabular reports generated with the WebFOCUS TABLE FILE command, data grids are generated in a GRAPH FILE request using PCHOLD FORMAT JSCHART.

Data grids are only available using chart attribute syntax. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

Note: The HTML5 data grid visualization does not support changing the theme (WebFOCUS StyleSheet), as the grid visualization does not currently support the WebFOCUS TABLE formatting syntax.

The following code segment shows the supported properties and their default values:

```
"dataGridProperties": {
    "altRowFill": "undefined",
    "cell": {
        "layout": {
             "width": {
                   "rule": "maxSize",
                   "value": 300},
    "cellPadding": {
      "left": 5,
      "top": 5,
      "right": 5,
      "bottom": 5},
             "height": {
                  "rule": "maxSize",
                  "value": 500}
            },
        "innerGridLines": {
            "horizontal": {
                "width": 1,
                "color": "#E0E0E0",
                "dash": ""},
            "vertical": {
                "width": 1,
                "color": "#EEEEEE",
                "dash": ""}},
        "labels": {
            "font": "10pt Sans-Serif",
            "color": "rgb(20, 20, 20)",
            "numberFormat": "auto",
            "align": "right",
            "valign": "top"},
        "tooltip": "undefined",
        "formatList": []
        },
    "rowHeader": {
        "fill": "#FAFAFA",
        "border": {
            "width": 1,
            "color": "#D0D0D0",
            "dash": ""
            },
        "innerGridLines": {
            "horizontal": {
                "width": 1,
                "color": "#D8D8D8",
                "dash": ""},
            "vertical": {
                "width": 1,
                "color": "#E0E0E0",
                "dash": ""}
            },
        "dividerLine": {
            "width": 1,
            "color": "#D0D0D0",
            "dash": ""
            },
        "title": {
            "text": [],
            "font": "bold 10pt Sans-Serif",
            "color": "rgb(20, 20, 20)",
            "align": "left",
            "valign": "top",
            "tooltip": "undefined",
            "dividerLine": {
                "width": 1,
                "color": "#D0D0D0",
                "dash": ""}
            },
        "labels": {
            "font": "10pt Sans-Serif",
            "color": "rgb(20, 20, 20)",
            "align": "left",
            "valign": "top",
            "mergeMatching": true,
            "content": []}
            },
         "colHeader": {
            "sorting": {
                "enabled": false},
            "resize": {
                "enabled": false},
            "fill": "#FAFAFA",
            "border": {
                "width": 1,
                "color": "#D0D0D0",
                "dash": ""},
            "innerGridLines": {
                "horizontal": {
                    "width": 1,
                    "color": "#E0E0E0",
                    "dash": ""},
                "vertical": {
                    "width": 1,
                    "color": "#ECECEC",
                    "dash": ""}
                },
            "dividerLine": {
                "width": 1,
                "color": "#D0D0D0",
                "dash": ""},
            "title": {
                "text": [],
                "font": "bold 10pt Sans-Serif",
                "color": "rgb(20, 20, 20)",
                "align": "right",
                "valign": "top",
                "tooltip": "undefined",
                    "dividerLine": {
                        "width": 1,
                        "color": "#EEEEEE",
                        "dash": ""}
                },
            "labels": {
                "font": "10pt Sans-Serif",
                "color": "rgb(20, 20, 20)",
                "align": "center",
                "valign": "top",
                "mergeMatching": true,
                "content": []},
            "lastLabels": {
                "font": "bold 10pt Sans-Serif",
                "color": "rgb(20, 20, 20)",
                "align": "right",
                "valign": "center"}
            },
        "columnTotals": {
            "visible": false,
            "calculation": "sum",
            "headerLabel": {
                "text": "TOTAL",
                "font": "bold 10pt Sans-Serif",
                "color": "rgb(20, 20, 20)",
                "align": "left",
                "valign": "center"},
            "cellLabel": {
                "font": "bold 10pt Sans-Serif",
                "color": "rgb(20, 20, 20)",
                "align": "right",
                "valign": "top"},
            "border": {
                "width": 1,
                "color": "#D0D0D0",
                "dash": ""},
            "dividerLine": {
                "width": 2,
                "color": "#D0D0D0",
                "dash": ""}
            },
        "scroll": {
            "enabled": true,
            "freezeRowHeader": true,
            "freezeColHeader": true,
            "size": 15,
            "color": "rgb(240, 240, 240)",
            "handle": {
                "color": "#C2C2C2",
                "hoverColor": "#A8A8A8",
                "border": {
                    "width": 0,
                    "color": "transparent",
                    "dash": ""}
                }
            }
        }
    }
```

Defining a Background Color for Alternate Rows in a Data Grid

Using the dataGridProperties:altRowFill property, you can define background colors that alternate on successive rows of the grid. If you define one color, rows will alternate between that color and the default fill.

Syntax: How to Define a Background Color for Alternate Rows in a Data Grid

```
dataGridProperties: {
 "altRowFill": "string"
                           }
```

where:

`"altRowFill": "string"`

Defines a background to fill alternating rows of the data grid.

Can be one of the following.

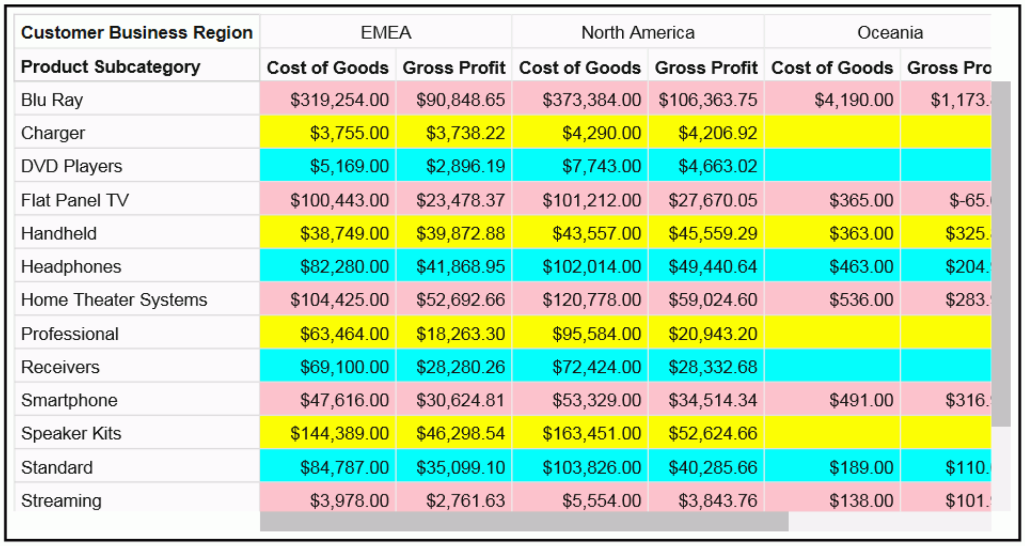
- undefined. This is the default value.
- A string that defines a single color, for example, "red".
- A string that defines an array of colors, for example, ["red","green"].

Example: Defining Alternating Background Colors for Rows in a Data Grid

The following request defines an array of colors that alternate as background colors for the rows in the data grid.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {
    "altRowFill": ["pink", "yellow", "cyan"]}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania	
Product Subcategory	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Pro**
Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.00	$1,173.
Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92		
DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02		
Flat Panel TV	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.
Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.00	$325.
Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.00	$204.
Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.00	$283.
Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20		
Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68		
Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.
Speaker Kits	$144,389.00	$46,298.54	$163,451.00	$52,624.66		
Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.00	$110.
Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.00	$101.

Formatting Cells in a Data Grid

You can format the layout of the cells, the grid lines, and the labels.

Syntax: How to Format the Cell Layout in a Data Grid

The dataGridProperties:cell:layout properties format the width and height of the data cells in the grid.

```
"dataGridProperties": {
    "cell": {
        "layout":
    "width": width,
    "cellPadding": {
       "left": number,
       "top": number,
       "right": number,
       "bottom": number}
    "height": height
      }
          }
 }
```

where:

`"width": width`

Defines the width of the cells in the grid. If undefined, the grid will be sized to completely fill the available space. Can be:

- A number of pixels.
- A rule and value pair in the following form.

  `{"rule": "string", "value": number}`

  where:

`rule: "string"`

Is one of the following.

- **"exactSize"**, which makes each cell exactly the size specified by *n*
- **"minSize"**, which makes each cell at least the size specified by *n*
- **"maxSize"**, which makes each cell at most the size specified by *n*

The default value is {"rule": "maxSize", "value": 300}

`"value": number`

Is a number of pixels.

`"cellPadding"`

Defines the margins inside the cell.

`"top":` *`number`*

Is the top margin of the cell in pixels. The default value is 5.

`"left":` *`number`*

Is the left margin of the cell in pixels. The default value is 5.

`"height":` *`height`*

Defines the height of the cells in the grid. If undefined, the grid will be sized to completely fill the available space. Can be:

- A number of pixels.
- A rule and value pair in the following form.

  `{"rule": "`*`string`*`", "value":` *`number`*`}`

  where:

`rule: "`*`string`*`"`

Is one of the following.

- **"exactSize"**, which makes each cell exactly the size specified by *n*
- **"minSize"**, which makes each cell at least the size specified by *n*
- **"maxSize"**, which makes each cell at most the size specified by *n*

The default value is {rule: "maxSize", value: 500}

`value:` *`number`*

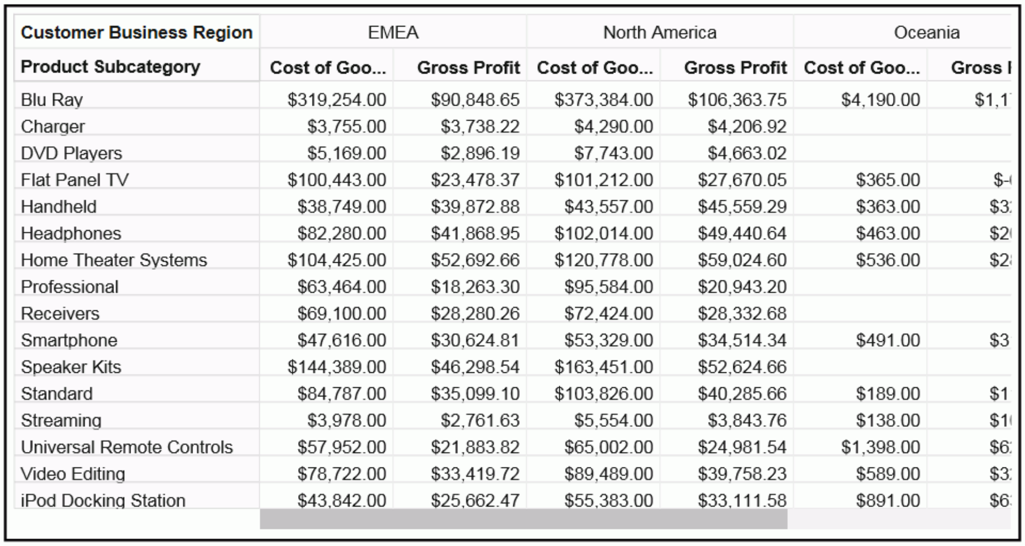
Is a number of pixels.

Example: **Formatting the Cell Layout in a Data Grid**

The following request sets the cell width to 100 pixels and the cell height to 20 pixels.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
dataGridProperties: {
cell:{
  layout:{
     width: 100,
     height: 20
            }
    }
                               }
*END
ENDSTYLE
END
```

The output is shown in the following image. Some of the column headers do not fit into the 100-pixel width, and are truncated. This could be solved by increasing the width or setting the minSize to 100 (width: {"rule": "minSize", "value": 100}) .



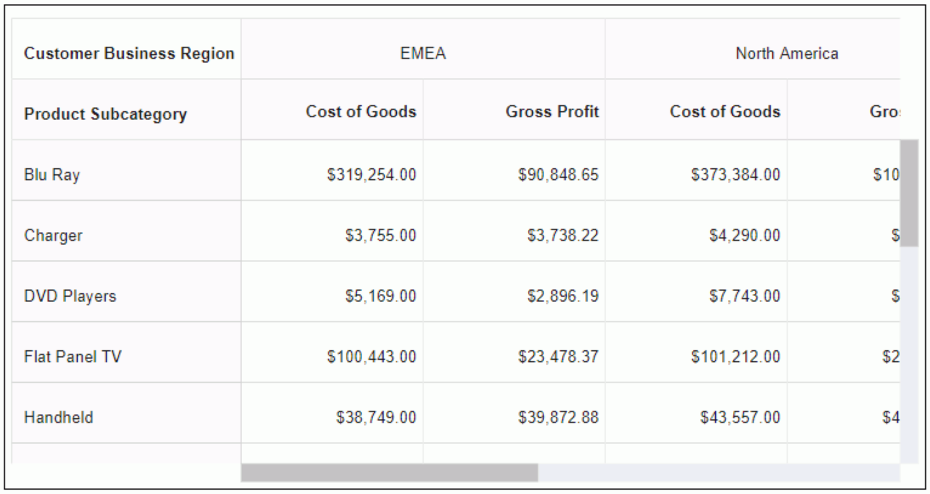
Customer Business Region	EMEA		North America		Oceania	
Product Subcategory	**Cost of Goo...**	**Gross Profit**	**Cost of Goo...**	**Gross Profit**	**Cost of Goo...**	**Gross**
Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.00	$1,1
Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92		
DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02		
Flat Panel TV	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-
Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.00	$3
Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.00	$2
Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.00	$2
Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20		
Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68		
Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$3
Speaker Kits	$144,389.00	$46,298.54	$163,451.00	$52,624.66		
Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.00	$1
Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.00	$1
Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.00	$6
Video Editing	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$3
iPod Docking Station	$43,842.00	$25,662.47	$55,383.00	$33,111.58	$891.00	$6

Example: Formatting the Data Grid Cell Padding

The following request generates a data grid where the minimum width of a cell is 150 pixels and the padding in each cell is 20 pixels on the top, 5 pixels on the right, 15 pixels on the bottom, and 10 pixels on the left.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties":
 {
  "cell":{
     "layout":
             {
              "width": {"rule": "minSize", "value": 150},
              "cellPadding": {"top": 20,
                              "right": 5,
                              "bottom": 15,
                              "left": 10}
            }
         }
 }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America	
Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gro
Blu Ray	$319,254.00	$90,848.65	$373,384.00	$10
Charger	$3,755.00	$3,738.22	$4,290.00	$
DVD Players	$5,169.00	$2,896.19	$7,743.00	$
Flat Panel TV	$100,443.00	$23,478.37	$101,212.00	$2
Handheld	$38,749.00	$39,872.88	$43,557.00	$4

Syntax: How to Format the Inner Grid Lines in a Data Grid

Inner grid lines are the divider lines between inner cells and the right (for horizontal grid lines) or bottom (for vertical grid lines) edge of the grid.

```
"dataGridProperties":
 "cell":{
  "innerGridLines": {
   "horizontal": {
    "width": number,
    "color": "string",
    "dash": "string"
              },
   "vertical": {
    "width": number,
    "color": "string",
    "dash": "string"
               }
                 }
       }
                                }
```

where:

`"width": number`

Is a number of pixels that defines the width of the horizontal inner grid lines. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the horizontal inner grid lines. The default value is "#E0E0E0".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the horizontal inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Multiple dashes can be defined within the dash string (for example, "2 4 4 2"), in which case, the dash types will alternate along the inner grid lines.

`"width":` *`number`*

Is a number of pixels that defines the width of the vertical inner grid lines. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the vertical inner grid lines. The default value is "#EEEEEE".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the vertical inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

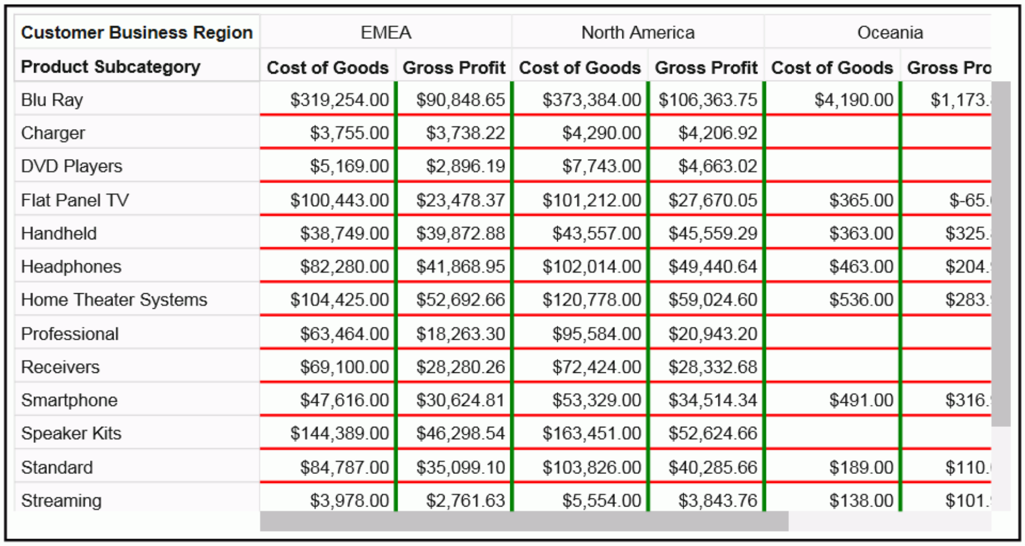
Multiple dashes can be defined within the dash string (for example, "2 4 4 2"), in which case, the dash types will alternate along the inner grid lines.

Example: Formatting the Inner Grid Lines of a Data Grid

The following request makes the horizontal grid lines red, with a width of 2, and the vertical grid lines green, with a width of 3.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"cell": {
    "innerGridLines": {
        "horizontal": {"width": 2, "color": "red", "dash": ""},
        "vertical": {"width": 3, "color": "green", "dash": ""}
    }}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania	
Product Subcategory	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Pro**
Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.00	$1,173.
Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92		
DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02		
Flat Panel TV	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.
Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.00	$325.
Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.00	$204.
Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.00	$283.
Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20		
Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68		
Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.
Speaker Kits	$144,389.00	$46,298.54	$163,451.00	$52,624.66		
Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.00	$110.
Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.00	$101.

Syntax: How to Format the Cell Labels in a Data Grid

The dataGrid:cell:labels properties format the values and tooltips in the grid cells. In order to override default tooltips, you must use a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For more information, see *WebFOCUS Chart Attribute Syntax* on page 143.

```
"dataGridProperties": {
"cell": {
"labels": {
   "font": "string",
   "color": "string",
   "numberFormat": "string",
   "align": "string",
   "valign": "string"
  },
  "tooltip": "string",
  "formatList": [
   {"row": number, "col": number, "repeat": number,
      "color": "string",
     "labels": {"font": "string", "color": "string"}} ...
   ]
 }
 }
 }
```

where:

`"font": "string"`

Is a string that defines the font attributes for the cell labels. The default value is '10pt Sans-Serif'.

`"color": "string"`

Is a string that defines the color of the cell labels. The default value is 'rgb(20, 20, 20)'.

`"numberFormat": "string"`

Is a number format for the cell labels. The default is 'auto'. For information about number formats, see *Introduction to JSON Properties for HTML5 Charts* on page 83.

`"align": "string"`

Is a string that defines the horizontal alignment for the cell labels. The default value is 'right'.

`"valign": "string"`

Is a string that defines the vertical alignment for the cell labels. The default value is 'top'.

`"tooltip": "`*`string`*`"`

Is a custom tooltip for the cell label. A custom tooltip requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

`"formatList"`

Is an array of formats for specific cells that you want to format differently from other cells. A formatList requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143. You can use the following properties for each element in the array.

`"row":` *`number`*

Is a row number.

`"col":` *`number`*

Is a column number.

`"repeat":` *`number`*

Is an increment. After the first row and/or column to use the format, the increment will be added to format additional cells with this format.

`"color": "`*`string`*`"`

Is a string that defines the fill color for the specified cells.

`"font": "`*`string`*`"`

Is a string that defines the font attributes for the specified cells.

`"color": "`*`string`*`"`

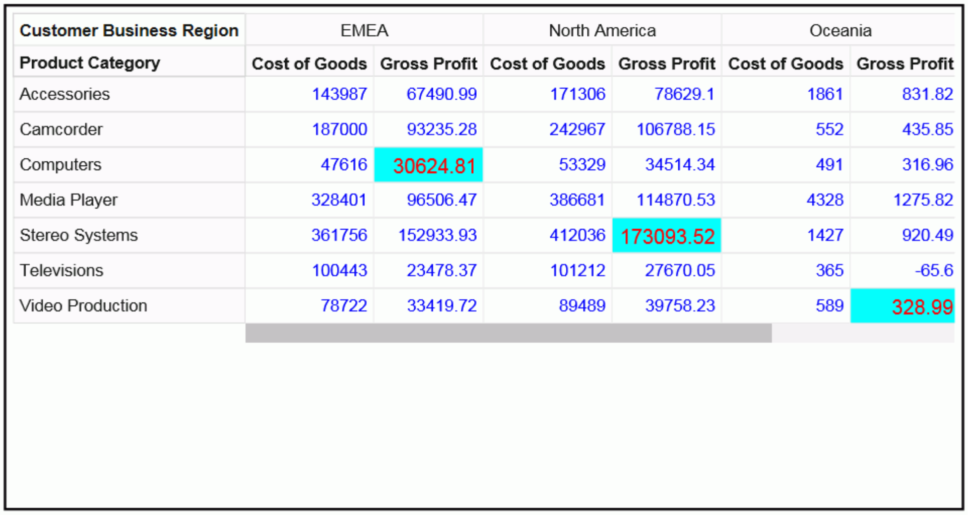
Is a string that defines the color for the labels in the specified cells.

Example: Formatting Cell Labels in a Data Grid

The following request formats most of the cell labels as blue. However, it formats the cell in row 2 and column 1 with a red 12-pt Arial font and an aqua fill color, and applies that format to all cells found by repeatedly incrementing the row and column number by 2.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"cell": {
    "labels": {"color": "blue"},
    "formatList": [
        {"row": 2, "col": 1, "repeat": 2, "color": "aqua",
            "labels": {"font": "12pt arial", "color": "red"}}]
    }}
*END
ENDSTYLE
END
```

The output is shown in the following image. Note that since row and column numbers start at zero (0), row 2 is the third row, and column 1 is the second column.



Customer Business Region	EMEA		North America		Oceania	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	143987	67490.99	171306	78629.1	1861	831.82
Camcorder	187000	93235.28	242967	106788.15	552	435.85
Computers	47616	30624.81	53329	34514.34	491	316.96
Media Player	328401	96506.47	386681	114870.53	4328	1275.82
Stereo Systems	361756	152933.93	412036	173093.52	1427	920.49
Televisions	100443	23478.37	101212	27670.05	365	-65.6
Video Production	78722	33419.72	89489	39758.23	589	328.99

Formatting the Row Header in a Data Grid

Using the dataGrid:rowHeader properties, you can format the row header fill color, border, inner grid lines, divider line, title, and labels.

Syntax: **How to Format the Row Header Fill Color in a Data Grid**

```
"dataGridProperties": {
"rowHeader": {
  "fill": "string",
          }
                        }
```

where:

`"fill": "string"`

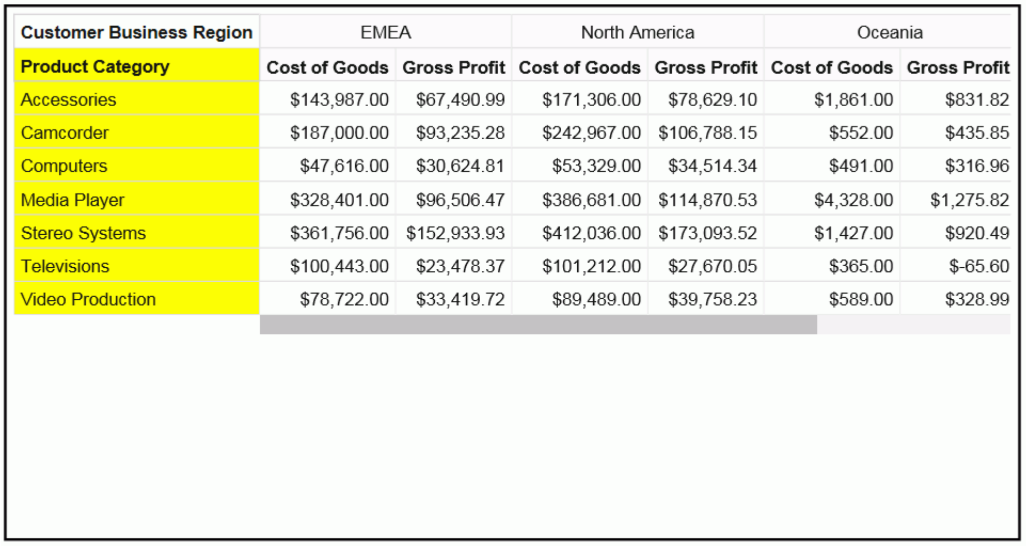
Is a string that defines the fill color for the row header cells. The default value is "#FAFAFA".

Example: **Formatting the Row Header Fill Color in a Data Grid**

The following request makes the data grid row header fill color yellow.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"rowHeader": {"fill": "yellow"}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania	
Product Category	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99

Syntax: How to Format the Row Header Border in a Data Grid

```
"dataGridProperties": {
    "rowHeader": {
     "border": {
   "width": number,
   "color": "string",
   "dash": "string"
  },
  }
}
```

where:

`"width": number`

Is a number that defines the row header border width, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the row header border color. The default value is "#D0D0D0".

`"dash": "string"`

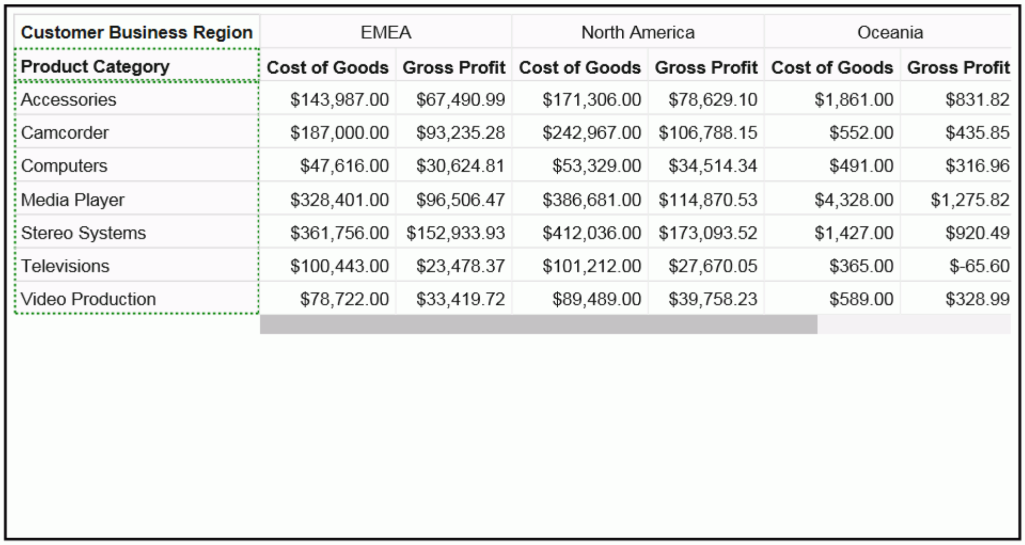
Is a string that defines the dash style of the row header border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Row Header Border in a Data Grid

The following request formats the row header border to be 3 pixels wide, green, and to have a dash style in which each dash is 2 pixels long and the space between dashes is 2 pixels.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"rowHeader": {
    "border": {"width": 3, "color": "green", "dash": "2 2"}
    }}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99

Syntax: How to Format the Row Header Inner Grid Lines in a Data Grid

The row header inner grid lines are the lines between the rows and columns in the row header.

```
"dataGridProperties":{
"rowHeader": {
    "innerGridLines": {
   "horizontal": {
    "width": number,
    "color": "string",
    "dash": "string"
               },
   "vertical": {
    "width": number,
    "color": "string",
    "dash": "string"
             }
                      }
           }
                                             }
```

where:

`"width": number`

Is a number that defines the width of the row header horizontal inner grid lines, in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the row header horizontal inner grid lines. The default value is "#D8D8D8".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the row header horizontal inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

`"width":` *`number`*

Is a number that defines the width of the row header vertical inner grid lines, in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the row header vertical inner grid lines. The default value is "#E0E0E0".

`"dash": "`*`string`*`"`

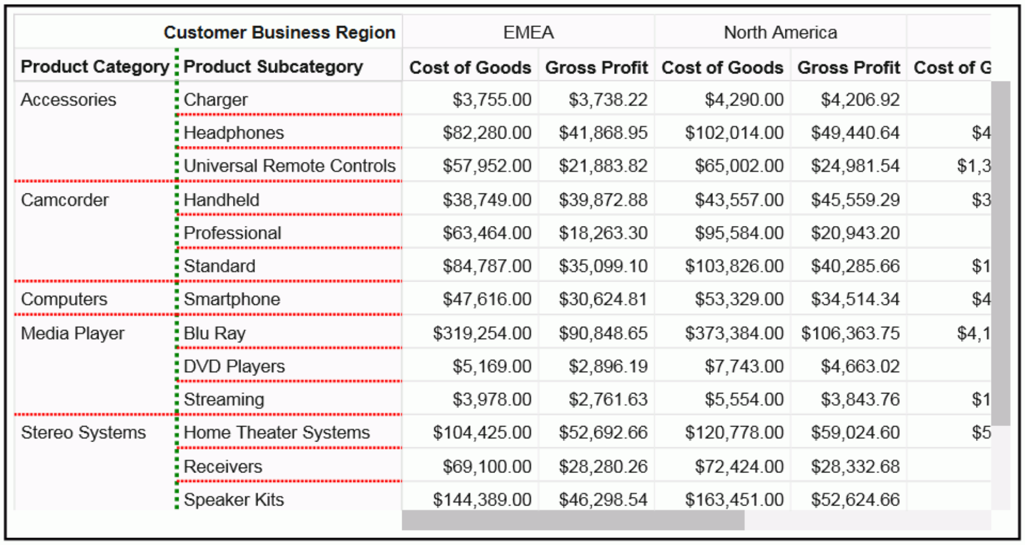
Is a string that defines the dash style of the row header vertical inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Row Header Inner Grid Lines in a Data Grid

The following request formats the row header horizontal inner grid lines to be red, 2 pixels wide, and have dashes that are 2 pixels long with 2 pixels between dashes. The vertical inner grid lines are formatted to be green, 3 pixels wide, and with dashes that are 3 pixels long with 3 pixels between the dashes.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"rowHeader": {
    "innerGridLines": {
        "horizontal": {"width": 2, "color": "red", "dash": "2 1"},
        "vertical": {"width": 3, "color": "green", "dash": "3 3"}
        }}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



	Customer Business Region	EMEA		North America		
Product Category	**Product Subcategory**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of G**
Accessories	Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
	Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$4
	Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,3
Camcorder	Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$3
	Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
	Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$1
Computers	Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$4
Media Player	Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,1
	DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
	Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$1
Stereo Systems	Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$5
	Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68	
	Speaker Kits	$144,389.00	$46,298.54	$163,451.00	$52,624.66	

Syntax: How to Format the Row Header Divider Line in a Data Grid

The row header divider line in a data grid is the line between the row header and the grid body cells.

```
"dataGridProperties":{
"rowHeader": {
    "dividerLine": {
   "width": number,
   "color": "string",
   "dash": "string"
                 },
            }
                       }
```

where:

`"width": number`

Is a number that defines the width of the row header divider line, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the color of the row header divider line. The default value is "#D0D0D0".

"dash": "*string*"

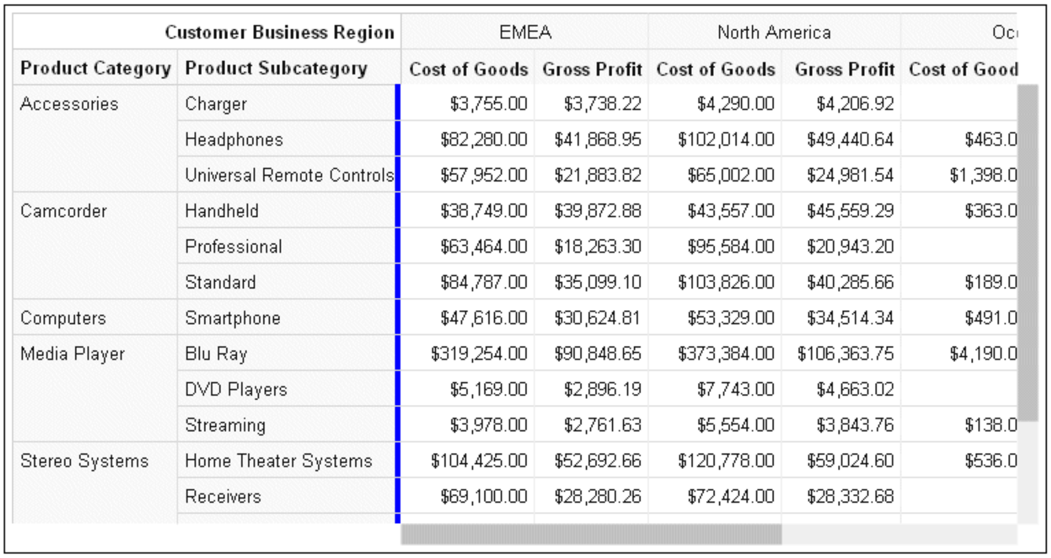
Is a string that defines the dash style of the row header divider line. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Row Header Divider Line in a Data Grid

The following request formats the row header divider line to be 8 pixels wide and blue.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"rowHeader": {
    "dividerLine": {"width": 8, "color": "blue"}}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region		EMEA		North America		Oc
Product Category	**Product Subcategory**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Good**
Accessories	Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
	Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.0
	Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.0
Camcorder	Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.0
	Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
	Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.0
Computers	Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.0
Media Player	Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.0
	DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
	Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.0
Stereo Systems	Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.0
	Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68	

How to Format the Row Header Titles in a Data Grid

Syntax:

Row header titles, by default, are defined in the Master File or the request. In order to override the default titles and tooltips, the properties must be included in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

```
"dataGridProperties":{
"rowHeader": {
   "title": {
   "text": ["string",...],
   "font": "string",
   "color": "string",
   "align": "string",
   "valign": "string",
   "tooltip": "string",
   "dividerLine": {
    "width": number,
    "color": "string",
    "dash": "string"
   }
  },
    }
 }
}
```

where:

`"text": ["`*`string`*`",...]`

Defines the row header title text. It can be a single string or an array of strings. Nested row titles are not supported. The default row header title is the AS name from the request or the field name or title from the Master File. Changing the row header titles requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

`"font": "`*`string`*`"`

Is a string that defines the font attributes for the row header titles. The default value is "bold 10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the row header titles. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment for the row header titles. The default value is "left".

`"valign": "`*`string`*`"`

Is a string that defines the vertical alignment for the row header titles. The default value is "top".

`"tooltip": "`*`string`*`"`

Defines a tooltip for the row header titles. The default value is *undefined*. Changing the row header titles requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143. For information about creating custom tooltips using callback functions, see *Introduction to JSON Properties for HTML5 Charts* on page 83.

`"width": `*`number`*

Is a number that defines the width of the row header title divider line, in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the row header title divider line. The default value is "#D0D0D0".

`"dash": "string"`

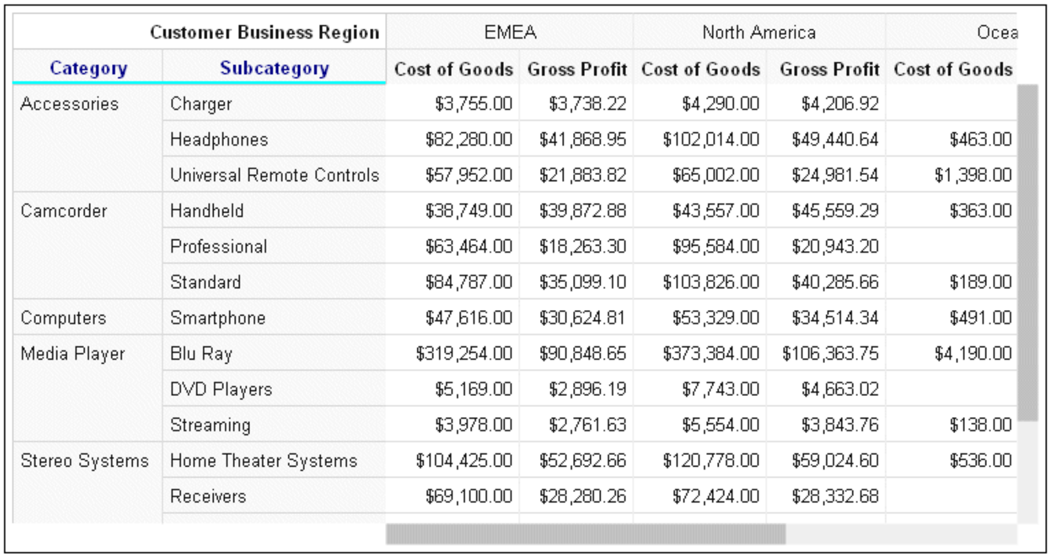
Is a string that defines the dash style of the row header title divider line. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Row Header Titles in a Data Grid

The following request formats the row header titles. The text for the two titles is navy blue and center aligned, with the values *Category* and *Subcategory*. The row header title divider line is aqua and 5 pixels wide. Note that the properties are in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"rowHeader": {
    "title": {
        "text": ["Category", "Subcategory"],
        "font": "bold 10pt Sans-Serif", "color": "navy",
        "align": "center", "valign": "top",
        "dividerLine": {"width": 5, "color": "aqua"}
        }}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



	Customer Business Region	EMEA		North America		Ocea
Category	Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods
Accessories	Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
	Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.00
	Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.00
Camcorder	Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.00
	Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
	Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.00
Computers	Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00
Media Player	Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.00
	DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
	Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.00
Stereo Systems	Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.00
	Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68	

Syntax: How to Format the Row Header Labels in a Data Grid

The row header labels in a data grid are the sort field values for each BY field assigned to the row attribute category. You can format the font, color, horizontal and vertical alignment, and content of the labels. You can also specify whether to merge consecutive labels with the same content or keep them separate.

```
"dataGridProperties":{
 "rowHeader": {
  "labels": {
    "font": "string",
    "color": "string",
    "align": "string",
    "valign": "string",
    "mergeMatching": boolean,
   "content": ["string",...]
  }
 }
}
```

where:

`"font": "string"`

Is a string that defines the font attributes of the row header labels. The default value is "10pt Sans-Serif".

`"color": "*string*"`

Is a string that defines the color of the row header labels. The default value is "rgb(20, 20, 20)".

`"align": "*string*"`

Is a string that defines the horizontal alignment of the row header labels. The default value is "left".

`"valign": "*string*"`

Is a string that defines the vertical alignment of the row header labels. The default value is "top".

`"mergeMatching": *boolean*`

Defines whether to repeat consecutive duplicate labels on each row or merge them. Valid values are:

- true, which merges consecutive duplicate row header labels. This is the default value.
- false, which repeats duplicate row header labels.

`"content": ["*string*",...]`

Defines the row header labels. It can be a single string, an array of strings, or an array of nested objects whose leaves are arrays of strings.

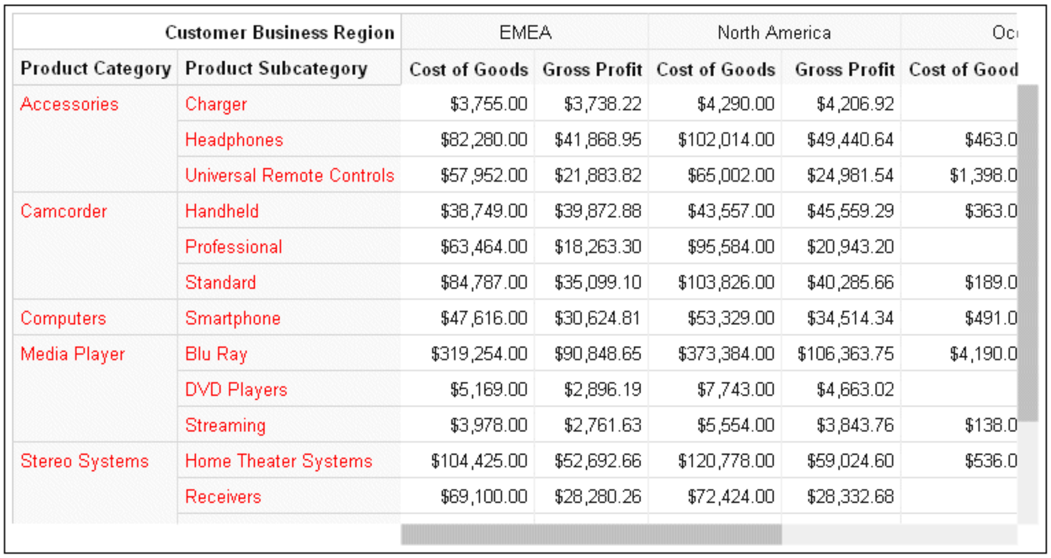
Note: If you replace the labels, your arrays of labels will be applied to the incoming data as you define them, without regard to the actual structure of the row header label arrays.

Example: Formatting the Row Header Labels in a Data Grid

The following request makes the row header labels red and uses the defaults for all other row header label properties.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"rowHeader": {
    "labels": {"color": "red"}
    }}
*END
ENDSTYLE
END
```

The output is shown in the following image.

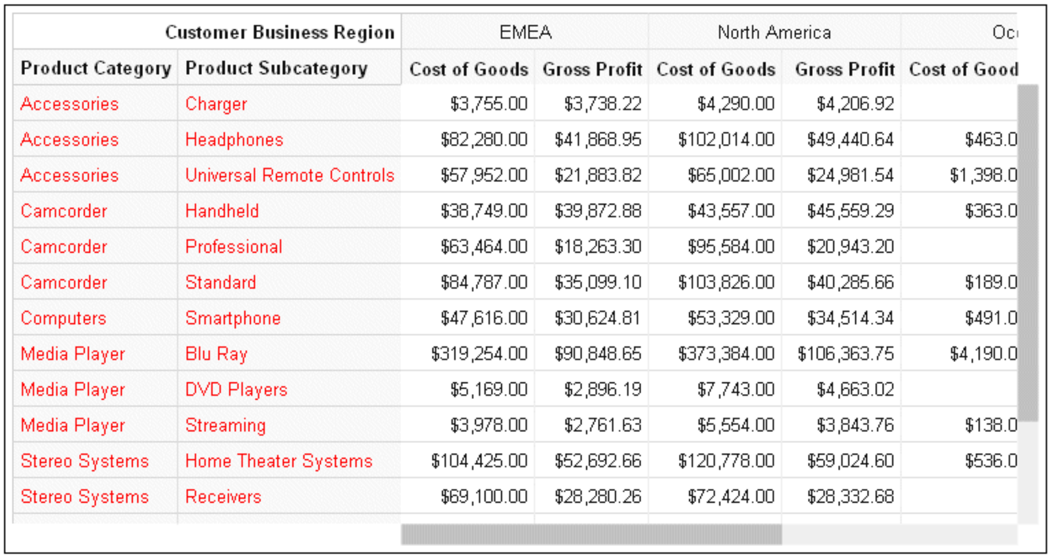


	Customer Business Region	EMEA		North America		Oc
Product Category	Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Good
Accessories	Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
	Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.0
	Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.0
Camcorder	Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.0
	Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
	Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.0
Computers	Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.0
Media Player	Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.0
	DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
	Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.0
Stereo Systems	Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.0
	Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68	

The following version of the request adds "mergeMatching": false to the row header labels properties.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"rowHeader": {
    "labels": {"color": "red", "mergeMatching": false}
    }}
*END
ENDSTYLE
END
```

The generated data grid is shown in the following image in which duplicate row header labels are displayed on each row.

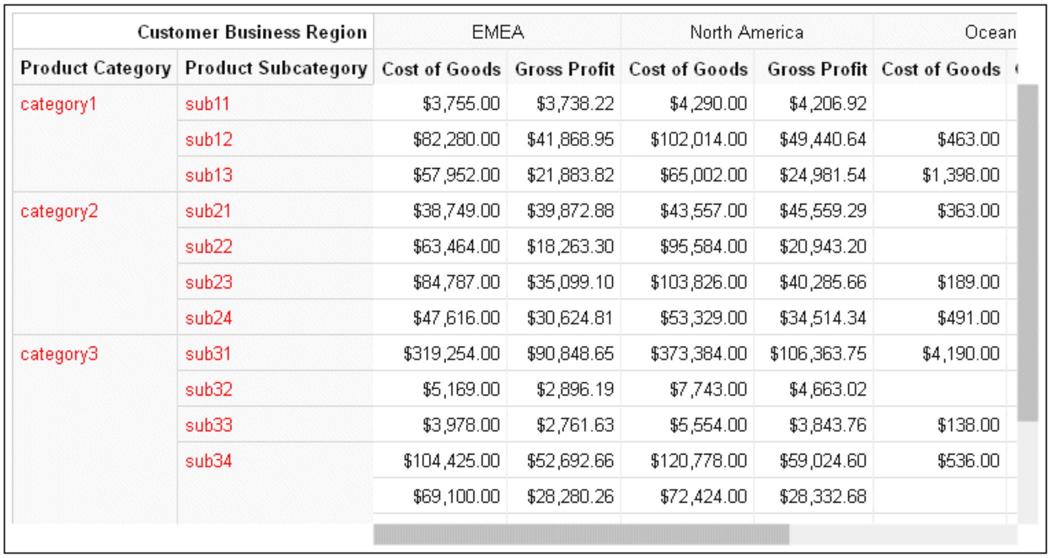


Customer Business Region		EMEA		North America		Oc
Product Category	Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Good
Accessories	Charger	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
Accessories	Headphones	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.0
Accessories	Universal Remote Controls	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.0
Camcorder	Handheld	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.0
Camcorder	Professional	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
Camcorder	Standard	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.0
Computers	Smartphone	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.0
Media Player	Blu Ray	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.0
Media Player	DVD Players	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
Media Player	Streaming	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.0
Stereo Systems	Home Theater Systems	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.0
Stereo Systems	Receivers	$69,100.00	$28,280.26	$72,424.00	$28,332.68	

The following version of the request specifies custom labels in the form of arrays of labels within objects. The structure of the arrays in the request does not match the actual array structure of the labels generated by default.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"rowHeader": {"labels": {
    "font": "10pt Sans-Serif", "color": "red",
    "align": "left", "valign": "top",
    "mergeMatching": true,
    "content": [
        {"category1": ["sub11", "sub12", "sub13"]},
        {"category2": ["sub21", "sub22", "sub23", "sub24"]},
        {"category3": ["sub31", "sub32", "sub33", "sub34"]}
        ]
    }}}
*END
ENDSTYLE
END
```

The output is shown in the following image. The data is assigned to the labels as described in the content object in the request.



Customer Business Region		EMEA		North America		Ocean
Product Category	Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods
category1	sub11	$3,755.00	$3,738.22	$4,290.00	$4,206.92	
	sub12	$82,280.00	$41,868.95	$102,014.00	$49,440.64	$463.00
	sub13	$57,952.00	$21,883.82	$65,002.00	$24,981.54	$1,398.00
category2	sub21	$38,749.00	$39,872.88	$43,557.00	$45,559.29	$363.00
	sub22	$63,464.00	$18,263.30	$95,584.00	$20,943.20	
	sub23	$84,787.00	$35,099.10	$103,826.00	$40,285.66	$189.00
	sub24	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00
category3	sub31	$319,254.00	$90,848.65	$373,384.00	$106,363.75	$4,190.00
	sub32	$5,169.00	$2,896.19	$7,743.00	$4,663.02	
	sub33	$3,978.00	$2,761.63	$5,554.00	$3,843.76	$138.00
	sub34	$104,425.00	$52,692.66	$120,778.00	$59,024.60	$536.00
		$69,100.00	$28,280.26	$72,424.00	$28,332.68	

Formatting the Column Header in a Data Grid

Using the dataGrid:colHeader properties, you can format the column header sorting, sizing, fill color, border, inner grid lines, divider line, title, labels, and last row labels.

The last row of the column header displays the column titles for the measure fields in the data grid. This row must be formatted separately using the dataGrid:colHeader:lastLabels properties.

Syntax: How to Format Interactive Sorting in a Data Grid

If you enable column header sorting, two-headed arrows display in the last row of each column header cell, which indicates that the column is displayed in its original order. Clicking an arrow toggles the chart between ascending, descending, and original order of that column. The arrow changes to an up arrow when the sort order is ascending and a down arrow when the sort order is descending. The arrow displays in bold for the active column.

```
"dataGridProperties":{
"colHeader": {
  "sorting": {
   "enabled": boolean
          }
            }
                       }
```

where:

`"enabled": boolean`

Enables or disables column header sorting. Valid values are:

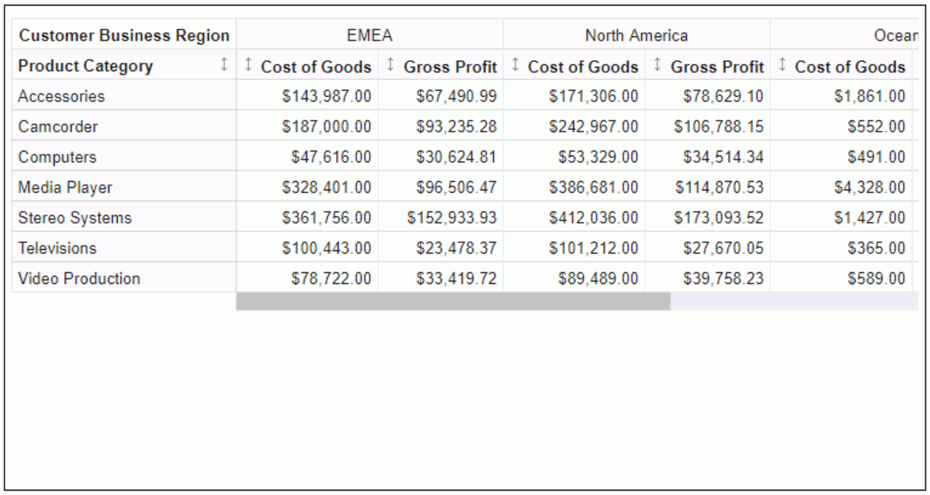
- ❏ true, which enables sorting the grid interactively based on a column.
- ❏ <u>false</u>, which disables interactive grid sorting. This is the default value.

Example: Enabling Interactive Data Grid Sorting

The following request enables interactive data grid sorting.

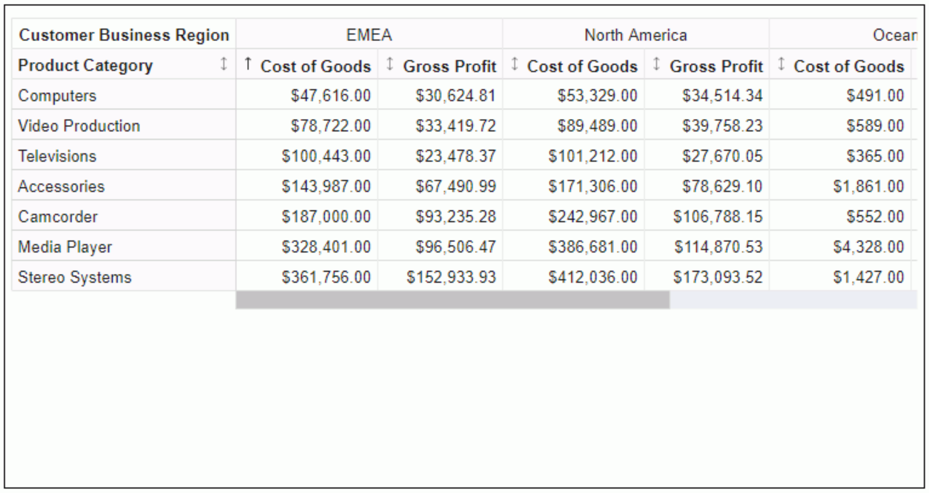
```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {"sorting": {"enabled":
true}}}
*END
ENDSTYLE
END
```

The output is shown in the following image. Clicking an arrow in a column header toggles between ascending, descending, and original order of values of that column.



Customer Business Region	EMEA		North America		Ocear
Product Category ↕	↕ Cost of Goods	↕ Gross Profit	↕ Cost of Goods	↕ Gross Profit	↕ Cost of Goods
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00

The following image shows the data grid sorted in ascending order of the Cost of Goods column under EMEA. Note that the arrow changes to an up arrow and becomes bold.



Customer Business Region	EMEA		North America		Ocean
Product Category ↕	↑ Cost of Goods	↕ Gross Profit	↕ Cost of Goods	↕ Gross Profit	↕ Cost of Goods
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00

Syntax:

How to Enable Interactive Column Resizing in a Data Grid

If you enable column resizing in a data grid, the user can drag the left edge of a column to resize it.

```
"dataGridProperties":{
"colHeader": {
  "resize": {
   "enabled": boolean
          }
            }
                      }
```

where:

`"enabled": boolean`

Enables or disables column resizing. Valid values are:

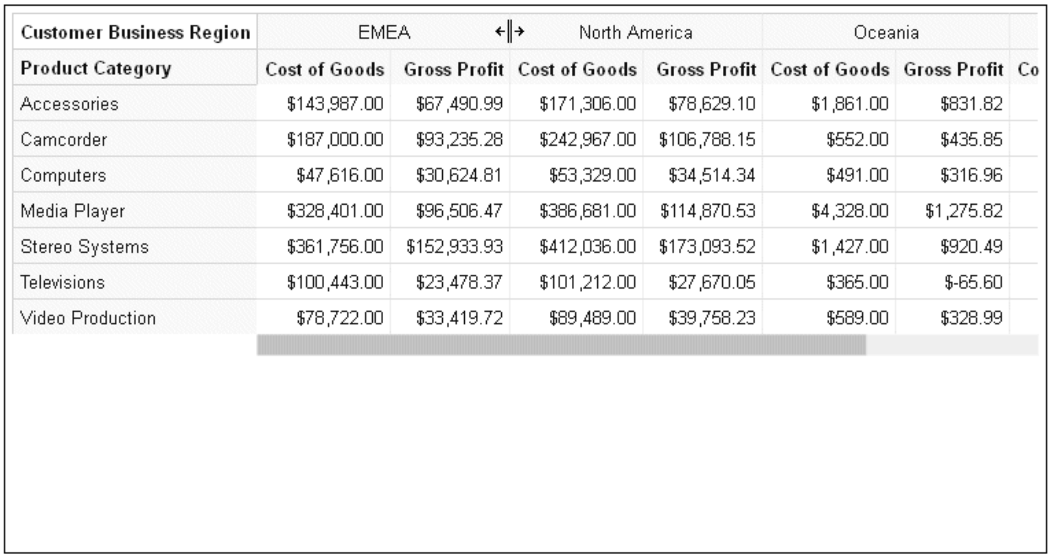
- ❏ true, which enables resizing the grid columns.
- ❏ false, which disables resizing the grid column. This is the default value.

Example: Enabling Interactive Column Resizing in a Data Grid

The following request enables column resizing.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {"resize": {"enabled":
true}}}
*END
ENDSTYLE
END
```

Positioning the mouse on the left edge of a column in the column header, other than the column right next to the row header, displays sizing handles that you can drag to resize the column, as shown in the following image.



Customer Business Region	EMEA		North America		Oceania		
Product Category	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Co**
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82	
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85	
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96	
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82	
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49	
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60	
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99	

Syntax: How to Format the Column Header Fill Color in a Data Grid

```
"dataGridProperties": {
"colHeader": {
  "fill": "string",
          }
                    }
```

where:

`"fill": "string"`

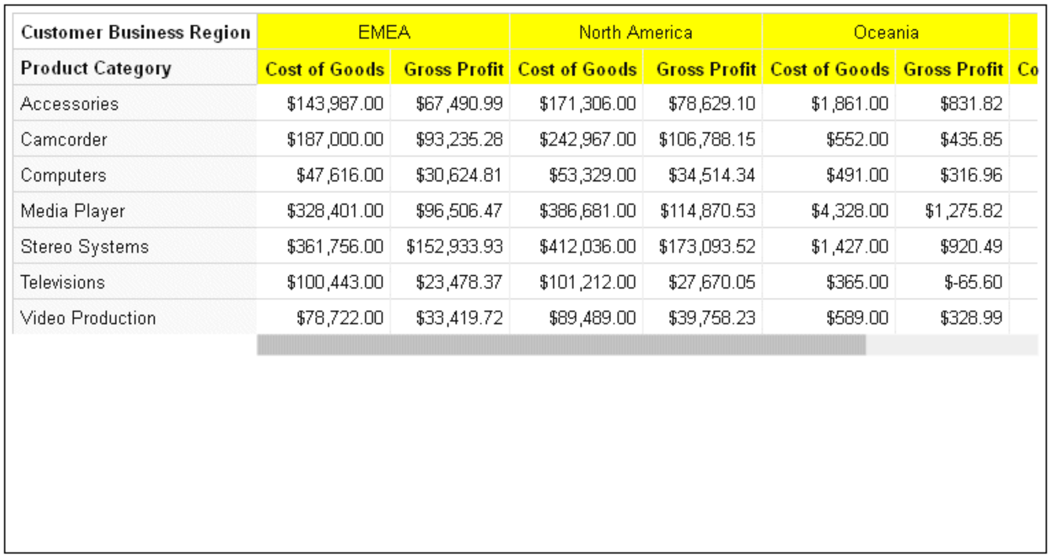
Is a string that defines the fill color for the column header cells. The default value is "#FAFAFA".

Example: Formatting the Column Header Fill Color in a Data Grid

The following request makes the data grid column header fill color yellow.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {"fill": "yellow"}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania		
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Co
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82	
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85	
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96	
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82	
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49	
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60	
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99	

Syntax: **How to Format the Column Header Border in a Data Grid**

```
"dataGridProperties": {
"colHeader": {
  "border": {
   "width": number,
   "color": "string",
   "dash": "string"
  },
  }
}
```

where:

`"width": number`

Is a number that defines the column header border width, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the column header border color. The default value is "#D0D0D0".

`"dash": "*string*"`

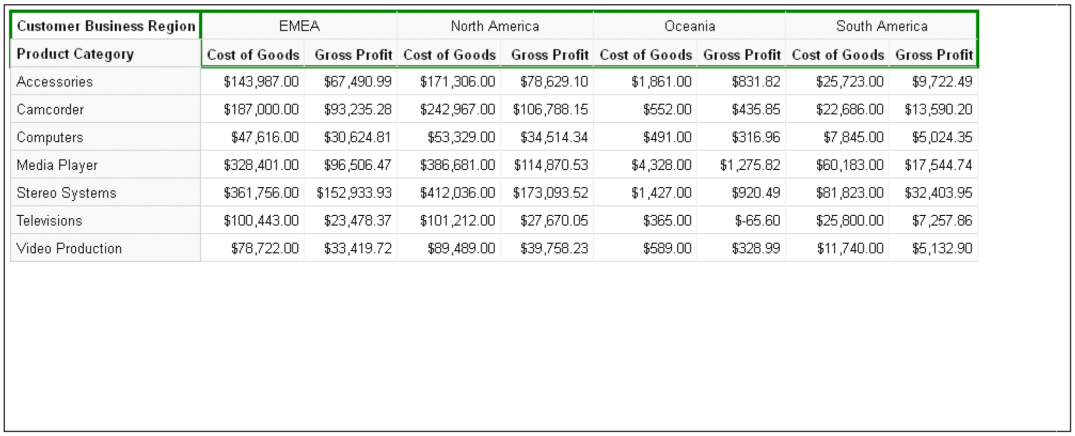
Is a string that defines the dash style of the column header border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Column Header Border in a Data Grid

The following request formats the column header border to be 3 pixels wide and green.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {
    "border": {"width": 3, "color": "green"}}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania		South America	
Product Category	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**	**Cost of Goods**	**Gross Profit**
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82	$25,723.00	$9,722.49
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85	$22,686.00	$13,590.20
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96	$7,845.00	$5,024.35
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82	$60,183.00	$17,544.74
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49	$81,823.00	$32,403.95
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60	$25,800.00	$7,257.86
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99	$11,740.00	$5,132.90

Syntax: How to Format Column Header Inner Grid Lines in a Data Grid

The column header inner grid lines are the lines between the rows and columns in the column header.

```
"dataGridProperties": {
"colHeader": {
    "innerGridLines": {
   "horizontal": {
    "width": number,
    "color": "string",
    "dash": "string"
               },
   "vertical": {
    "width": number,
    "color": "string",
    "dash": "string"
             }
                      }
           }
                                          }
```

where:

`"width": number`

Is a number that defines the width of the column header horizontal inner grid lines, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the color of the column header horizontal inner grid lines. The default value is "#E0E0E0".

`"dash": "string"`

Is a string that defines the dash style of the column header horizontal inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

`"width": number`

Is a number that defines the width of the column header vertical inner grid lines, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the color of the column header vertical inner grid lines. The default value is "#ECECEC".

`"dash": "string"`

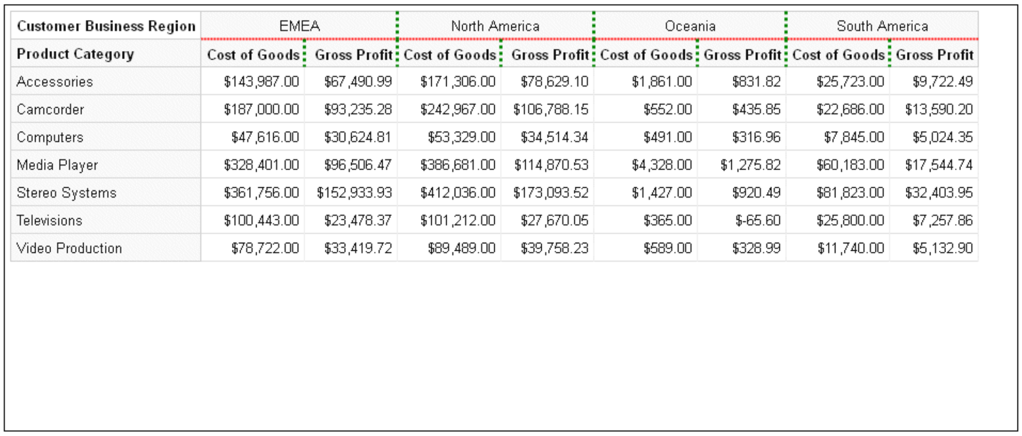
Is a string that defines the dash style of the column header vertical inner grid lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting Column Header Inner Grid Lines in a Data Grid

The following request formats the column header horizontal inner grid lines to be red, 2 pixels wide, and have dashes that are 2 pixels long with 2 pixels between dashes. The vertical inner grid lines are formatted to be green, 3 pixels wide, and with dashes that are 3 pixels long with 3 pixels between the dashes.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET HAXIS 950
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {
    "innerGridLines": {
        "horizontal": {"width": 2, "color": "red", "dash": "2 1"},
        "vertical": {"width": 3, "color": "green", "dash": "3 3"}
}}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania		South America	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82	$25,723.00	$9,722.49
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85	$22,686.00	$13,590.20
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96	$7,845.00	$5,024.35
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82	$60,183.00	$17,544.74
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49	$81,823.00	$32,403.95
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60	$25,800.00	$7,257.86
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99	$11,740.00	$5,132.90

Syntax: How to Format the Column Header Divider Line in a Data Grid

The column header divider line in a data grid is the line between the column header and the grid body cells.

```
"dataGridProperties": {
"colHeader": {
    "dividerLine": {
   "width": number,
   "color": "string",
   "dash": "string"
                },
           }
                      }
```

where:

`"width": number`

Is a number that defines the width of the column header divider line, in pixels. The default value is 1.

`"color": "string"`

Is a string that defines the color of the column header divider line. The default value is "#D0D0D0".

`"dash": "*string*"`

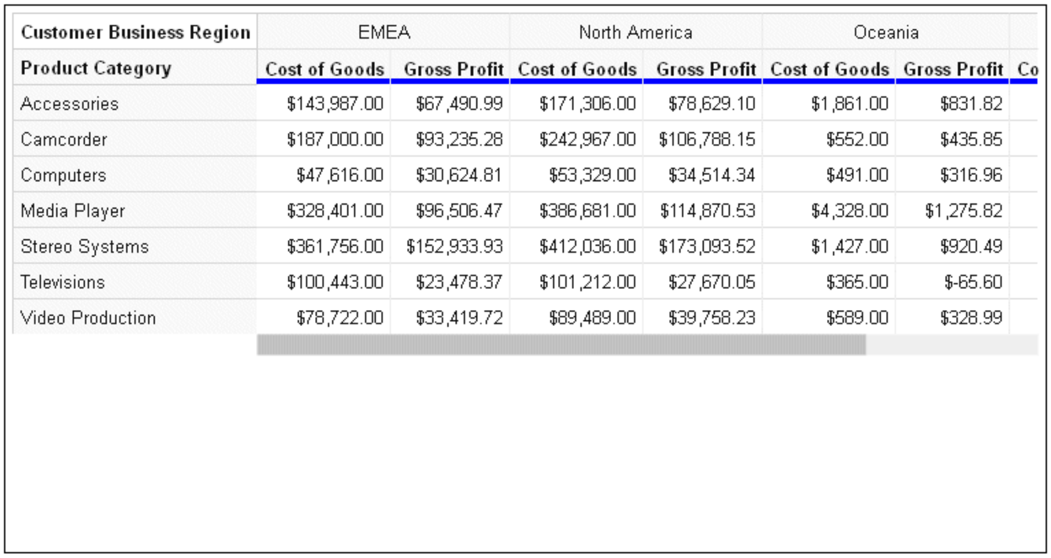
Is a string that defines the dash style of the column header divider line. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Column Header Divider Line in a Data Grid

The following request formats the column header divider line to be 8 pixels wide and blue.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS
"dataGridProperties": {"colHeader": {
    "dividerLine": {"width": 8, "color": "blue"}}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	EMEA		North America		Oceania		
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Co
Accessories	$143,987.00	$67,490.99	$171,306.00	$78,629.10	$1,861.00	$831.82	
Camcorder	$187,000.00	$93,235.28	$242,967.00	$106,788.15	$552.00	$435.85	
Computers	$47,616.00	$30,624.81	$53,329.00	$34,514.34	$491.00	$316.96	
Media Player	$328,401.00	$96,506.47	$386,681.00	$114,870.53	$4,328.00	$1,275.82	
Stereo Systems	$361,756.00	$152,933.93	$412,036.00	$173,093.52	$1,427.00	$920.49	
Televisions	$100,443.00	$23,478.37	$101,212.00	$27,670.05	$365.00	$-65.60	
Video Production	$78,722.00	$33,419.72	$89,489.00	$39,758.23	$589.00	$328.99	

Syntax: **How to Format Column Header Titles in a Data Grid**

Column header titles, by default, are defined in the Master File or the request. In order to override the default titles and tooltips, the properties must be included in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

```
"dataGridProperties":{
"colHeader": {
   "title": {
   "text": ["string",...],
   "font": "string",
   "color": "string",
   "align": "string",
   "valign": "string",
   "tooltip": "string",
   "dividerLine": {
    "width": number,
    "color": "string",
    "dash": "string"
   }
  },
    }
 }
}
```

where:

`"colHeader":`

Defines the properties of the data grid column header.

`"text": ["`*`string`*`",...]`

Defines the column header title text. It can be a single string or an array of strings. Nested column titles are not supported. The default column header title is the field name or title from the Master File or the AS name from the request. Changing the row header titles requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143.

`"font": "`*`string`*`"`

Is a string that defines the font attributes for the column header titles. The default value is "bold 10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the column header titles. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment for the column header titles. The default value is "right".

`"valign": "`*`string`*`"`

Is a string that defines the vertical alignment for the column header titles. The default value is "top".

`"tooltip": "`*`string`*`"`

Defines a tooltip for the column header titles. The default value is *undefined*. Changing the column header tooltip requires that the properties be in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet. For information, see *WebFOCUS Chart Attribute Syntax* on page 143. For information about creating custom tooltips using callback functions, see *Introduction to JSON Properties for HTML5 Charts* on page 83

`"dividerLine":`

Defines the properties of the column header title divider line.

`"width":` *`number`*

Is a number that defines the width of the column header title divider line in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the column header title divider line. The default value is "#EEEEEE".

`"dash": "`*`string`*`"`

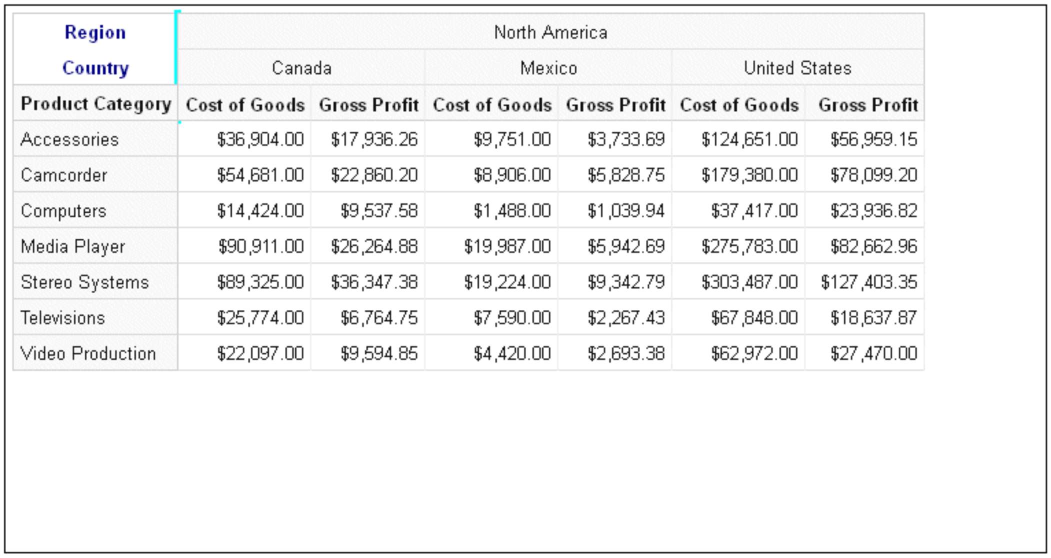
Is a string that defines the dash style of the column header title divider line. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Column Header Titles in a Data Grid

The following request formats the column header titles. The text for the two titles is navy blue and center aligned, with the values *Region* and *Country*. The column header title divider line is aqua and 5 pixels wide. Note that the properties are in a *GRAPH_JS_FINAL block in the WebFOCUS StyleSheet.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'North America'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"colHeader": {
    "title": {
        "text": ["Region", "Country"],
        "font": "bold 10pt Sans-Serif", "color": "navy",
        "align": "center", "valign": "top",
        "dividerLine": {"width": 5, "color": "aqua"}
        }}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Region	North America					
Country	Canada		Mexico		United States	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$36,904.00	$17,936.26	$9,751.00	$3,733.69	$124,651.00	$56,959.15
Camcorder	$54,681.00	$22,860.20	$8,906.00	$5,828.75	$179,380.00	$78,099.20
Computers	$14,424.00	$9,537.58	$1,488.00	$1,039.94	$37,417.00	$23,936.82
Media Player	$90,911.00	$26,264.88	$19,987.00	$5,942.69	$275,783.00	$82,662.96
Stereo Systems	$89,325.00	$36,347.38	$19,224.00	$9,342.79	$303,487.00	$127,403.35
Televisions	$25,774.00	$6,764.75	$7,590.00	$2,267.43	$67,848.00	$18,637.87
Video Production	$22,097.00	$9,594.85	$4,420.00	$2,693.38	$62,972.00	$27,470.00

Syntax:

How to Format Column Header Labels in a Data Grid

The column header labels in a data grid are the sort field values for each BY field assigned to the column attribute category. You can format the font, color, horizontal and vertical alignment, and content of the labels. You can also specify whether to merge consecutive labels with the same content or keep them separate.

```
"dataGridProperties":{
 "colHeader": {
  "labels": {
    "font": "string",
    "color": "string",
    "align": "string",
    "valign": "string",
    "mergeMatching": boolean,
   "content": ["string",...]
  }
 }
}
```

where:

`"font": "string"`

Is a string that defines the font attributes of the column header labels. The default value is "10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the column header labels. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment of the column header labels. The default value is "center".

`"valign": "`*`string`*`"`

Is a string that defines the vertical alignment of the column header labels. The default value is "top".

`"mergeMatching": `*`boolean`*

Defines whether to repeat duplicate consecutive labels on each column or merge them. Valid values are:

- true, which merges duplicate column header labels. This is the default value.
- false, which repeats duplicate row header labels.

`"content": ["`*`string`*`",...]`

Defines the column header labels. Can be a single string, an array of strings, or an array of nested objects whose leaves are arrays of strings. You must account for the last row of the column header (the row that contains the column titles for the measure fields) in your arrays.

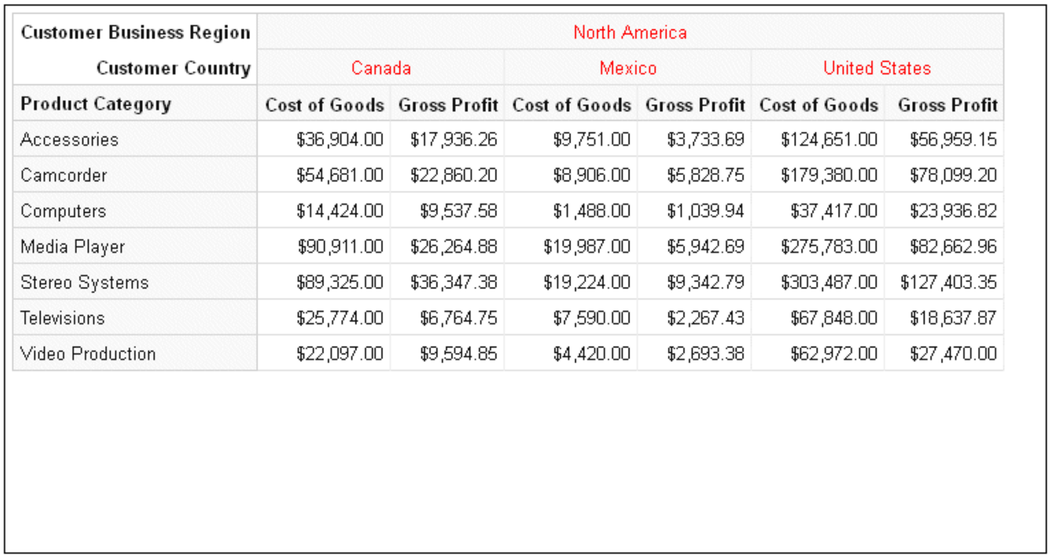
Note: If you replace the labels, your arrays of labels will be applied to the incoming data as you define them, without regard to the actual structure of the column header label arrays.

Example: Formatting the Column Header Labels in a Data Grid

The following request makes the column header labels red and uses the defaults for all other column header label properties.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'North America'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"colHeader": {"labels": {"color": "red"}}}

*END
ENDSTYLE
END
```

The output is shown in the following image.

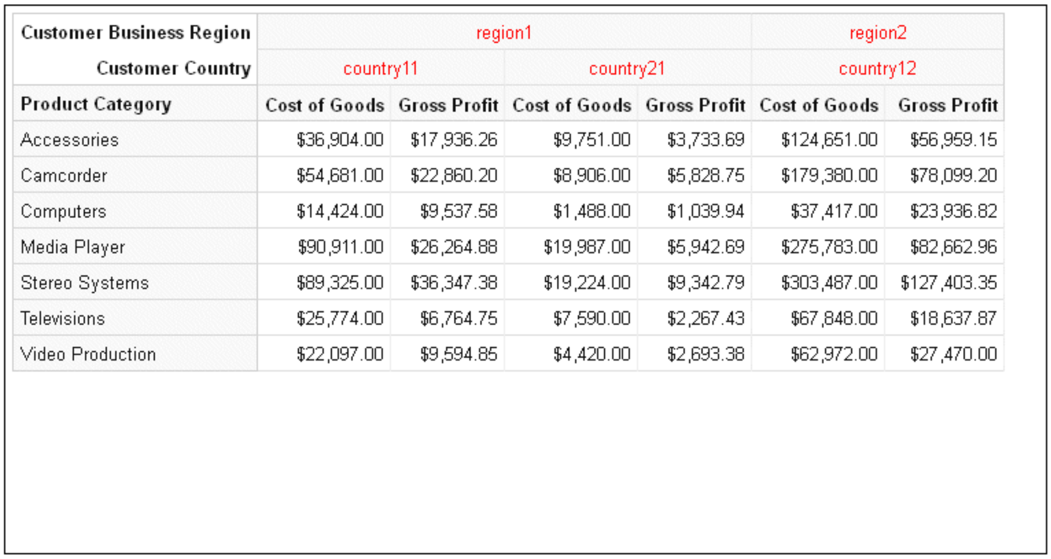


Customer Business Region	North America					
Customer Country	Canada		Mexico		United States	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$36,904.00	$17,936.26	$9,751.00	$3,733.69	$124,651.00	$56,959.15
Camcorder	$54,681.00	$22,860.20	$8,906.00	$5,828.75	$179,380.00	$78,099.20
Computers	$14,424.00	$9,537.58	$1,488.00	$1,039.94	$37,417.00	$23,936.82
Media Player	$90,911.00	$26,264.88	$19,987.00	$5,942.69	$275,783.00	$82,662.96
Stereo Systems	$89,325.00	$36,347.38	$19,224.00	$9,342.79	$303,487.00	$127,403.35
Televisions	$25,774.00	$6,764.75	$7,590.00	$2,267.43	$67,848.00	$18,637.87
Video Production	$22,097.00	$9,594.85	$4,420.00	$2,693.38	$62,972.00	$27,470.00

The following version of the request specifies custom labels in the form of arrays of labels within objects. In order to describe the arrays and objects correctly, you must account for the last row in the column header, which contains the column titles of the measure fields. In this example, the structure of the arrays in the request does not match the actual array structure of the default labels.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'North America'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"colHeader": {
    "labels": {"color": "red",
        "content": [{"region1": [{"country11": ["Cost of Goods","Gross
Profit"]},
                {"country21": ["Cost of Goods", "Gross Profit"]}],
            "region2": [{"country12": ["Cost of Goods", "Gross Profit"]}]
        }]
    }}}
*END
ENDSTYLE
END
```

The output is shown in the following image. The data is assigned to the labels as described in the content object in the request.



Customer Business Region	region1				region2	
Customer Country	country11		country21		country12	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$36,904.00	$17,936.26	$9,751.00	$3,733.69	$124,651.00	$56,959.15
Camcorder	$54,681.00	$22,860.20	$8,906.00	$5,828.75	$179,380.00	$78,099.20
Computers	$14,424.00	$9,537.58	$1,488.00	$1,039.94	$37,417.00	$23,936.82
Media Player	$90,911.00	$26,264.88	$19,987.00	$5,942.69	$275,783.00	$82,662.96
Stereo Systems	$89,325.00	$36,347.38	$19,224.00	$9,342.79	$303,487.00	$127,403.35
Televisions	$25,774.00	$6,764.75	$7,590.00	$2,267.43	$67,848.00	$18,637.87
Video Production	$22,097.00	$9,594.85	$4,420.00	$2,693.38	$62,972.00	$27,470.00

Syntax:

How to Format the Last Row of Labels in the Column Header of a Data Grid

The last row of column header labels is the row that corresponds to the measure field column titles in the data grid. Using the dataGridProperties:colHeader:lastLabels properties, you can apply formatting to these labels.

```
"dataGridProperties": {
 "colHeader": {
  "lastLabels": {
    "font": "string",
    "color": "string",
    "align": "string",
    "valign": "string",
        }
           }
               }
```

where:

`"font": "string"`

Is a string that defines the font attributes of the last row of column header labels. The default value is "10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the last row of column header labels. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment of the last row of column header labels. The default value is "right".

`"valign": "`*`string`*`"`

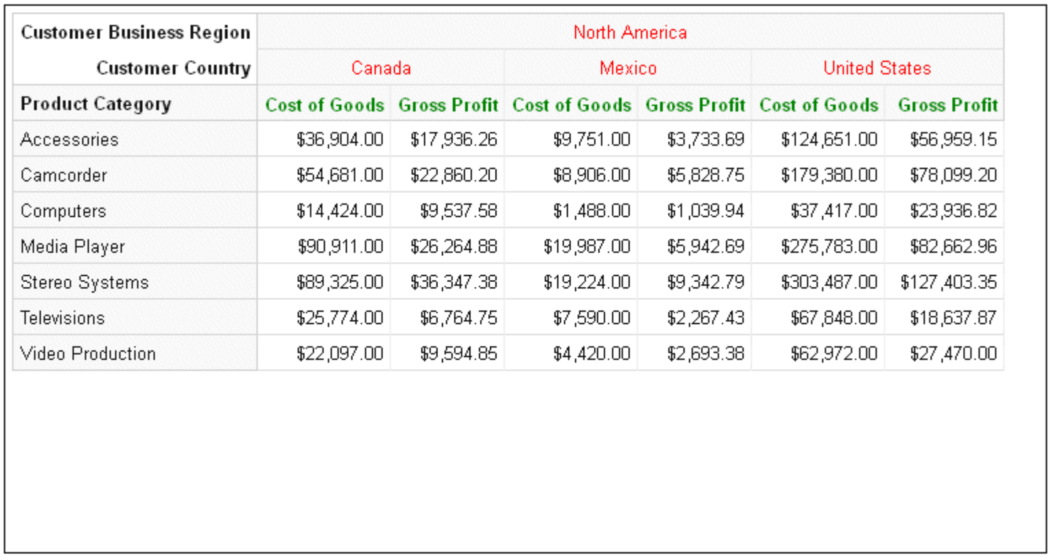
Is a string that defines the vertical alignment of the last row of column header labels. The default value is "center".

Example: Formatting the Last Row of Column Header Labels in a Data Grid

The following request makes the last row of titles in the column header green, while the other rows are red:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'North America'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"colHeader": {
    "labels": {"color": "red"},
    "lastLabels": {"color": "green"}
    }}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	North America					
Customer Country	Canada		Mexico		United States	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$36,904.00	$17,936.26	$9,751.00	$3,733.69	$124,651.00	$56,959.15
Camcorder	$54,681.00	$22,860.20	$8,906.00	$5,828.75	$179,380.00	$78,099.20
Computers	$14,424.00	$9,537.58	$1,488.00	$1,039.94	$37,417.00	$23,936.82
Media Player	$90,911.00	$26,264.88	$19,987.00	$5,942.69	$275,783.00	$82,662.96
Stereo Systems	$89,325.00	$36,347.38	$19,224.00	$9,342.79	$303,487.00	$127,403.35
Televisions	$25,774.00	$6,764.75	$7,590.00	$2,267.43	$67,848.00	$18,637.87
Video Production	$22,097.00	$9,594.85	$4,420.00	$2,693.38	$62,972.00	$27,470.00

Generating and Formatting Column Totals in a Data Grid

Using the dataGridProperties:columnTotals properties, you can generate column totals on the data grid and format their properties, calculations, titles, and divider lines.

Syntax: How to Generate and Format Column Totals on a Data Grid

```
"dataGridProperties":
{
"columnTotals": {
  "visible": boolean,
  "calculation": "string",
  "headerLabel": {
   "text": "string",
   "font": "string",
   "color": "string",
   "align": "string",
   "valign": "string"
  },
  "cellLabel": {
   "font": "string",
   "color": "string",
   "align": "string",
   "valign": "string"
  },
  "border": {
   "width": number,
   "color": "string",
   "dash": "string"
  },
  "dividerLine": {
   "width": number,
   "color": "string",
   "dash": "string"
  }
 }
}
```

where:

`"visible": boolean`

Controls whether column totals display on the data grid. Valid values are:

- true, which displays column totals.
- false, which does not display column totals. This is the default value.

`"calculation": "string"`

Controls how the column totals are calculated. Valid values are

- "sum", which adds the values in each column to produce the column total. This is the default value.
- "min", which calculates the minimum value in each column to produce the column total.

- "max", which calculates the maximum value in each column to produce the column total.
- "mean", which averages the values in each column to produce the column total.

`"headerLabel":`

Defines the properties of the header label for the column total row.

`"text": "`*`string`*`"`

Is a string that defines the header label for the column total row. The default value is "TOTAL".

`"font": "`*`string`*`"`

Is a string that defines the font attributes for the column total header label. The default value is "bold 10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the column total header label. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment of the column total header label. The default value is "left".

`"valign": "`*`string`*`"`

Is a string that defines the vertical alignment of the column total header label. The default value is "center".

`"cellLabel":`

Defines the properties of the cells in the column total row.

`"font": "`*`string`*`"`

Is a string that defines the font attributes of the values in the cells of the column total row. the default value is "bold 10pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the values in the cells of the column total row. The default value is "rgb(20, 20, 20)".

`"align": "`*`string`*`"`

Is a string that defines the horizontal alignment of the values in the cells of the column total row. The default value is "right".

`"valign": "`*`string`*`"`

Is a string that defines the vertical alignment of the values in the cells of the column total row. The default value is "top".

`"border":`

Defines the properties of the border around the bottom and sides of the column total row.

`"width":` *`number`*

Is a number that defines the width of the border around the bottom and sides of the column total row, in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that defines the color of the border around the bottom and sides of the column total row. The default value is "#D0D0D0".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border around the bottom and sides of the column total row. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

`"dividerLine"`

Defines the properties of the divider line above the column total row.

`"width":` *`number`*

Is a number that defines the width of the divider line on top of the column total row, in pixels. The default value is 2.

`"color": "`*`string`*`"`

Is a string that defines the color of the divider line on top of the column total row. The default value is "#D0D0D0".

`"dash": "`*`string`*`"`

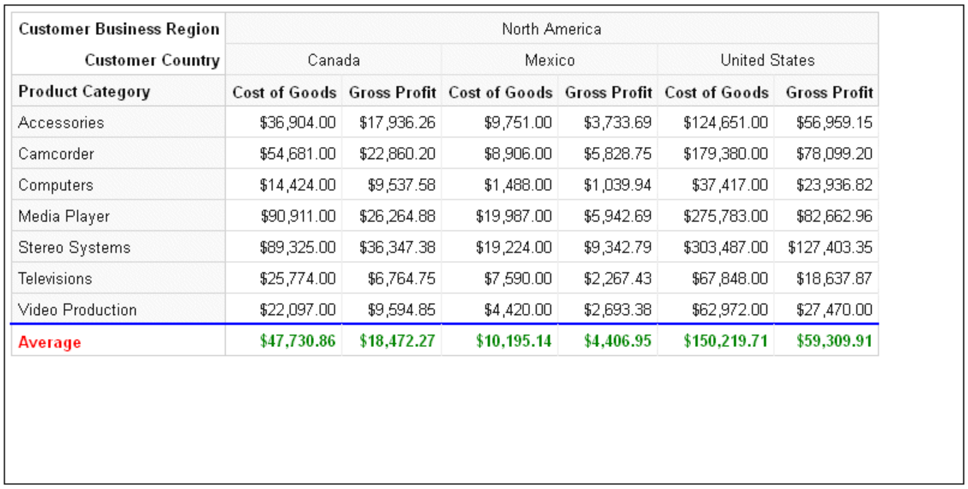
Is a string that defines the dash style of the divider line on top of the column total row. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: **Generating and Formatting a Column Total Row on a Data Grid**

The following request generates a column total row using the "mean" calculation. The header label for the column total row is *Average* and it is red. The calculated values in the cells of the row are green. The divider line is blue.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
BY COUNTRY_NAME
WHERE BUSINESS_REGION EQ 'North America'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET HAXIS 800
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {"columnTotals": {
    "visible": true, "calculation": "mean",
    "headerLabel": {"text": "Average", "color": "red"},
    "cellLabel": {"color": "green"},
    "dividerLine": {"width": 2, "color": "blue"}
}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customer Business Region	North America					
Customer Country	Canada		Mexico		United States	
Product Category	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit
Accessories	$36,904.00	$17,936.26	$9,751.00	$3,733.69	$124,651.00	$56,959.15
Camcorder	$54,681.00	$22,860.20	$8,906.00	$5,828.75	$179,380.00	$78,099.20
Computers	$14,424.00	$9,537.58	$1,488.00	$1,039.94	$37,417.00	$23,936.82
Media Player	$90,911.00	$26,264.88	$19,987.00	$5,942.69	$275,783.00	$82,662.96
Stereo Systems	$89,325.00	$36,347.38	$19,224.00	$9,342.79	$303,487.00	$127,403.35
Televisions	$25,774.00	$6,764.75	$7,590.00	$2,267.43	$67,848.00	$18,637.87
Video Production	$22,097.00	$9,594.85	$4,420.00	$2,693.38	$62,972.00	$27,470.00
Average	$47,730.86	$18,472.27	$10,195.14	$4,406.95	$150,219.71	$59,309.91

Generating and Formatting Scroll Bars on a Data Grid

By default, scroll bars are generated on a data grid when the entire grid does not fit in the draw area. Using the dataGridProperties:scroll properties, you can disable scroll bars, format scroll bars, and specify whether to freeze column and row headers.

Syntax: How to Generate and Format Scroll Bars on a Data Grid

```
"dataGridProperties": {
    "scroll": {
        "enabled": boolean,
  "freezeRowHeader": boolean,
  "freezeColHeader": boolean,
  "size": number,
  "color": "string",
  "handle": {
   "color": "string",
   "hoverColor": "string",
   "border": {
    "width": number,
    "color": "string",
    "dash": "string"
           }
         }
       }
              }
```

where:

`"enabled": ` *`boolean`*

Enables or disables scroll bars on a data grid. Valid values are:

- ❏ true, which enables scroll bars. This is the default value.
- ❏ false, which disables scroll bars.

`"freezeRowHeader": ` *`boolean`*

Defines whether to freeze or scroll the row header on a data grid. Valid values are:

- ❏ true, which freezes the row header. This is the default value.
- ❏ false, which scrolls the row header.

`"freezeColHeader": ` *`boolean`*

Defines whether to freeze or scroll the column header on a data grid. Valid values are:

- ❏ true, which freezes the column header. This is the default value.
- ❏ false, which scrolls the column header.

`"size": ` *`number`*

Is a number that defines the height of the scroll bars, in pixels. The default value is 15.

`"color": "`*`string`*`"`

Is a string that defines the color of the scroll bars. The default value is "rgb(240, 240, 240)".

`"handle":`

Defines the properties of the scroll bar handle.

`"color": "`*`string`*`"`

Is a string that defines the color of the scroll bar handles (the pieces you drag in order to scroll the data grid). The default value is "#C2C2C2".

`"hoverColor": "`*`string`*`"`

Is a string that defines the color of a scroll bar handle when the mouse hovers over it. The default value is "#A8A8A8".

`"border":`

Defines the properties of the scroll bar handle border.

`"width": ` *`number`*

Is a number that defines the width of the scroll bar handle border, in pixels. The default value is zero (0).

`"color": "string"`

Is a string that defines the color of the scroll bar handle border. The default value is "transparent".

`"dash": "string"`

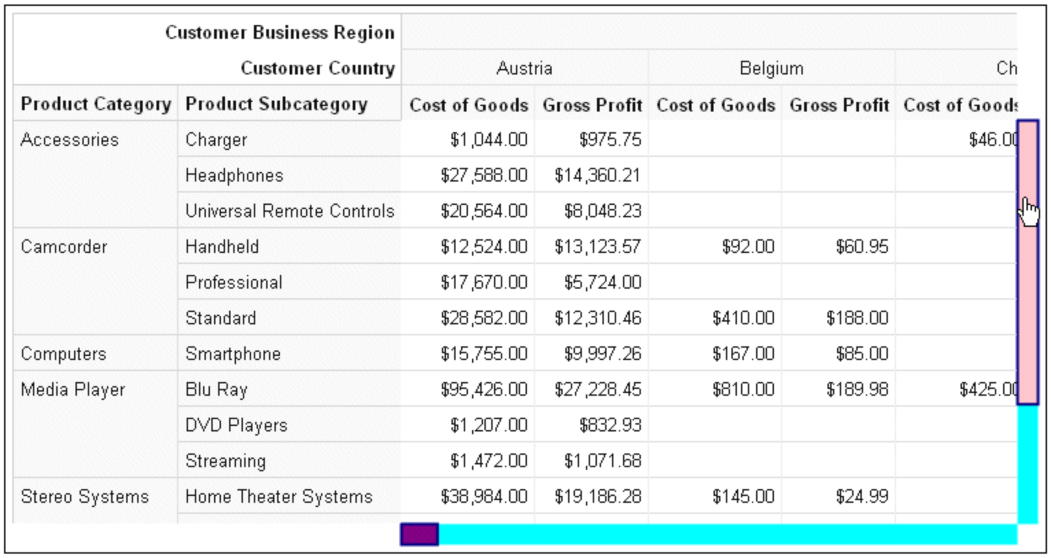
Is a string that defines the dash style of the scroll bar handle border. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

Example: Formatting the Scroll Bars on a Data Grid

The following request makes the scroll bar aqua and the handle purple with a navy 2-pixel border and a pink hover color.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
BY BUSINESS_REGION
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH DATAGRID
ON GRAPH SET STYLE *
type=data, column=product_category,  bucket=row, $
type=data, column=product_subcateg,  bucket=row, $
type=data, column=business_region,   bucket=column, $
type=data, column=country_name,   bucket=column, $
type=data, column=cogs_us,    bucket=measure, $
type=data, column=gross_profit_us, bucket=measure, $
*GRAPH_JS_FINAL
"dataGridProperties": {
    "scroll": {"color": "aqua",
        "handle": {"color": "purple", "hoverColor": "pink",
            "border": {"width": 2, "color": "navy"}}
        }}
*END
ENDSTYLE
END
```

The output is shown in the following image, with the mouse hovering over the vertical scroll bar handle.



	Customer Business Region					
	Customer Country	Austria		Belgium		Ch
Product Category	Product Subcategory	Cost of Goods	Gross Profit	Cost of Goods	Gross Profit	Cost of Goods
Accessories	Charger	$1,044.00	$975.75			$46.00
	Headphones	$27,588.00	$14,360.21			
	Universal Remote Controls	$20,564.00	$8,048.23			
Camcorder	Handheld	$12,524.00	$13,123.57	$92.00	$60.95	
	Professional	$17,670.00	$5,724.00			
	Standard	$28,582.00	$12,310.46	$410.00	$188.00	
Computers	Smartphone	$15,755.00	$9,997.26	$167.00	$85.00	
Media Player	Blu Ray	$95,426.00	$27,228.45	$810.00	$189.98	$425.00
	DVD Players	$1,207.00	$832.93			
	Streaming	$1,472.00	$1,071.68			
Stereo Systems	Home Theater Systems	$38,984.00	$19,186.28	$145.00	$24.99	

Funnel Chart Properties (funnelProperties)

A funnel chart is basically a pie chart that shows only one group of data at a time. The series in the group are stacked in the funnel with the first series at the top and the last series at the bottom. In a funnel chart, the display field functions as the group, and the sort field values function as the series.

These properties control the general layout of a funnel chart.

The following code segment shows the default values:

```
"funnelProperties": {
    "topWidth": "100%",
    "baseWidth": "20%",
    "riserGap": 0,
    "groupLabel": {
        "visible": false,
        "font": "10pt Sans-Serif",
        "color": "black"
        }
    }
```

Series-specific properties control the color of funnel segments.

Controlling the Width at the Top of the Funnel

The topWidth property controls the width of the top of the funnel.

Syntax: How to Control the Width at the Top of the Funnel

```
"funnelProperties": {
    "topWidth": width}
```

where;

`"topWidth": width`

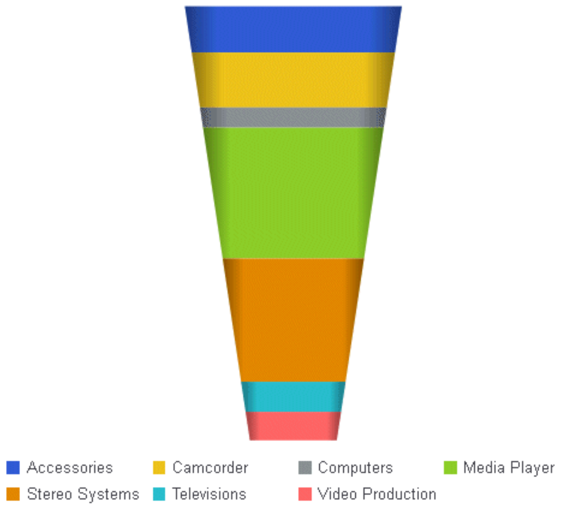
Can be a number that defines the width in pixels or a string that includes a percent symbol, enclosed in double quotation marks ("), that represents a percentage of the available space in the frame (that is, "0%" through "100%"). The default value is "100%".

Example: Controlling the Width at the Top of the Funnel

The following request creates a funnel chart in which the top width is reduced to 50%:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"funnelProperties": {"topWidth": "50%"}
*END
ENDSTYLE
END
```

The output is:



Controlling the Width at the Base of the Funnel

The baseWidth property controls the width of the base of the funnel.

Syntax: How to Control the Width at the Base of the Funnel

```
"funnelProperties": {
   "baseWidth": width}
```

where;

`"baseWidth": width`

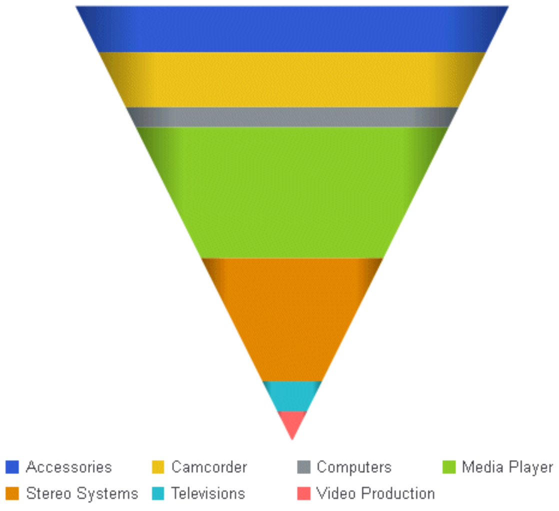
Can be a number that defines the width in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that represents a percentage value (that is, from "0%" to 100%"). The default value is "20%".

Example: Controlling the Width at the Base of the Funnel

The following request creates a funnel chart in which the base width is 1 pixel:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"funnelProperties": {"baseWidth": 1}
*END
ENDSTYLE
END
```

The output is:



Controlling the Space Between Layers in a Funnel Chart

The riserGap property controls the empty space between layers in the funnel.

Syntax: How to Control the Space Between Layers in a Funnel Chart

```
"funnelProperties": {
   "riserGap": gap},
```

where:

`"riserGap": gap`

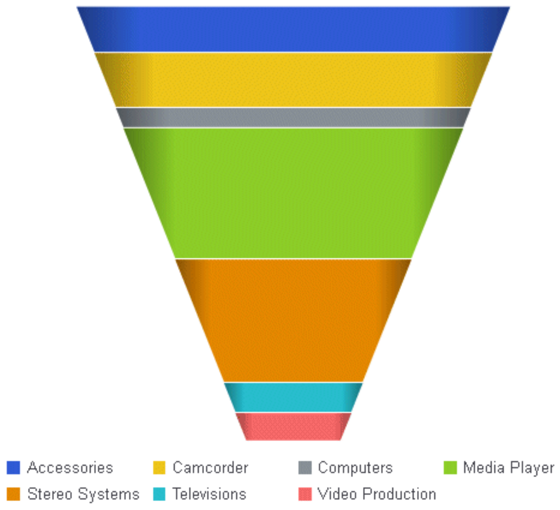
Can be a number that defines the width of the riser gap in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that represents a percentage value "0%" to "100%" (of average riser height). For example, riserGap: "10%" will use 10% of each funnel layer as white space between each segment. The default value is zero (0).

Example: Controlling the Gap Between Funnel Layers

The following request creates a funnel chart in which the gap between layers is 1 pixel:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"funnelProperties": {"riserGap": 1}
*END
ENDSTYLE
END
```

The output is:



Formatting the Group Label in a Funnel Chart

Syntax: How to Format a Funnel Chart Group Label

The groupLabel properties control the visibility and format of the group label in a funnel chart.

These properties do not control the text that displays as the group label. That is controlled by the general groupLabels property. The default group label is the field name for the measure in the request.

```
"funnelProperties": {
   "groupLabel": {
      "visible": boolean,
      "font": "string",
      "color": "string"
   },
},
```

where:

`"visible": boolean`

Valid values are:

- ❑ true, which makes the group label visible.
- ❑ false, which makes the group label not visible. This is the default value.

`"font": "string"`

Is a string that defines the size, style, and, typeface of the label. The default value is "10pt Sans-Serif".

`"color": "string"`

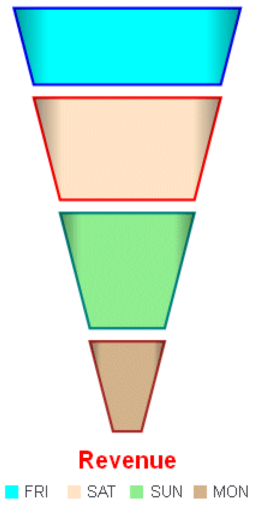
Is a string that defines the color of the label. The default value is "black".

Example: Controlling a Funnel Chart Group Label

The following request creates a funnel chart with a red group label in a 14pt Sans-Serif font. The top width is 50%, the base width is 1 pixel, and the riser gap is 10 pixels. The group label is the measure field name (the default label):

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
ACROSS TIME_DAYNAME COLUMNS 'FRI' AND 'SAT' AND 'SUN' AND 'MON'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"funnelProperties": {"topWidth": "50%", "baseWidth": 1, "riserGap": 10,
    "groupLabel": {
        "visible": true, "font": "bold 14pt Sans-Serif", "color": "red"}},
"series": [
    {"series": 0, "color": "cyan", "border": {"width": 2, "color": "blue"}},
    {"series": 1, "color": "bisque", "border": {"width": 2, "color": "red"}},
    {"series": 2, "color": "lightgreen", "border": {"width": 2, "color": "teal"}},
    {"series": 3, "color": "tan", "border": {"width": 2, "color": "brown"}}
    ]
*END
ENDSTYLE
END
```

The output is:



Gauge Chart Properties (gaugeProperties)

A gauge chart represents one or more values as needles or a single needle and markers on a circular surface. This chart type is typically used by executive dashboard applications.

These properties control the general layout of a gauge chart.

The following code segment shows the default values for the gauge properties:

```
"gaugeProperties": {
    "startAngle": 135,
    "endAngle": 45,
    "secondaryNeedlesAsMarkers": false,
    "secondaryRingFillColor": "undefined",
    "layout": "gauge",
    "groupLabel": {
        "visible": false,
        "font": "10pt Sans-Serif",
        "color": "black"},
    "totalLabel": {
        "visible": false,
        "font": "10pt Sans-Serif",
        "color": "black",
        "numberFormat": "auto"},
    "fill": {
        "color": "transparent"},
    "needleBase": {
        "size": "6%",
        "color": "#1f77b4",
        "border": {
            "width": 0,
            "color": "transparent",
            "dash": ""}
        },
    "axisWidth": "22%",
    "axisTickLength": "30%",
    "axisMinorTickLength": "8%",
    "outerBorder": {
        "width": "10%",
        "fill": {
            "color": "#1f77b4"},
        "border": {
            "width": 0,
            "color": "transparent",
            "dash": ""
            }
        }
    }
```

Note: The yaxis properties control the range and format of values, the format of grid lines, and color bands shown on the gauge axis. By default, minor grid lines are visible and major grid lines are not visible. The axisTickLength property controls the length of major grid lines, if they are visible. The axisMinorTickLength property controls the length of minor grid lines.

Controlling the Layout of a Gauge Chart

By default, a gauge chart has a needle and a band with tick marks. The *simple* layout has only the band and the total label.

Syntax: How to Control the Layout of a Gauge Chart

```
"gaugeProperties": {
   "layout": "string"   },
```

where:

`"layout": "string"`

Can be:

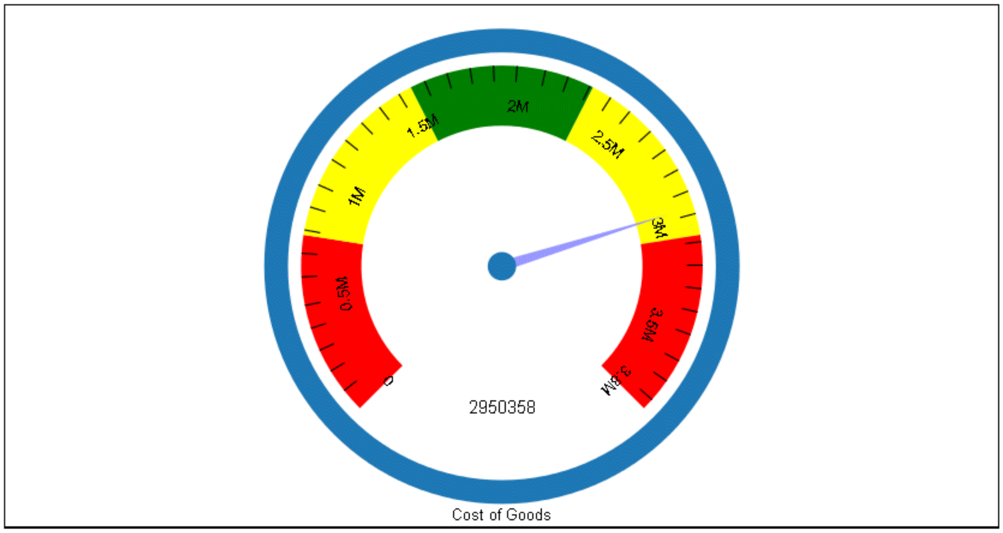
- "gauge". Lays out the chart as a traditional gauge chart with a needle. This is the default value.
- "simple". Lays out the chart with just a band and a total label.

Example: Controlling the Layout of a Gauge Chart

The following request generates a traditional gauge chart with default properties:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
END
```

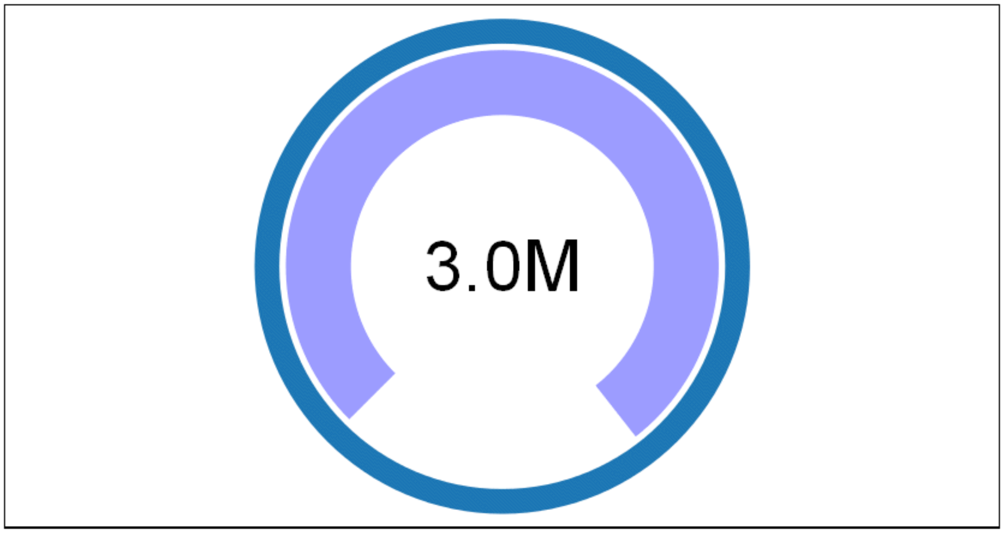
The output is shown in the following image:



The following version of the request sets the gauge chart layout to 'simple':

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"layout": "simple"}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Controlling the Length of Major Axis Tick Marks in a Gauge Chart

The axisTickLength property defines the length of axis major grid tick marks. Note that gauge chart major axis tick marks are not visible, by default.

Syntax: How to Control the Length of Axis Major Grid Tick Marks in a Gauge Chart

```
"gaugeProperties": {
   "axisTickLength": length   },
```

where:

`"axisTickLength": length`

Can be a number that defines the length in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that defines the width as a percentage of the overall gauge. The default value is "30%".

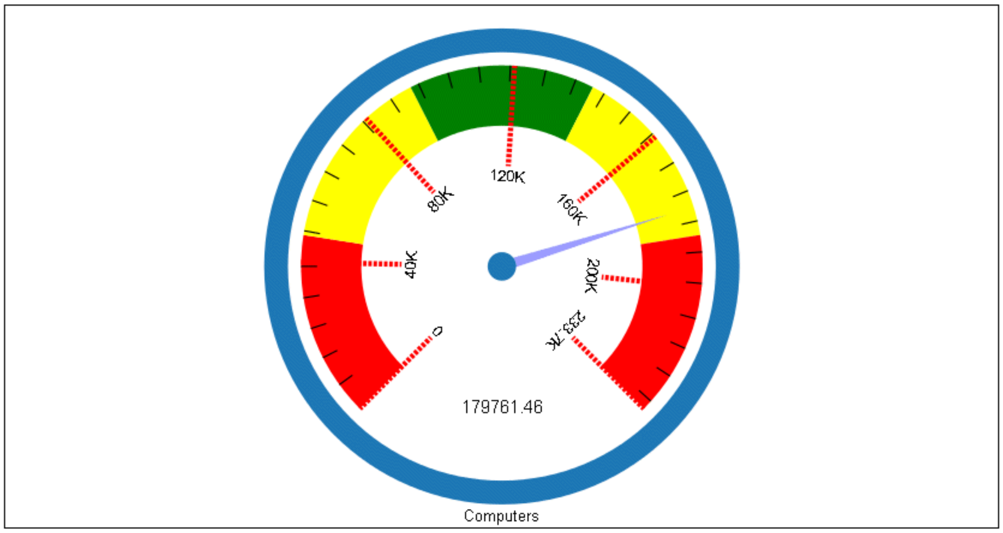
Note: Use the yaxis majorGrid property to format the tick marks.

Example: Controlling the Length of Major Axis Tick Marks in a Gauge Chart

The following request sets the axis tick length to 50%. The y-axis major grid properties set the line style:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"axisTickLength": "50%"},
"yaxis": {"majorGrid": {
    "visible": true, "lineStyle": {
        "width": 4, "color": "red", "dash": "2 2"}
    }}
*END
ENDSTYLE
END
```

The output is:



Controlling the Length of Minor Axis Tick Marks in a Gauge Chart

The axisMinorTickLength property defines the length of axis minor grid tick marks. Note that gauge chart minor axis tick marks are visible, by default.

Syntax: How to Control the Length of Axis Minor Grid Tick Marks in a Gauge Chart

```
"gaugeProperties": {
   "axisMinorTickLength": length   },
```

where:

`"axisMinorTickLength": length`

Can be a number that defines the length in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that defines the width as a percentage of the overall gauge. The default value is "8%".

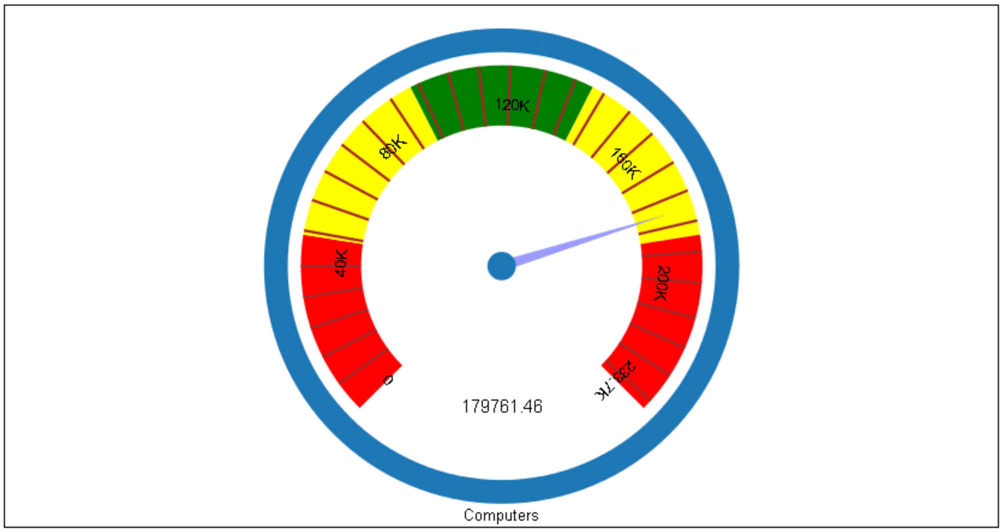
Note: Use the yaxis minorGrid property to format the tick marks.

Example: Controlling the Length of Minor Axis Tick Marks in a Gauge Chart

The following request sets the minor axis tick length to 30%. The y-axis minor grid properties set the line style:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"axisMinorTickLength": "30%"},
"yaxis": {"minorGrid": {
    "lineStyle": {
        "width": 2, "color": "brown", "dash": "2 2"}
    }}
*END
ENDSTYLE
END
```

The output is:



Controlling the Width of the Gauge Axis

The axisWidth property defines the width of the gauge axis.

Syntax: How to Control the Width of the Gauge Axis

```
"gaugeProperties": {
   "axisWidth": width   },
```

where:

`"axisWidth": width`

Can be a number that defines the width in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that defines the width as a percentage of the overall gauge. The default value is "22%".

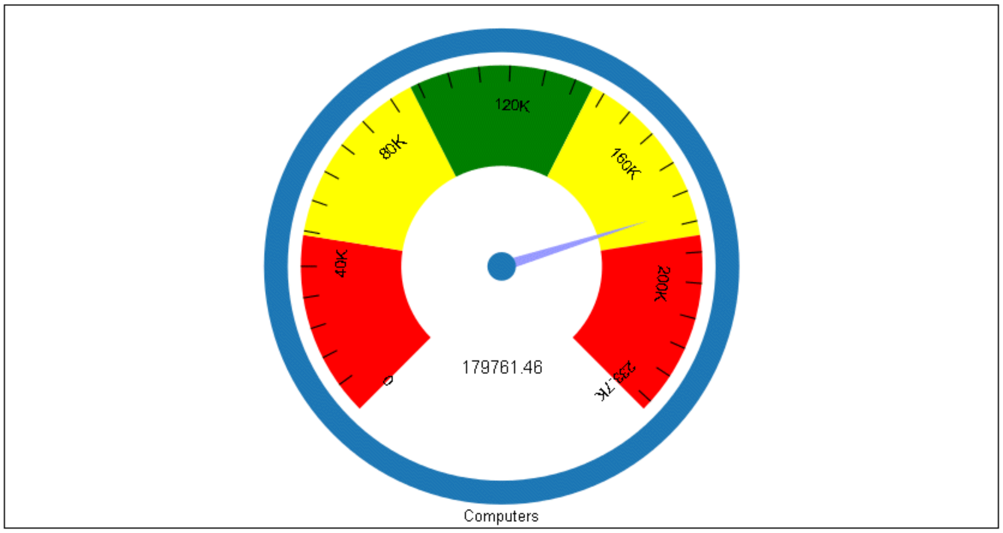
Note: Use the yaxis colorBands property to change the default colors on the gauge axis.

Example: Controlling the Axis Width in a Gauge Chart

The following request increases the width of the gauge axis to 50%:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"axisWidth": "50%"}
*END
ENDSTYLE
END
```

On the chart output, the color bands that comprise the axis occupy 50% of the gauge area:



Controlling the Fill Color of the Interior of the Gauge Face

The fill property controls the fill color of the interior of the gauge face.

Syntax: How to Control the Fill Color of the Interior of the Gauge Face

```
"gaugeProperties": {
   "fill": {
      "color": "color"   },
}
```

where:

`"color": "`*`color`*`"`

Can be a JSON gradient definition or a string, enclosed in double quotation marks ("), that specifies a color name or numeric specification string, or a gradient definition.

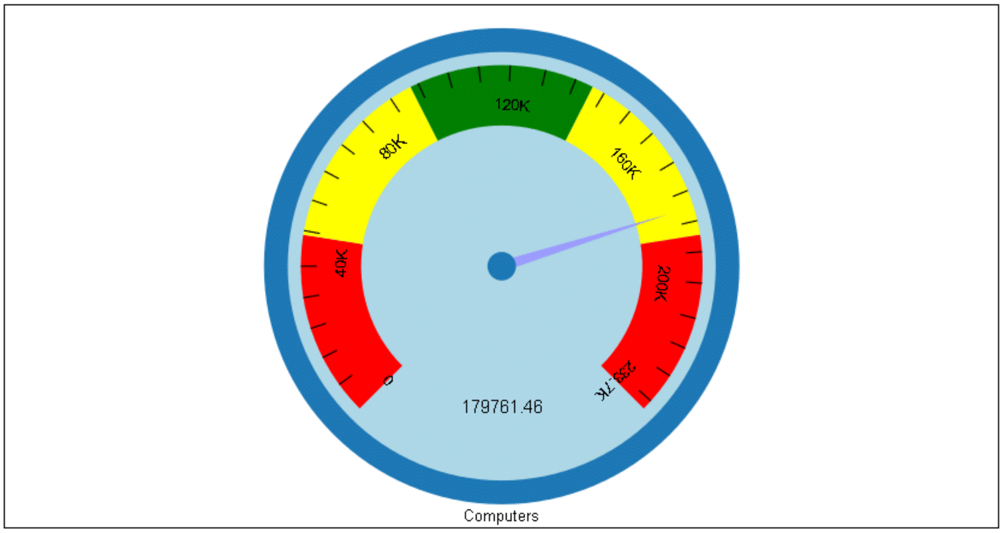
Note: For information about defining colors and gradients, see, *Colors and Gradients* on page 85.

Example: Controlling the Fill Color of the Gauge Face

The following request fills the interior of the gauge face with the color "lightBlue".

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"fill": {"color": "lightBlue"}}
*END
ENDSTYLE
END
```

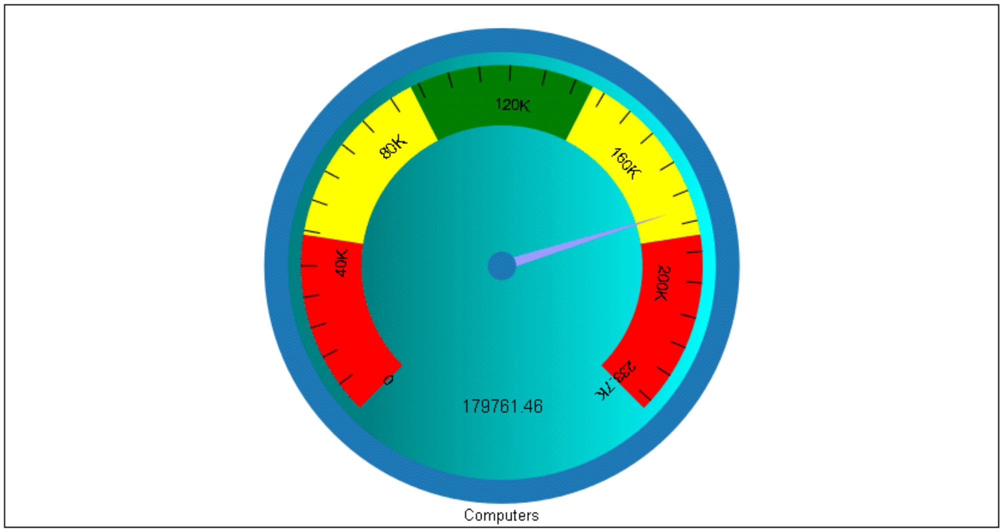
The output is:



The following changes the fill color to a linear gradient that transitions from teal to cyan:

```
"gaugeProperties": {
    "fill": {"color": "linear-gradient(0%,0%,100%,0%, 20% teal, 95% cyan)"}}
```

The output is:



Formatting the Group Label in a Gauge Chart

The groupLabel properties control the visibility and format of the group label in a gauge chart.

Syntax: How to Format the Group Label in a Gauge Chart

```
"gaugeProperties": {
   "groupLabel": {
      "visible": boolean,
      "font": "string",
      "color": "string"
   },
},
```

where:

`"visible": boolean`

Valid values are:

- true, which makes the group label visible. This is the default value.
- false, which makes the group label not visible.

`"font": "string"`

Is a string that defines the size, style, and, typeface of the label. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the label. The default value is "black".

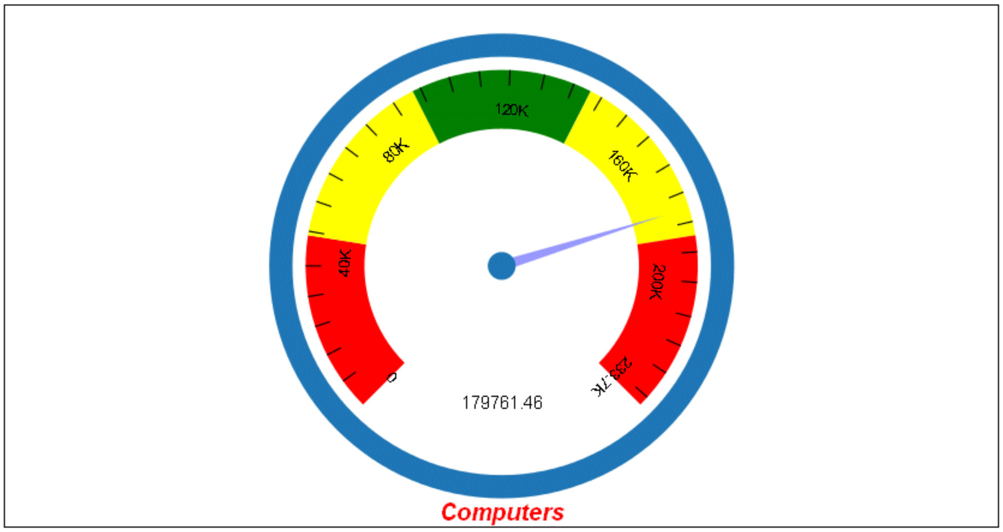
Note: The gaugeProperties groupLabel property does not change the text that displays for the group label. The general groupLabels property does that. The default group label is the sort field value for which the gauge is drawn.

Example: Formatting the Gauge Group Label

The following request makes the group label red and bold-italic 14pt Sans-Serif:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"groupLabel": {
    "font": "bold italic 14pt Sans-Serif", "color": "red"}}
*END
ENDSTYLE
END
```

The output is:



Formatting the Base of the Gauge Needle

The needleBase properties control the format of the base of the gauge needle.

Syntax: How to Format the Base of the Gauge Needle

```
"gaugeProperties": {
   "needleBase": {
      "size": size,
      "color": "color",
      "border": {
         "width": number,
         "color": "string",
         "dash": "string"
      }
   },
},
```

where:

`"size": size`

Can be a number that defines the radius in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that defines the radius as a percentage of the overall gauge. The default value is "6%".

`"color": "`*`color`*`"`

Can be a JSON gradient definition, or a string, enclosed in single quotation marks (''), that defines a color name or numeric specification string, or a gradient defined as a string. For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"border":`

Defines the properties of the needle base border.

`"width":` *`number`*

Is a number of pixels that defines the width of the border around the needle base. The default value is zero (no border).

`"color": "`*`string`*`"`

Is a color defined by a name or numeric specification string. The default value is "transparent" (no border).

`"dash": "`*`string`*`"`

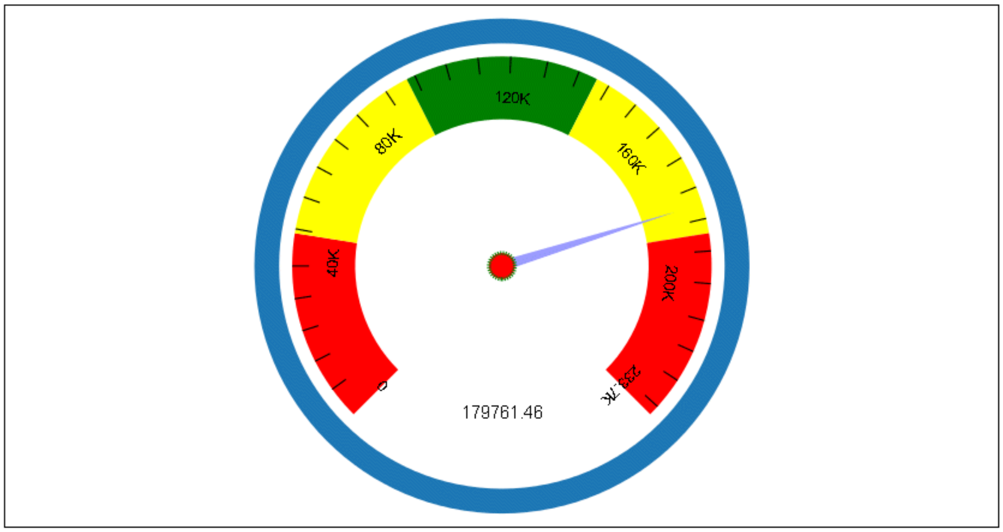
Is a string that defines the border dash style. The default value is `""`, which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Example: Formatting the Gauge Needle Base

The following request makes the gauge needle base red, with a radius of 10 pixels and a green dashed border with a width of 3 pixels:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "needleBase": {
        "size": 10, "color": "red",
        "border": {"width": 3, "color": "green", "dash": "1 1"}
        }
}
*END
ENDSTYLE
END
```

The output is:



Formatting the Outer Border in a Gauge Chart

The outerBorder properties control the format of the outer border in a gauge chart.

Syntax: How to Format the Outer Border in a Gauge Chart

```
"gaugeProperties": {
   "outerBorder": {
      "width": width,
      "shape": "string",
      "fill": {
         "color": "color",
      },
      "border": {
         "width": number,
         "color": "string",
         "dash": "string"
      }
   }
},
```

where:

`"width": width`

Can be a number that defines the width of the outer border ring in pixels, or a string that includes a percent symbol, enclosed in double quotation marks ("), that defines the outer border ring as a percentage of the overall gauge. The default value is "10%".

`"shape": "string"`

Controls the shape of the outer border. Valid values are:

- "circle", which draws the outer border as a circle around the gauge band, needles, and total label. This is the default value.
- "tight", which draws the outer border around the gauge band and needles only (also known as a semi-circle gauge shape).

`"color": "color"`

Defines the fill color of the gauge outer border. Can be a JSON gradient definition or a string, enclosed in single quotation marks ("), that specifies a color name or numeric specification string, or a gradient defined as a string. The default value is "#1f77b4". For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"border":`

Defines the properties of the border around the gauge outer border.

`"width": number`

Is a number of pixels that defines the width of the border of the outer border of the gauge. The default value is zero (no border).

`"color": "string"`

Is a color defined by a name or numeric specification string. The default value is "transparent" (no border).

`"dash": "string"`

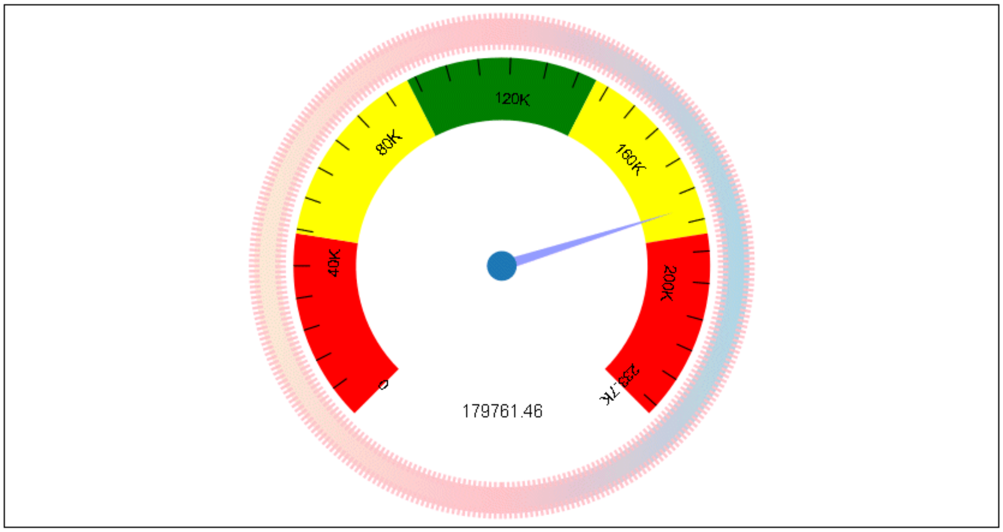
Is a string that defines the border dash style. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Example: Formatting the Outer Border of a Gauge Chart

The following request makes the outer border of the gauge chart 20 pixels wide, with a fill color that is a linear gradient that transitions from antique white to pink to light blue. The border of the outer border is an 8 pixel wide pink dashed line:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "outerBorder": {
        "width": 20, "fill": {
            "color": "linear-gradient(0%,0,100%,0, 0 antiquewhite, 0.5 pink,1
lightblue)"},
        "border": {"width": 8, "color": "pink", "dash": "2 2"}}}
*END
ENDSTYLE
END
```

The output is:

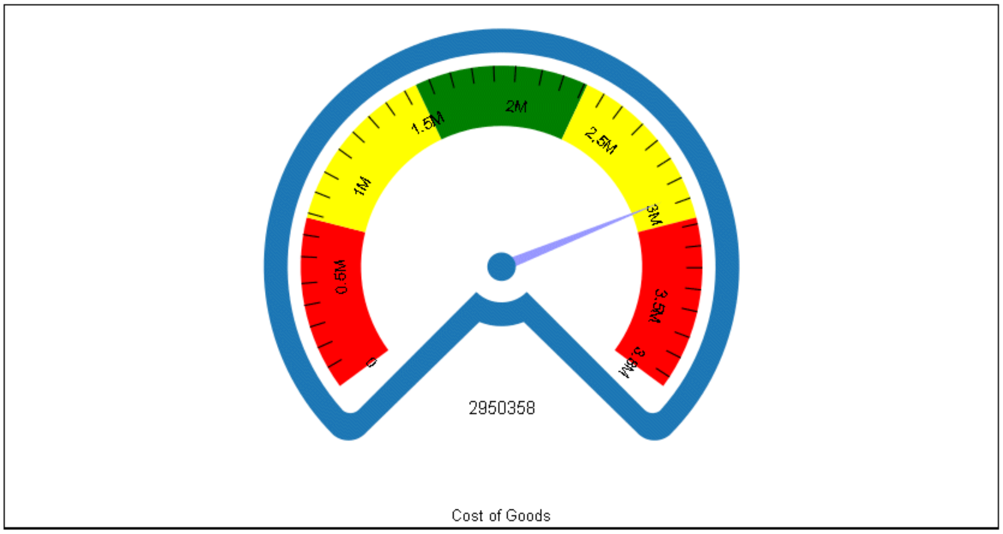


Example: Drawing a Tight Gauge Outer Border

The following request sets the shape of the gauge outer border to "tight":

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"outerBorder": {"shape": "tight"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Drawing Secondary Gauge Needles as Markers

When a gauge includes multiple needles (multiple measures), use the secondaryNeedlesAsMarkers property to draw secondary needles as markers.

Syntax: How to Draw Secondary Gauge Needles as Markers

```
"gaugeProperties": {
   "secondaryNeedlesAsMarkers": boolean,
},
```

where:

`"secondaryNeedlesAsMarkers": boolean`

Valid values are:

- true, which draws the secondary needles as markers.
- false, which draws multiple needles. The default value is false.

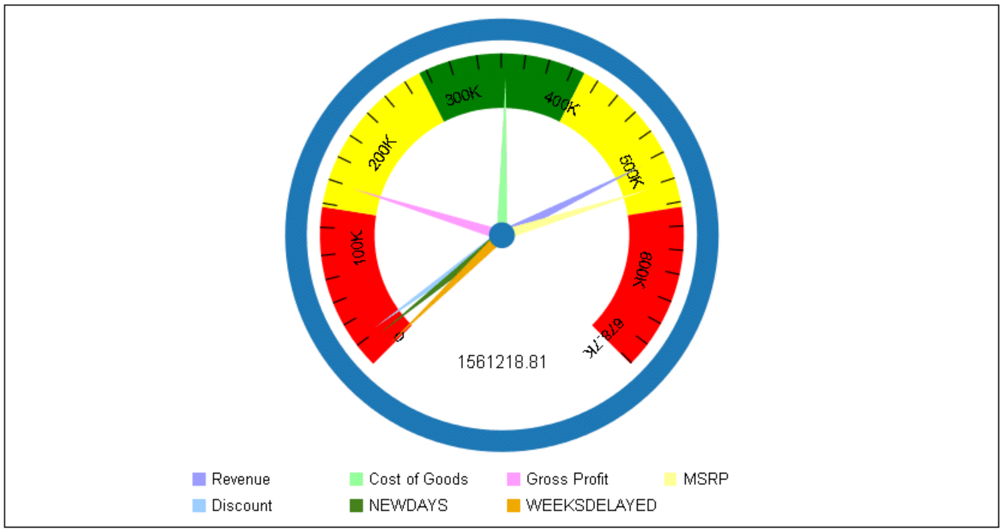
Note: When secondaryNeedlesAsMarkers is true, use the series:*marker* property to control the size and format of the markers.

Example: Drawing Secondary Gauge Chart Needles as Markers

The following request draws the secondary needles as needles:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US COGS_US GROSS_PROFIT_US MSRP_US
       DISCOUNT_US AND COMPUTE
NEWDAYS = DAYSDELAYED *20;
WEEKSDELAYED = NEWDAYS/7;
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "secondaryNeedlesAsMarkers": false}
*END
ENDSTYLE
END
```

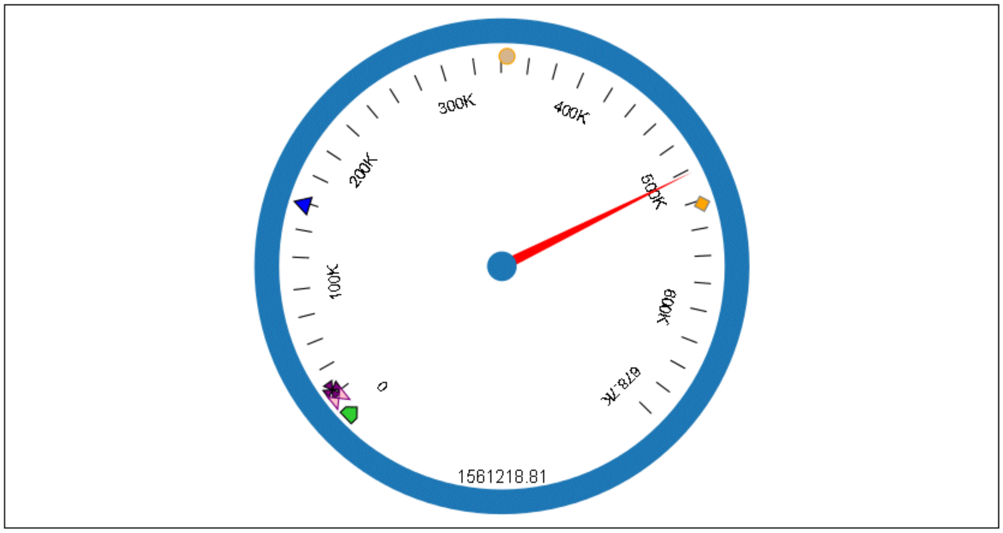
The output is:



The following version of the request, draws the secondary needles as markers, and sets marker shapes, sizes, borders, and colors for them. To make the markers more visible, the axis width is set to zero. Notice that series 0 is the primary needle, and is still drawn as a needle:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US COGS_US GROSS_PROFIT_US MSRP_US
       DISCOUNT_US AND COMPUTE
NEWDAYS = DAYSDELAYED *20;
WEEKSDELAYED = NEWDAYS/7;
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Accessories'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "axisWidth": 0,
    "secondaryNeedlesAsMarkers": true},
"series": [
    {"series": 0, "color": "red"},
    {"series": 1, "color": "tan", "marker": {
        "shape": "circle", "size": 12, "border": {"width": 1, "color":
"orange"}}},
    {"series": 2,"color": "blue", "marker": {
        "shape": "triangle", "size": 12, "border": {"width": 1, "color":
"black"}}},
    {"series": 3, "color": "orange", "marker": {
        "shape": "diamond", "size": 12, "border": {"width": 1, "color":
"grey"}}},
    {"series": 4, "color": "purple", "marker": {
        "shape": "pirateCross", "size": 12, "border": {"width": 1, "color":
"black"}}},
    {"series": 5, "color": "pink", "marker": {
        "shape": "hourglass", "size": 12, "border": {"width": 1, "color":
"purple"}}},
    {"series": 6, "color": "limegreen", "marker": {
        "shape": "house", "size": 12, "border": {"width": 1, "color":
"black"}}}
    ]

*END
ENDSTYLE
END
```

The output is:



Controlling the Start and End Angles of a Gauge Chart Axis

The startAngle and endAngle properties control the start and end angles of the gauge value band. Angles are defined in degrees. Positive angles draw clockwise, negative angles draw counter clockwise. Zero will start or end the band at the 3 o'clock position. 90 will start or end the band at the 6 o'clock position. -90 specifies the 12 o'clock position.

Note: The gauge chart axis is also called a *value band*.

Syntax: How to Control the Start and End Angles of a Gauge Chart Axis

```
"gaugeProperties": {
   "startAngle": number,
   "endAngle": number},
```

where:

`"startAngle": number`

Is a number that defines the angle (in degrees) at which to start the gauge value band. The default value is 135.

`"endAngle": number`

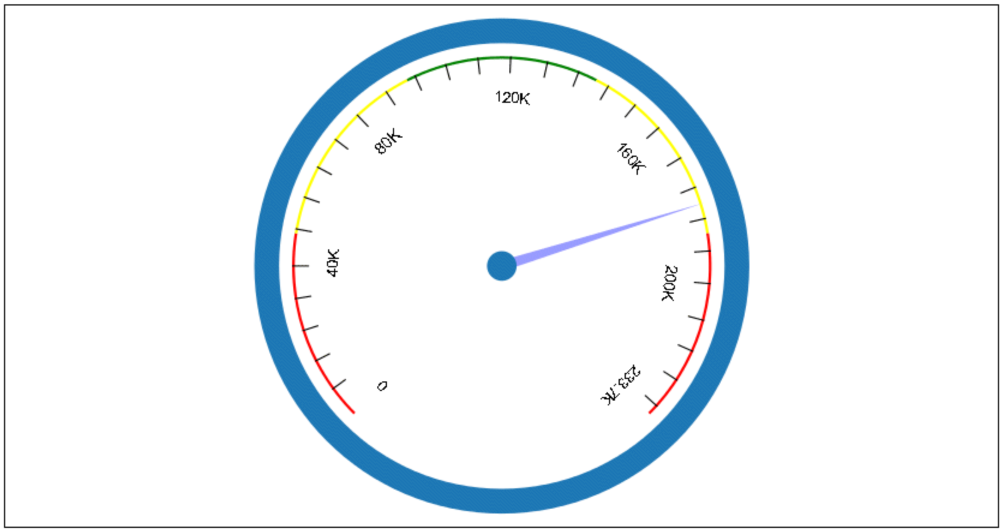
Is a number that defines the angle (in degrees) at which to end the gauge value band. The default value is 45.

Example: Controlling the Start and End Angles in the Gauge Chart Axis

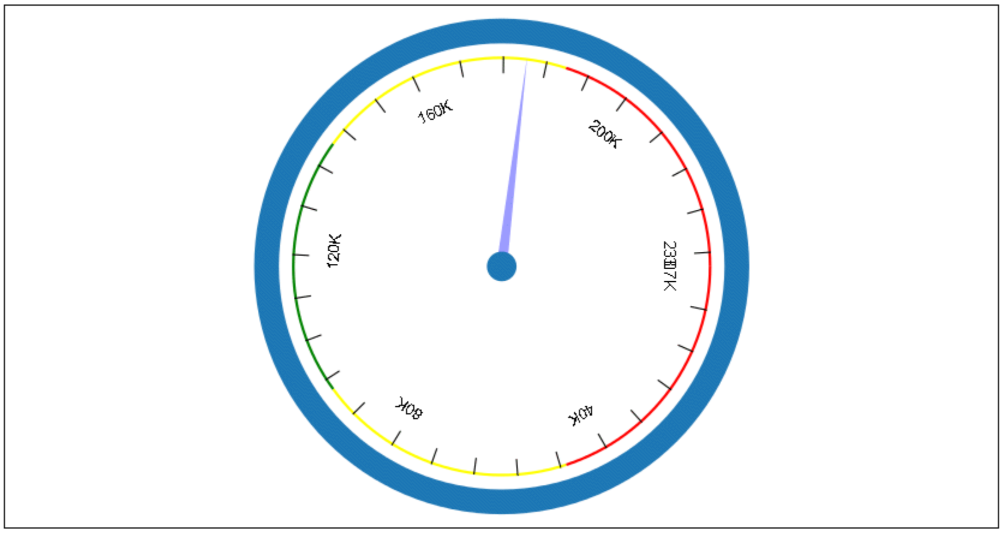
The following request uses the default start angle (135) and end angle (45) for the gauge chart value band. The axis width is set to 2 pixels:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false}, "totalLabel": {"visible": false},
    "axisWidth": 2, "startAngle": 135, "endAngle": 45}
*END
ENDSTYLE
END
```

The output is:



Changing the start and end angles to zero (0), makes the value band a complete circle:



Controlling the Secondary Ring Fill Color in a Simple Gauge Chart

When the gauge chart layout is "simple", you can define a color to fill the portion beyond the start and end angles.

Syntax: How to Control the Secondary Ring Fill Color in a Simple Gauge Chart

```
"gaugeProperties": {
   "layout": "simple",
   "secondaryRingFillColor": "string"
               }
```

where:

`"secondaryRingFillColor": "string"`

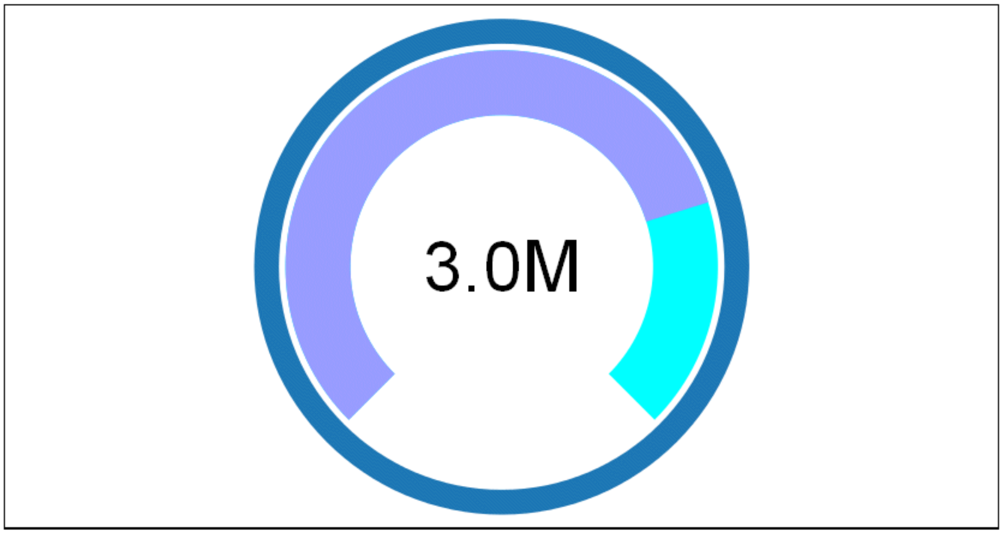
Is a color definition for filling the portion of the gauge band outside the start and end angles. The default value is *undefined*. This property has no effect in a traditional gauge layout.

Example: Controlling the Secondary Ring Fill Color in a Simple Gauge Chart

The following request sets the portion of the ring outside of the default start and end angles to cyan:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {"layout": "simple",
    "secondaryRingFillColor": "cyan"}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Formatting a Gauge Chart Total Label

The totalLabel properties control the visibility and format of the total label in a gauge chart.

Syntax: How to Format a Gauge Chart Total Label

```
"gaugeProperties": {
   "totalLabel": {
      "visible": boolean,
      "font": "string",
      "color": "string",
      "numberFormat": numformat   },
},
```

where:

`visible: boolean`

Valid values are:

- true, which makes the label visible.
- false, which makes the label not visible. This is the default value.

`"font": "string"`

Is a string that defines the size, style, and, typeface of the label. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the label using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"numberFormat": numformat`

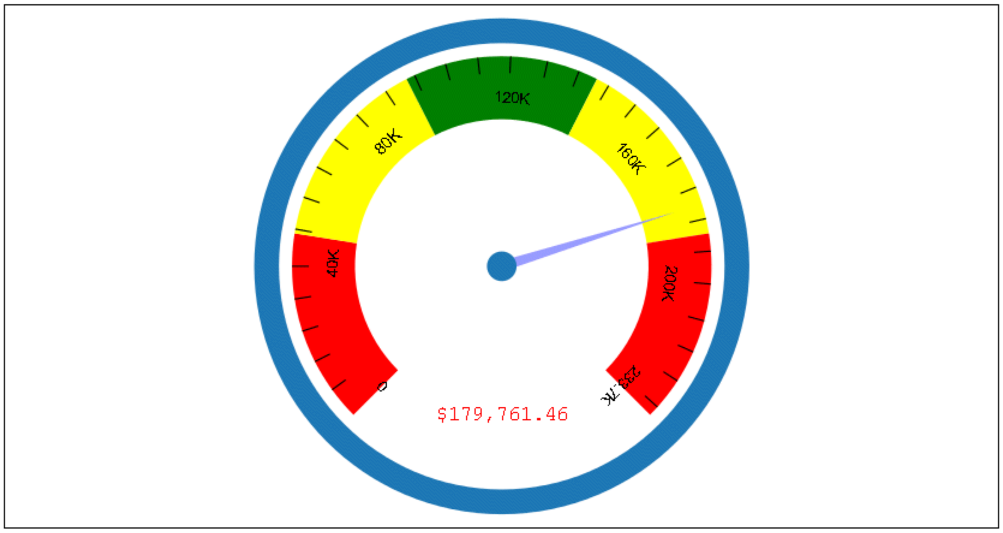
Can be specified as a JSON object, a format string, or a user-defined function. The default value is "auto". For information on specifying number formats, see *Formatting Numbers* on page 108.

Example: Formatting a Gauge Chart Total Label

The following request makes the gauge chart total label visible, with a red 12pt Courier font, and a currency number format:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Computers'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET STYLE *
*GRAPH_JS
"gaugeProperties": {
    "groupLabel": {"visible": false},
    "totalLabel": {"visible": true, "font": "12pt Courier", "color": "red",
        "numberFormat": {"mode": "currency"}}}
*END
ENDSTYLE
END
```

The output is:



Histogram Chart Properties (histogramProperties)

A histogram is a graphical representation of the frequency distribution of data. The histogramProperties object controls the properties associated with the distribution of data.

Syntax: How to Control Histogram Chart Properties

```
"histogramProperties": {
   "binCount": number,
   "binSize": size,
   "startBinValue": number}
```

where:

`"binCount": number`

Is the number of group labels to draw, or undefined (automatic). The default value is undefined.

`"binSize": size`

Is the size of the value range for assigning data values to a bin. Can be a number, an array of numbers (for variable bin sizes), or undefined (automatic). If binSize is an array, binCount is ignored and assumed to be the length of the binSize array. The default value is undefined.

`"startBinValue": number`

Is a number to use as the first bin, or undefined (automatic). The default value is undefined.

Note:

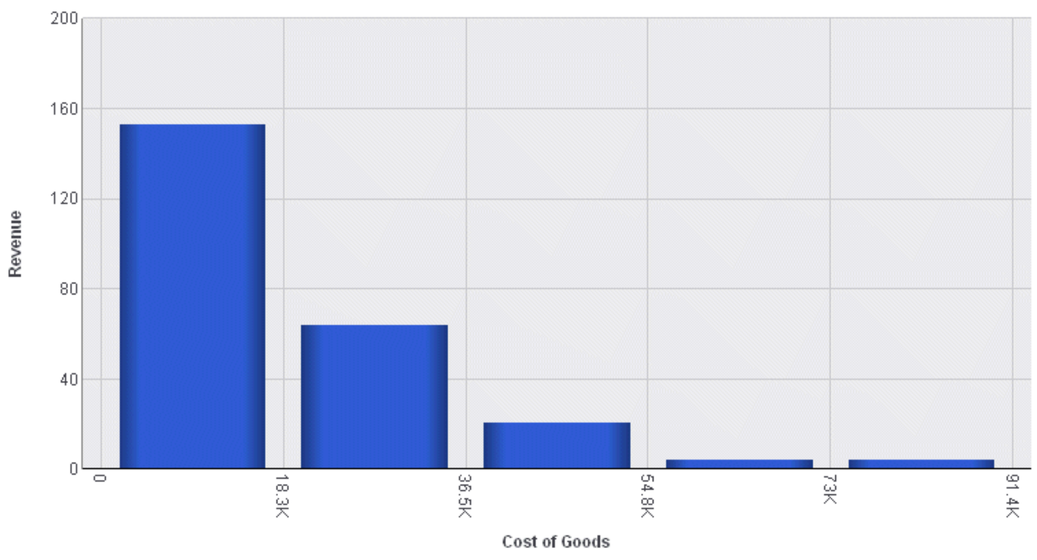
- The blaProperties orientation property draws the chart in horizontal or vertical format. However, instead of using this property, you can control the orientation of the chart using the LOOKGRAPH parameter, which has a value for each histogram orientation. For information about the LOOKGRAPH parameter, see *Controlling the Chart Type* on page 40.
- In a vertical histogram, the y-axis is on the left side of the chart and the x-axis is drawn on the bottom of the chart.
- In a horizontal histogram, the x-axis is on the left side of the chart and the y-axis axis is drawn on the bottom of the chart.
- Use blaProperties:barGroupGapWidth to control the gap between histogram risers.
- The series-specific properties control the colors and borders (if any) of histogram risers.
- Use yaxis properties to control the format of y-axis title and labels.
- Use xaxis properties to control the format of x-axis title and labels. X-axis labels are calculated. They are not defined by the groupLabels property.

Example: Setting the Histogram Bin Count

The following request creates a vertical histogram (ON GRAPH SET LOOKGRAPH VHISTOGR) with a bin count of 5:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY COGS_US
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VHISTOGR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"histogramProperties": {"binCount": 5},
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is:



Matrix Chart Properties (matrixProperties)

The matrix chart properties define formats for the matrix labels, headers, and lines and the types of charts in each cell. Matrix charts require the use of WebFOCUS chart attribute syntax, described in *WebFOCUS Chart Attribute Syntax* on page 143.

On a matrix column bar, line, or area chart, the x-axis title is concatenated to the end of the matrix column title instead of being repeated on each individual x-axis.

The following lists the matrix chart properties and their default values:

```
"matrixProperties": {
    "chartType": 'bar',
    "rowLabels": {
        "labels": undefined,
        "font": "8pt Sans-Serif",
        "color": "black",
        "tooltip": undefined
               },
    "colLabels": {
      "labels": undefined,
      "font": "8pt Sans-Serif",
      "color": "black",
      "tooltip": undefined
         },
    "rowHeader": {
      "text": undefined,
      "font": "10pt Sans-Serif",
      "color": "black",
      "tooltip": undefined
         },
    "colHeader": {
      "text": undefined,
      "font": "10pt Sans-Serif",
      "color": "black",
      "tooltip": undefined
         },
    "cellBorder": {
      "width": 1,
      "color": "grey",
      "dash": ""
         },
    "layout":
        {"cellPadding":
             {
             "top": 0,
             "bottom": 0,
             "left": 0,
             "right": 0
             }
          }
    "minCellSize": "auto",
   "yaxis": {
     "min": "autoPerRow",
     "max": "autoPerRow"
        }
}
```

Controlling the Chart Type in the Cells of the Matrix

The chartType property controls the type of chart that will be displayed in the matrix cells. You must be sure that you have the correct sort fields and measures, and the corresponding attribute categories for the type of chart you want to display in the matrix cells.

For example, bar, line, and area charts support the same attribute categories, so you can display them interchangeably by changing the chartType property value. You can even use the WebFOCUS Amper Autoprompting feature to present the user with a drop-down list of chart types to choose from.

Syntax: How to Control the Chart Type Displayed in a Matrix Chart

```
"matrixProperties": {
    "chartType": "string"},
```

where:

`"chartType": "string"`

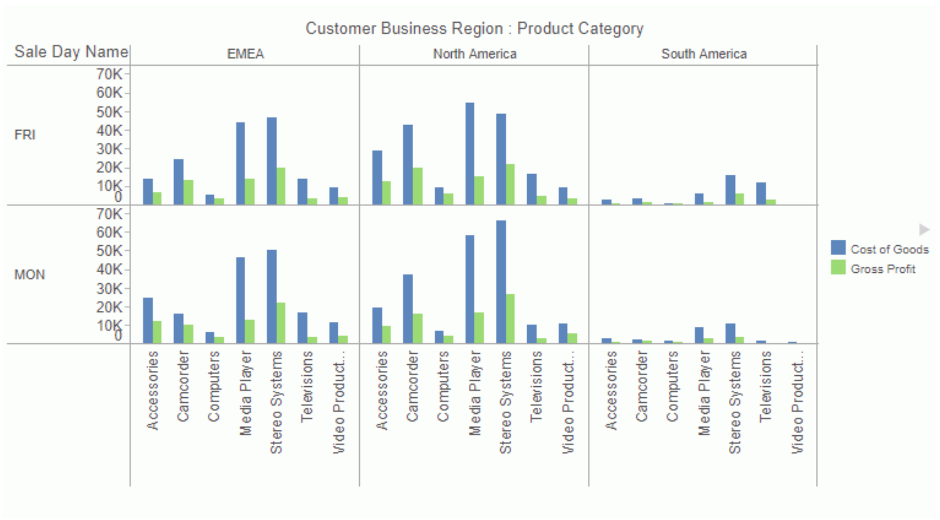
Is a supported chart type for the attribute categories in the request. The default value is "bar".

Example: Controlling the Chart Type Displayed in a Matrix Chart

The following request uses a Dialogue Manager amper variable in the definition of the chartType property. The -SET command sets its value to "bar":

```
-SET &TYPE=bar;
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"chartType": "&TYPE"}
*END
ENDSTYLE
END
```

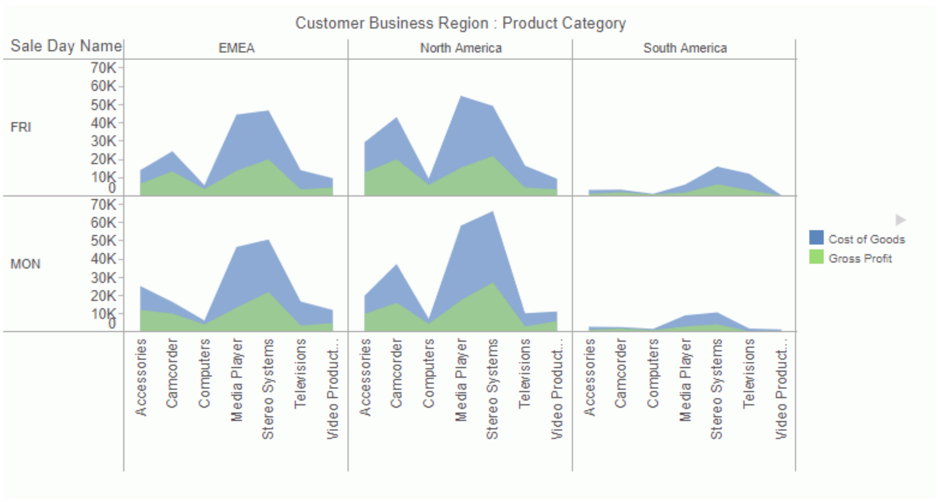
The output is shown in the following image:



Changing the -SET command to the following, generates an area chart:

```
-SET &TYPE=area;
```

The output is shown in the following image:



Formatting the Matrix Row Labels

The matrix row labels are the labels displayed at the left of each row.

Syntax: How to Format the Matrix Row Labels

```
"matrixProperties": {
       "rowLabels": {
       "labels": undefined,
       "font": "string",
       "color": "string",
       "tooltip": "string"
}
```

where:

`"font": "string"`

Is a font string for the row label text. The default value is '8pt Sans-Serif'.

`"color": "string"`

Is a color string for the row labels. The default value is 'black'.

`"tooltip": "string"`

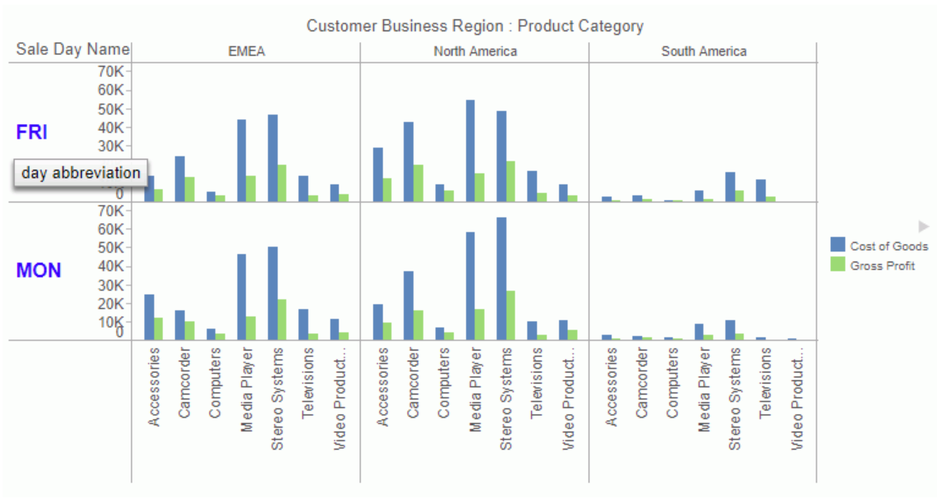
Is a tooltip to display when the mouse hovers over a row label.

Example: Formatting the Matrix Row Labels

The following request formats the row labels in blue, bold, 12pt Sans Serif font, with the tooltip *day abbreviation*:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"rowLabels": {
    "font": "bold 12pt Sans-Serif",
    "color": "blue",
    "tooltip": "day abbreviation"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Formatting the Matrix Column Labels

The matrix column labels are the labels displayed at the top of each column.

Syntax: How to Format the Matrix Column Labels

```
"matrixProperties": {
       "colLabels": {
       "labels": undefined,
       "font": "string",
       "color": "string",
       "tooltip": "string"
}
```

where:

`"font": "string"`

Is a font string for the column label text. The default value is "8pt Sans-Serif".

`"color": "string"`

Is a color string for the column labels. The default value is "black".

`"tooltip": "string"`

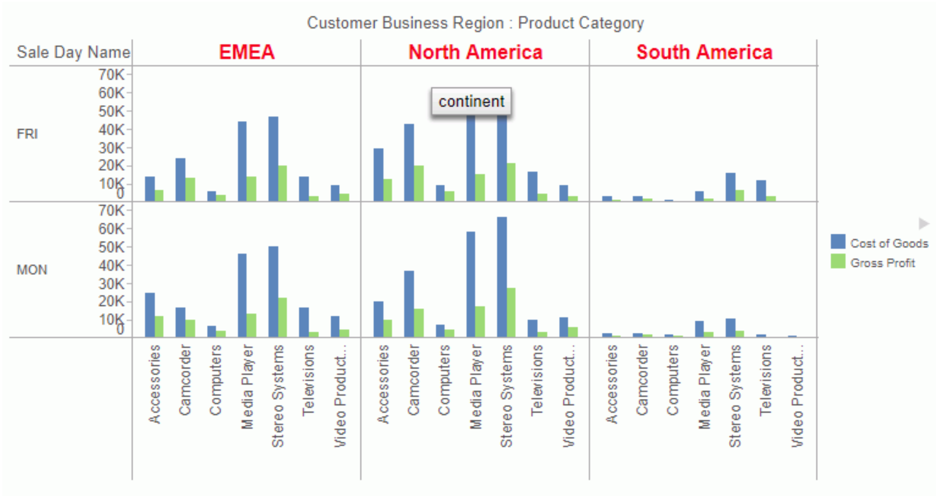
Is a tooltip to display when the mouse hovers over a column label.

Example: Formatting the Matrix Column Labels

The following request formats the column labels in red, bold, 12pt Sans Serif font, with the tooltip *continent*:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"colLabels": {
    "font": "bold 12pt Sans-Serif",
    "color": "red",
    "tooltip": "continent"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Formatting the Matrix Row Header

The matrix row header is the single label drawn at the top of the rows.

Syntax: How to Format the Matrix Row Header

```
"matrixProperties": {
       "rowHeader": {
       "text": undefined,
       "font": "string",
       "color": "string",
       "tooltip": "string"
}
```

where:

`"font": "string"`

Is a font string for the row header text. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a color definition string for the row header. The default value is "black".

`"tooltip": "string"`

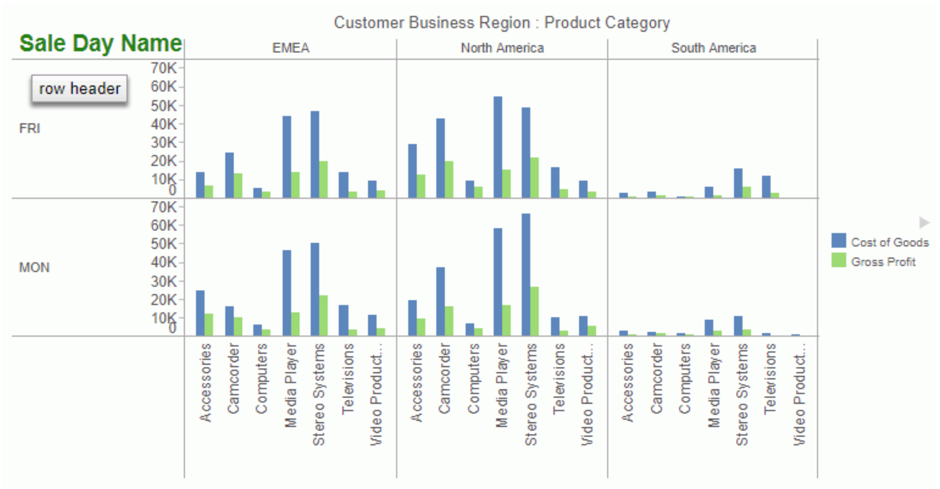
Is a tooltip to display when the mouse hovers over the row header.

Example: Formatting the Matrix Row Header

The following request formats the row header in green, bold, 14pt Sans Serif font, with the tooltip *row header*:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"rowHeader": {
    "font": "bold 14pt Sans-Serif",
    "color": "green",
    "tooltip": "row header"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Formatting the Matrix Column Header

The matrix column header is the single label drawn at the top of the columns.

Syntax: How to Format the Matrix Column Header

```
"matrixProperties": {
       "colHeader": {
       "text": undefined,
       "font": "string",
       "color": "string",
       "tooltip": "string"
}
```

where:

`"font": "string"`

Is a font string for the column header text. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a color definition string for the column header. The default value is "black".

`"tooltip": "string"`

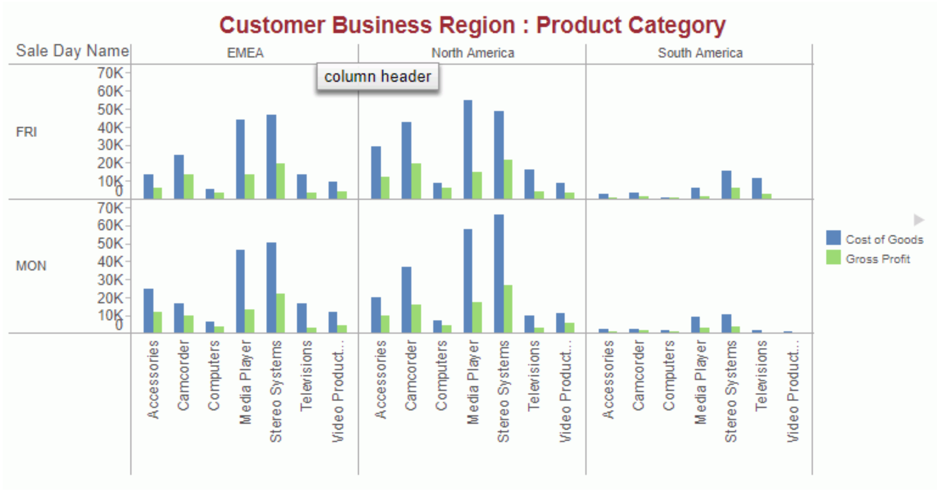
Is a tooltip to display when the mouse hovers over the column header.

Example: Formatting the Matrix Column Header

The following request formats the column header in brown, bold, 14pt Sans Serif font, with the tooltip *column header*:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"colHeader": {
    "font": "bold 14pt Sans-Serif",
    "color": "brown",
    "tooltip": "column header"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Formatting the Matrix Cell Borders

The cell borders are the lines drawn between the cells of the matrix.

Syntax: How to Formatting the Matrix Cell Borders

```
"matrixProperties": {
       "cellBorder": {
       "width": number,
       "color": "string",
       "dash": "string"
}
```

where:

`"width": number`

Is the width of the cell borders in pixels. The default value is 1.

`"color": "string"`

Is a color definition string for the cell borders. The default value is "grey".

`"dash": "string"`

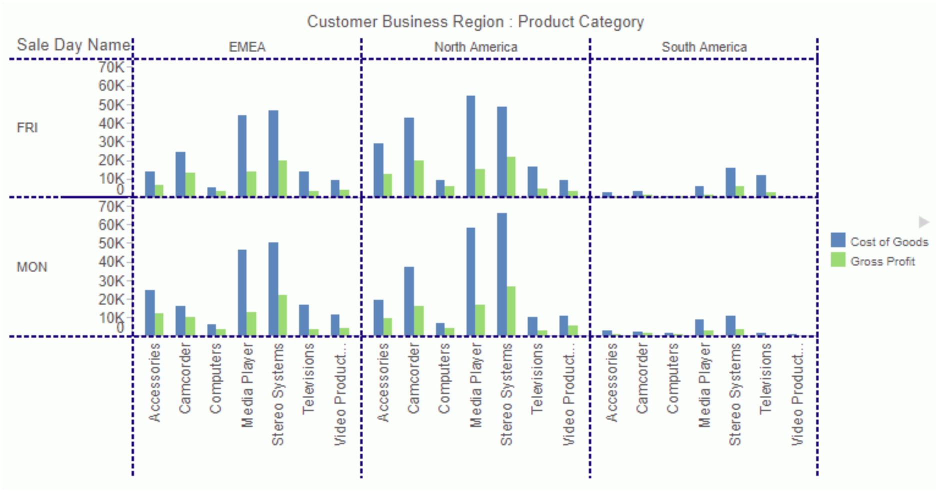
Is the dash style for the cell borders. Specify a dash width in pixels followed by the number of pixels between dashes. The default is no dash (““).

Example: Formatting the Matrix Cell Borders

The following request formats the cell borders to be navy with a width of 2 pixels and a dash style in which the length of each dash is 4 pixels and the space between dashes is 2 pixels:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {"cellBorder": {
    "width": 2,
    "color": "navy",
    "dash": "4 2"}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Controlling Cell Padding in Matrix Cells

You can set a number of pixels of blank space (padding) to add to the top, bottom, left, and right margins of matrix cells using the layout: cellpadding properties. This technique is especially useful when the divider lines are turned off.

Reference: Control Cell Padding

```
matrixProperties":
{
 "layout":
          {"cellPadding":
             {
             "top": number,
             "bottom": number,
             "left": number,
             "right": number
             }
          }
}
```

where:

`"top": number`
: Defines the number of pixels of padding on the top of matrix cells. The default value is zero (0).

`"bottom": number`
: Defines the number of pixels of padding on the bottom of matrix cells. The default value is zero (0).

`"left": number`
: Defines the number of pixels of padding on the left of matrix cells. The default value is zero (0).

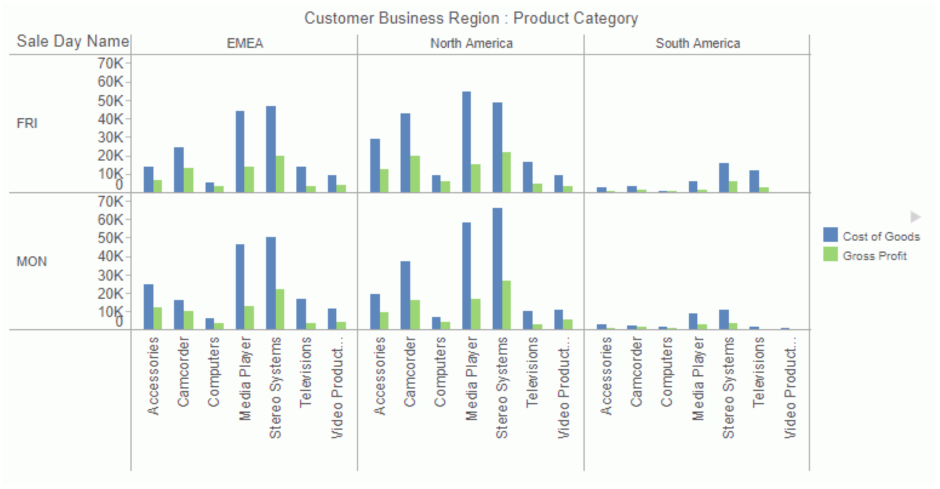
`"right": number`
: Defines the number of pixels of padding on the right of matrix cells. The default value is zero (0).

Example: Padding Matrix Cells

The following request creates a matrix chart with the default padding.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
ENDSTYLE
END
```

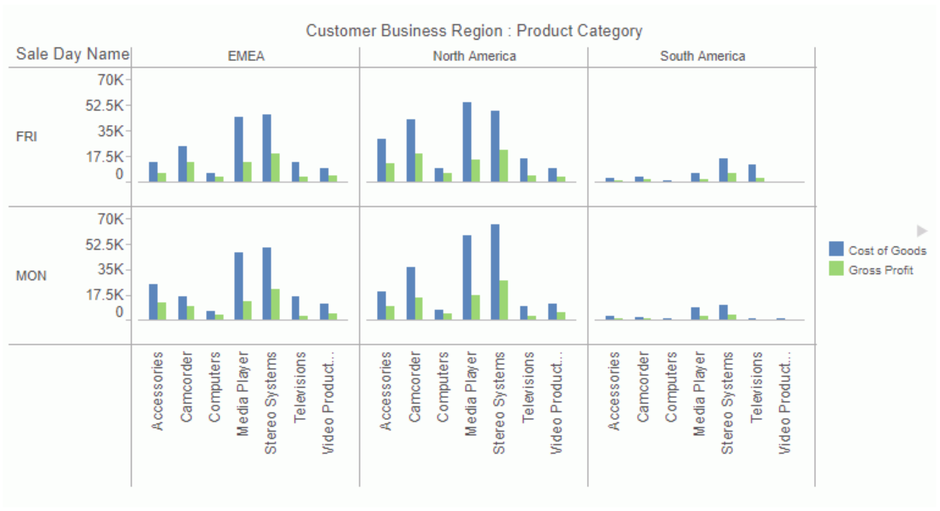
The output is shown in the following image.



The following version of the request adds 10 pixels to the top of the cells, 20 pixels to the bottom, 5 pixels to the left, and 10 pixels to the right.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties":
 {
    "chartType": "bar",
    "layout": {
          "cellPadding":
            {
             "top": 10,
             "bottom": 20,
             "left": 5,
             "right": 10
            }
           }
 }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Controlling the Matrix Chart Minimum Cell Size

You can specify a minimum width and height for the cells in a matrix chart, or use 'auto' to have the chart engine make the cells as small as possible while still being readable.

Syntax: How to Control the Minimum Cell Size in a Matrix Chart

```
matrixProperties: {
       "minCellSize": {"width": number, "height": number}}
```

where:

`"minCellSize"`

Defines the minimum cell size. If the cell size multiplied by data size is bigger than the chart area, scrolling will be activated. If it is smaller than the chart area, the cells will grow to fill it. The value can be "auto" to let the chart engine determine the minimum size.

`"width": number`

Is the minimum width in pixels.

`"height": number`

Is the minimum height in pixels.

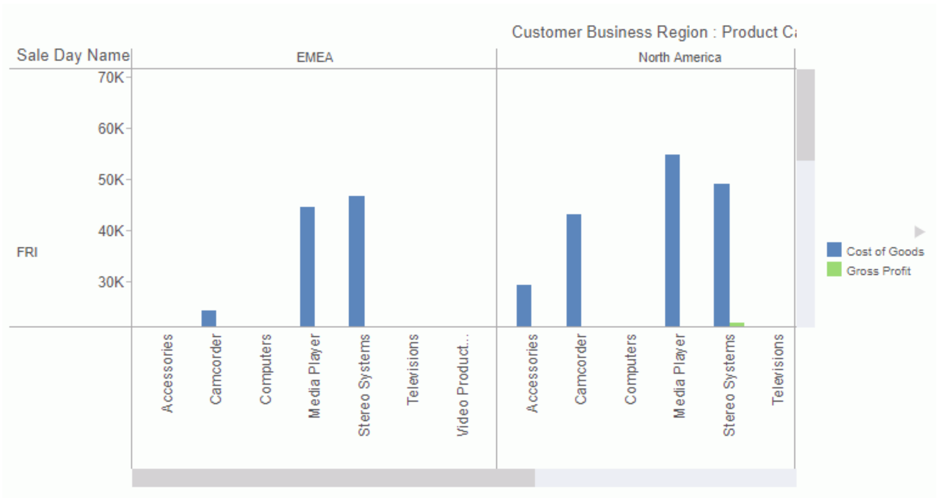
Note: Use {"width": 0, "height": 0} to evenly split the available space.

Example: Setting the Minimum Cell Size in a Matrix Chart

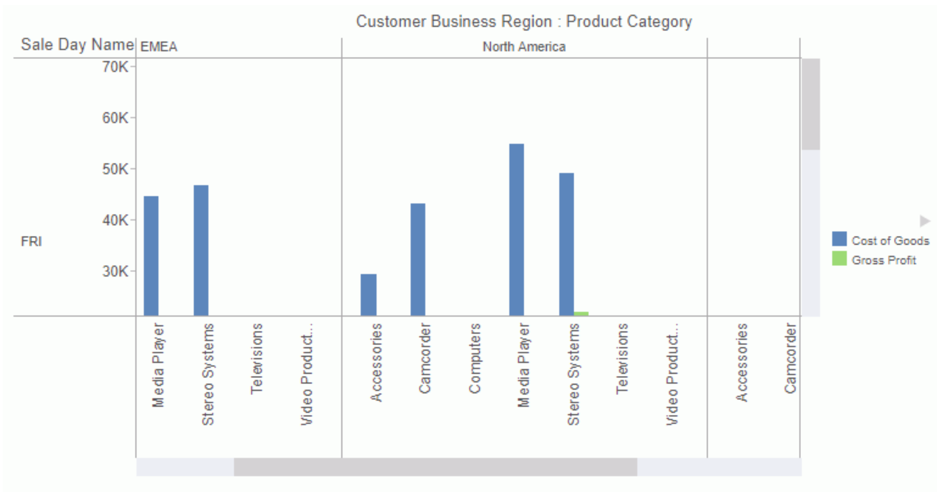
The following request defines the minimum cell size to be {width:300, height:300}, which activates scrolling:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DAYNAME
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
WHERE TIME_DAYNAME EQ 'FRI' OR 'MON'
WHERE BUSINESS_REGION NE 'Oceania'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=TIME_DAYNAME, BUCKET=row, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"matrixProperties": {
    "minCellSize": {"width": 300, "height": 300}}
*END
ENDSTYLE
END
```

The unscrolled output is shown in the following image:



The following image shows the output with the scroll bars dragged to show the complete chart for the cell representing column *North America* and row MON:



Setting the Numeric Axis Scope in Matrix Charts

Using the axis min and max properties, you can set the scope of values on a numeric axis in a matrix chart separately for each matrix row or column. The scope can be set to specific minimum and maximum values or be automatically calculated separately for each row or column.

Syntax: How to Set the Numeric Axis Scope in Matrix Charts

On a vertical chart, the y-axis scope is applied to each matrix row, and the x-axis scope is applied to each matrix column. On a horizontal chart the y-axis scope is applied to each matrix column, and the x-axis scope is applied to each matrix row.

For automatically calculated axis values, use the following properties.

```
{
 "axisname": {
        "min": "autoPerRow",
        "max": "autoPerRow"
        }
}
```

where:

"*axisname*"

Is the name of the numeric axis, either "yaxis" or "xaxis".

`"min": "autoPerRow"`

Calculates a unique automatic minimum axis value for each row or column.

`"max": "autoPerRow"`

Calculates a unique automatic maximum axis value for each row or column.

For specifically set axis values, use the following properties.

```
{
 "axisname": {
        "min": [number, number, number, ...],
        "max": [number, number, number, ...]
        }
}
```

where:

`"axisname"`

Is the name of the numeric axis, either "yaxis" or "xaxis".

`"min": [number, number, number, ...]`

Sets a minimum value for each axis in the matrix chart. Assuming the numeric axis is the y-axis, the first element in the array is the minimum y-axis value in row 1 of the matrix, the second element is the minimum y-axis value for row 2 in the matrix, and so on. If an element is null (the array has two consecutive commas, starts with a comma, or has fewer elements than the matrix has rows or columns), the chart engine provides a value for those axes. If the array has more entries than the number of rows in the matrix, the extra numbers are ignored.

`"max": [number, number, number, ...]`

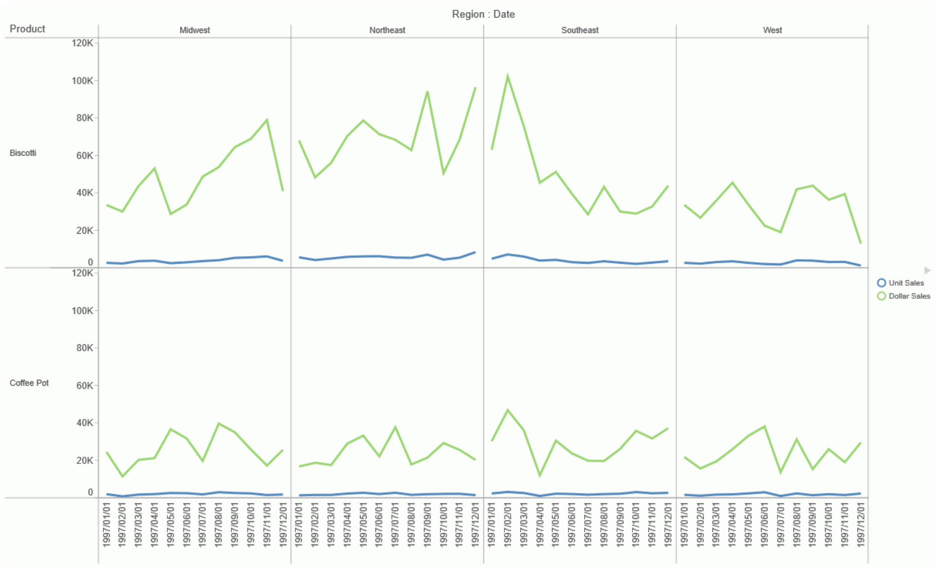
Sets a maximum value for each axis in the matrix chart. Assuming the numeric axis is the y-axis, the first element in the array is the maximum y-axis value in row 1 of the matrix, the second element is the maximum y-axis value for row 2 in the matrix, and so on. If an element is null (the array has two consecutive commas, starts with a comma, or has fewer elements than the matrix has rows or columns), the chart engine provides a value for those y-axes. If the array has more entries than the number of rows in the matrix, the extra numbers are ignored.

Example: Setting the Minimum and Maximum Y-Axis Values in a Matrix Chart

The following request generates two matrix rows and uses the default y-axis scope.

```
GRAPH FILE GGSALES
SUM UNITS DOLLARS
BY PRODUCT
BY REGION
BY DATE
WHERE PRODUCT EQ 'Biscotti' OR 'Coffee Pot'
WHERE DATE GE '19970101'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=PRODUCT, BUCKET=row, $
TYPE=DATA, COLUMN=REGION, BUCKET=column, $
TYPE=DATA, COLUMN=UNITS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DOLLARS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DATE, BUCKET=x-axis, $
ENDSTYLE
END
```

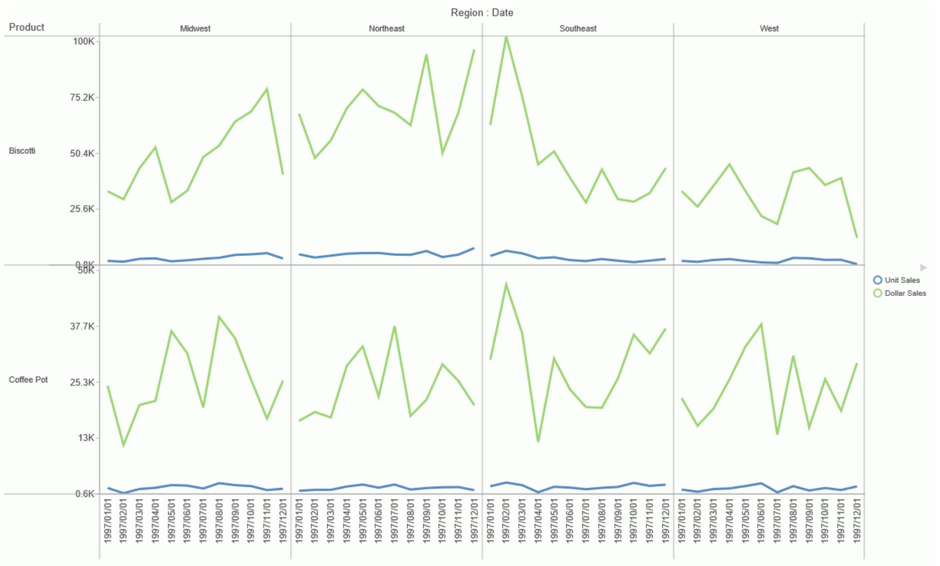
The output is shown in the following image. Both y-axes have the scope zero to 120,000.



The following version of the request generates two matrix rows and sets a minimum and maximum value for the y-axis in each row.

```
GRAPH FILE GGSALES
SUM UNITS DOLLARS
BY PRODUCT
BY REGION
BY DATE
WHERE PRODUCT EQ 'Biscotti' OR 'Coffee Pot'
WHERE DATE GE '19970101'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=PRODUCT, BUCKET=row, $
TYPE=DATA, COLUMN=REGION, BUCKET=column, $
TYPE=DATA, COLUMN=UNITS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DOLLARS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DATE, BUCKET=x-axis, $
*GRAPH_JS
"yaxis":
{
"min": [800, 600],
"max": [100000, 50000]
}
ENDSTYLE
END
```

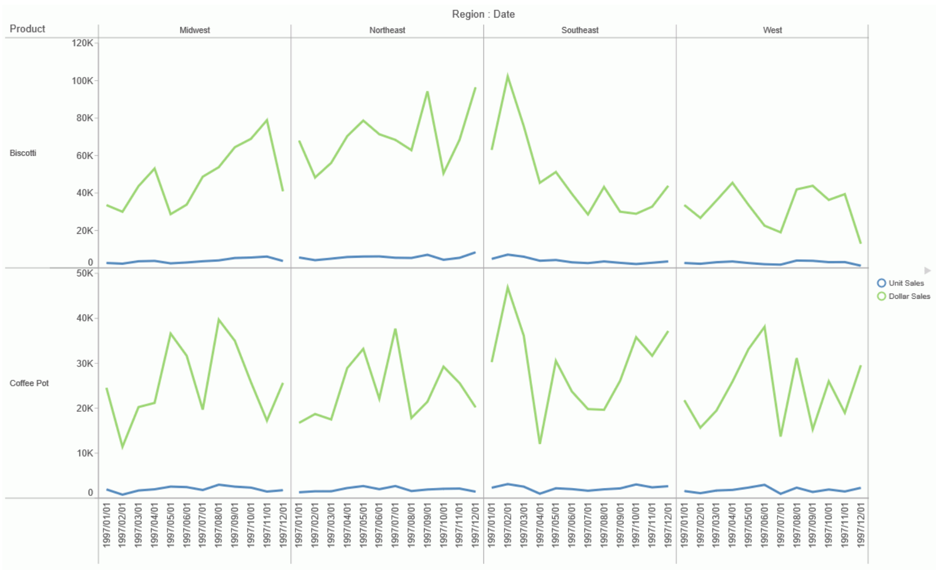
The output is shown in the following image. The y-axis scope for the first row is 800 to 100,000, and for the second row is 600 to 50,000.



The following version of the request generates two matrix rows and uses "autoPerRow" for the y-axis in each row.

```
GRAPH FILE GGSALES
SUM UNITS DOLLARS
BY PRODUCT
BY REGION
BY DATE
WHERE PRODUCT EQ 'Biscotti' OR 'Coffee Pot'
WHERE DATE GE '19970101'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=PRODUCT, BUCKET=row, $
TYPE=DATA, COLUMN=REGION, BUCKET=column, $
TYPE=DATA, COLUMN=UNITS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DOLLARS, BUCKET=y-axis, $
TYPE=DATA, COLUMN=DATE, BUCKET=x-axis, $
*GRAPH_JS
"yaxis":
{
"min": "autoPerRow",
"max": "autoPerRow"
}
ENDSTYLE
END
```

The output is shown in the following image. The chart engine calculates a unique scope for each y-axis, depending on the chart data for that row. For row 1, the y-axis scope is zero to 120,000. For row 2, the y-axis scope is zero to 50,000.



Parabox Chart Properties (paraboxProperties)

This property defines the active group in a parabox chart.

Syntax: How to Control Parabox Properties

```
"paraboxProperties": {
  "activeGroup": "string"
},
```

where:

`"activeGroup": "string"`

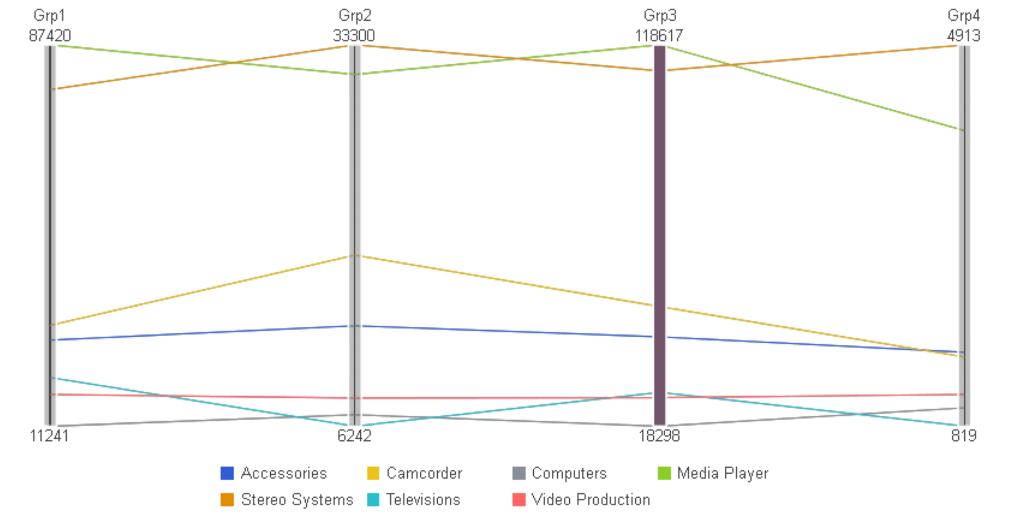
Is a string that defines the group (specified by its label) that is active. The active group defines coloring for the lines that come through it. The default value is undefined.

Example: Setting the Active Group in a Parabox

The following request makes Grp3 the active group:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US AS 'Cost'
GROSS_PROFIT_US AS 'Profit'
REVENUE_US DISCOUNT_US MSRP_US
AVE.COGS_US AS 'Ave. Cost'
AVE.GROSS_PROFIT_US AS 'Ave. Profit'
MAX.REVENUE_US
MIN.DISCOUNT_US
MDN.MSRP_US AS 'MDN MSRP'
BY PRODUCT_CATEGORY
ON TABLE PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PARABOX
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"groupLabels": ["Grp1", "Grp2", "Grp3", "Grp4"],
"paraboxProperties": {"activeGroup": "Grp3"}
*END
ENDSTYLE
END
```

On the output, Grp3 is a different color from the other groups:



Pie Chart Properties (pieProperties)

These properties control the basic layout of pie charts.

The following code segment shows the default values for pie chart properties:

```
"pieProperties": {

    "holeSize": 0,
    "rotation": 0,
    "skew": 0,
    "label": {
        "visible": false,
        "font": "10pt Sans-Serif",
        "color": "black"
        },
    "totalLabel": {
        "visible": false,
        "font": "10pt Sans-Serif",
        "color": "black",
        "numberFormat": "auto"
        },
    "feelerLine": (deprecated, use "dataLabels":"feelerLine")



"otherSlice": {
        "threshold": "undefined",
        "legendLabel": "Other",
        "color": "grey",
        "border": {
            "width": 1,
            "color": "transparent",
            "dash": ""
            },
        "showDataValues": true,
        "marker": {
            "shape": "square",
            "border": {
                "width": 0,
                "color": "transparent",
                "dash": ""
                }
            }
        },


    "explodeClick": {
        "enabled": false,
        "duration": 700,
        "distance": 25,
        "limitExplodeCount": false
        }
    }
```

Note: For additional properties that affect pie charts, see the following series-specific and chart-wide properties:

❑ *Defining a Border for Series Risers* on page 421 to draw borders around pie slices.

- *Pushing a Slice Away From a Pie Chart* on page 448 to explode (move away from the center) a slice from a pie.
- *Deleting a Slice From a Pie Chart* on page 446 to delete a slice from a pie.
- For charts with multiple pies, use the chartsPerRow property to defined the number of pie charts to draw in a horizontal row. For information, see *Controlling the Number of Charts in a Horizontal Row* on page 234.
- Set the dataLabels: position property to "outside" to draw data text labels outside pie slices, with feeler lines. For information, see *Showing and Formatting Data Text Labels* on page 430.
- Set the dataLabels: content property to "%" to show slice values as a percent of the total pie.
- Set the dataLabels: feelerLine property to format the feeler lines. For information, see *Showing and Formatting Data Text Labels* on page 430.

Controlling Animation When a Pie Slice Is Clicked

The explodeClick properties enable or disable and define animation when a chart user clicks on a pie slice.

Syntax: How to Control Animation When a Pie Slice Is Clicked

```
"pieProperties": {
   "explodeClick": {
      "enabled": boolean,
      "duration": number,
      "distance": number,
      "limitExplodeCount": boolean   }
}
```

where:

`"enabled": boolean`

Valid values are:

- true, which enables the explode slice animation on mouse click.
- false, which disables explode slice animation on mouse click. This is the default value.

`"duration": number`

Specifies the duration of the animation in milliseconds (a value of 1000 equals one second). Smaller values mean quicker animation. Larger values mean slower animation. The default value is 700.

`"distance": number`

Is the distance in pixels to explode the clicked slice from the pie. The default value is 25.

`"limitExplodeCount": boolean`

Valid values are:

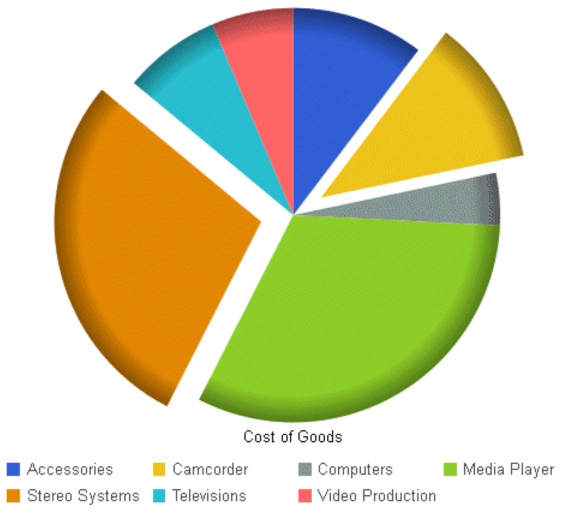
- true, to limit the explode slice to a single exploded slice.
- false, to allow multiple exploded slices. This is the default value.

Example: Controlling Animation When Clicking a Pie Slice

The following request enables the explodeClick property:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {
    "explodeClick": {"enabled": true}}
*END
ENDSTYLE
END
```

On the output, two slices have been exploded by clicking them:



To return an exploded slice to its original position, click it again.

Drawing a Ring Pie (Pie Chart With a Hole in the Middle)

The holeSize property defines the size of a transparent circle to draw in the center of a pie chart.

Note: You can also use the LOOKGRAPH parameter value PIERING to draw a pie chart with a hole in the center. For more information about LOOKGRAPH parameter settings, see *Controlling the Chart Type* on page 40.

Syntax: How to Control the Size of the Pie Hole

```
"pieProperties": {
   "holeSize": size   }
```

where:

`"holeSize": size`

Defines the size of the hole. Valid values are:

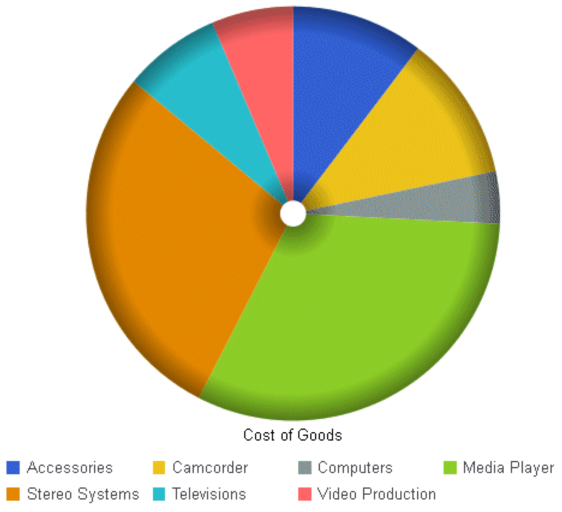
- ❏ A number that defines the size in pixels.
- ❏ A string that includes a percent symbol (%), enclosed in double quotation marks ("), that defines the size as a percent of the pie radius (for example, "10%"). The default value is zero (0).

Example: Setting the Pie Chart Hole Size

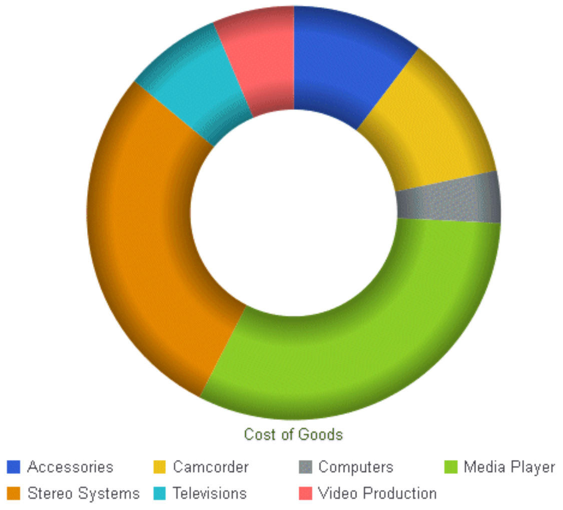
The following request generates a pie chart with a 10-pixel hole in the center:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {"holeSize": 10}
*END
ENDSTYLE
END
```

The output is:



Changing the hole size to "50%" generates the following chart:



Formatting the Pie Chart Label

The label property controls the visibility and format of the pie chart label.

Syntax: How to Format the Pie Chart Label

```
"pieProperties": {
   "label": {
      "visible": boolean,
      "font": "string",
      "color": "string"
   }
}
```

where:

`"visible": boolean`

Valid values are:

- true, which makes the label visible. This is the default value.
- false, which makes the label not visible.

`"font": "string"`

Is a string that defines the size, style, and typeface of the pie chart label. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the pie chart label, using a color name or numeric specification string. The default value is "black".

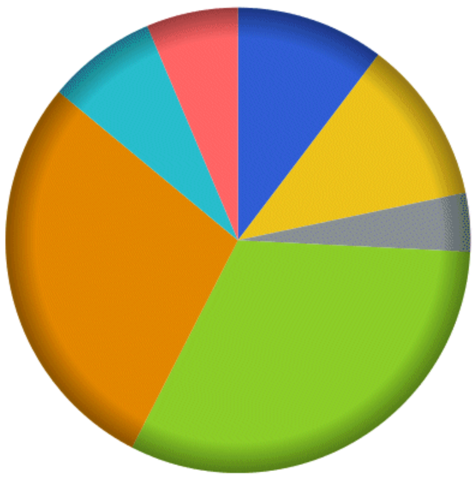
For information about specifying colors, see *Colors and Gradients* on page 85.

Example: Formatting the Pie Chart Label

The following request makes the pie label dark slate blue with a font that is 14 pt Sans-Serif and bold:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"pieProperties": {"label": {
    "visible": true, "font": "Bold 14pt Sans-Serif", "color": "DarkSlateBlue"}}
*END
ENDSTYLE
END
```

The output is:


Cost of Goods

Generating a Multi-Ring Pie Chart

The multiRing property enables or disables a multi-ring pie chart. A multi-ring pie chart is analogous to a stacked bar chart. Each slice in the pie is divided into different colored concentric rings. This sub-chart type uses the same data as a normal pie chart.

Syntax: How to Generate a Multi-Ring Pie Chart

```
"pieProperties": {
   "multiRing": boolean   }
```

where:

`"multiRing": boolean`

Valid values are:

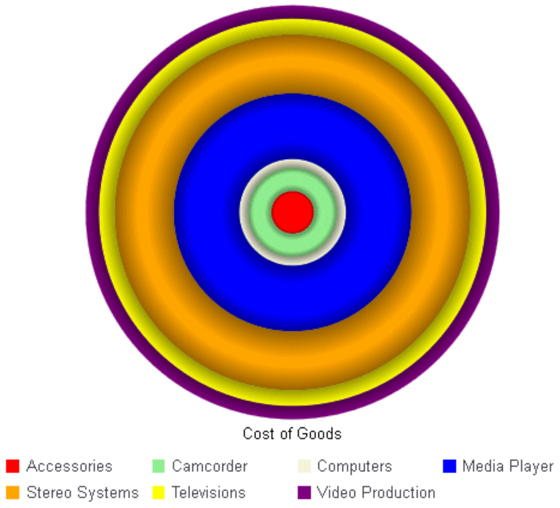
- ❑ true, which always draws one pie chart with multiple rings in each slice.
- ❑ false, which draws a traditional pie chart. False is the default value.

Example: Generating a Multi-Ring Pie Chart

The following request generates a multi-ring pie chart:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {"multiRing": true},
"series": [
    {"series": "all", "border": {"width": 1, "color": "grey"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "lightgreen"},
    {"series": 2, "color": "beige"},
    {"series": 3, "color": "blue"},
    {"series": 4, "color": "orange"},
    {"series": 5, "color": "yellow"},
    {"series": 6, "color": "purple"}]
*END
ENDSTYLE
END
```

The output is:



Merging Pie Slices Below a Specified Value

The otherSlice property controls the visibility and format of pie slices that are below a specified value. Once you define an otherSlice value, any slices that fall into that category are merged into one slice.

In addition, you can define an interaction property that drills down into the other slice when clicked. For more information, see *Special Topics* on page 739.

Syntax: How to Format Pie Slices Below a Specified Value

```
"pieProperties": {
   "otherSlice": {
      "threshold": value,
      "legendLabel": "string",
      "color": "color",
      "border": {"width": number, "color": "string", "dash": "string"}
      "showDataValues": boolean,
      "marker": {
         "shape": "string",
         "border": {"width": number, "color": "string", "dash": "string"}
      }
   }
}
```

where:

`"threshold": value`

Defines the value below which the data is merged into the other slice category. Valid values are:

- ❑ A number (absolute data value). If threshold is a number, any values below this threshold number are combined into the other slice.
- ❑ A percent string, enclosed in double quotation marks (") that includes a percent symbol. If threshold is a percent string (for example, "2%"), any slice whose overall contribution to the pie is less than this value is merged into the other slice.
- ❑ A "top *n*" string (for example "top 2"), where *n* defines the number of values that will *not* be merged into the other slice category.
- ❑ undefined, which means the other slice is disabled. This is the default value.

`"legendLabel": "string"`

Is a string identifying the label to show in the legend for the other slice. The default value is "Other".

`"color": "`*`color`*`"`

Defines a color for the other slice. Valid values are:

- ❑ A JSON object that defines a color gradient.
- ❑ A color string that specifies a gradient string or a color name or numeric specification. The default value is "grey".

For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"border":`

Defines the properties of the other slice border.

`"width":` *`number`*

Is the width of the other slice border in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a color defined by a name or numeric specification string that controls the color of the other slice border. The default value is "transparent".

`"dash": "`*`string`*`"`

Is a string that defines the other slice border dash style. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

`"showDataValues":` *`boolean`*

Valid values are:

- ❑ true, which shows the data text label for the other slice.
- ❑ false, which does not show the data text label for the other slice. This is the default value.

`"marker":`

Defines the properties of the marker to show in the legend area for the other slice.

`"shape": "`*`string`*`"`

Is a string that defines the shape of the marker to show in the legend area for the other slice. Supported shapes are arrow, bar, circle, cross, diamond, fiveStar, hexagon, hourglass, house, pin, pirateCross, plus, rectangle, sixStar, square, thinPlus, tick, circlePlus, circleMinus, or triangle. The default value is "square". Note that bar, cross, circlePlus, circleMinus, and tick markers require a border width and color.

`"border":`

Defines the properties of the marker border.

`"width":` *`number`*

Is the width of the marker border in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a color defined by a name or numeric specification string that controls the color of the marker border. The default value is "transparent".

`"dash": "`*`string`*`"`

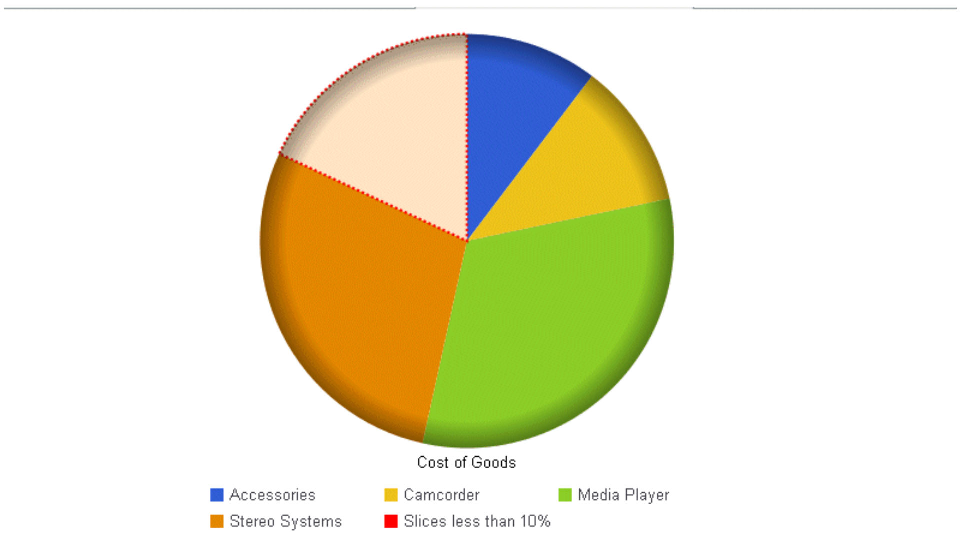
Is a string that defines the marker border dash style. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Generating a Pie Chart Other Slice

The following request merges all slices with a value less than 10% into the other slice. The other slice has the label *Slices less than 10%* in the legend. Its color is bisque and its border is a red dash. Its marker shape in the legend is a red square:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {
    "totalLabel": {"visible": false},
        "otherSlice": {
            "threshold": "10%", "legendLabel": "Slices less than 10%", "color":
"bisque",
            "border": {"width": 2, "color": "red", "dash": "2 2"},
            "marker": {"shape": "square", "width": 1, "color": "red"}}}
*END
ENDSTYLE
END
```

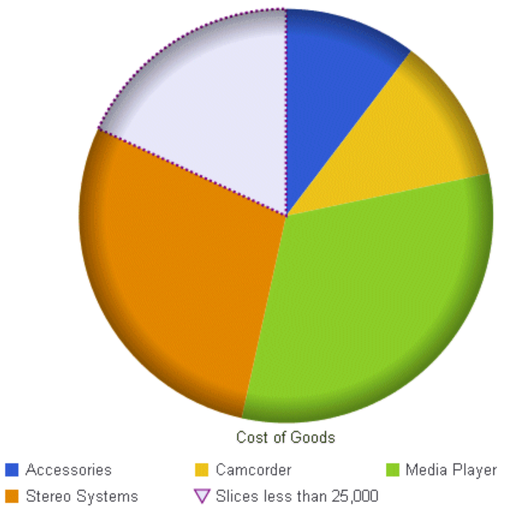
The output is:



In the following request, the other slice contains all values less than 25,000. It is lavender with a purple dashed border, and the marker for the other slice in the legend is a lavender triangle with a purple border. The legend text is *Slices less than 25,000*.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {
    "totalLabel": {"visible": false},
    "otherSlice": {"threshold": 25000, "legendLabel": "Slices less than 25,000",
        "color": "lavender", "border": {"width": 2, "color": "purple", "dash": "2 2"},
        "marker": {"shape": "triangle",
            "border": {"width": 1, "color": "purple"}}}}
*END
ENDSTYLE
END
```

The output is:



Rotating a Pie Chart

The rotation property rotates a pie chart a specified number of degrees.

Syntax: How to Rotate a Pie Chart

```
"pieProperties": {
   "rotation": number   }
```

where:

`"rotation": number`

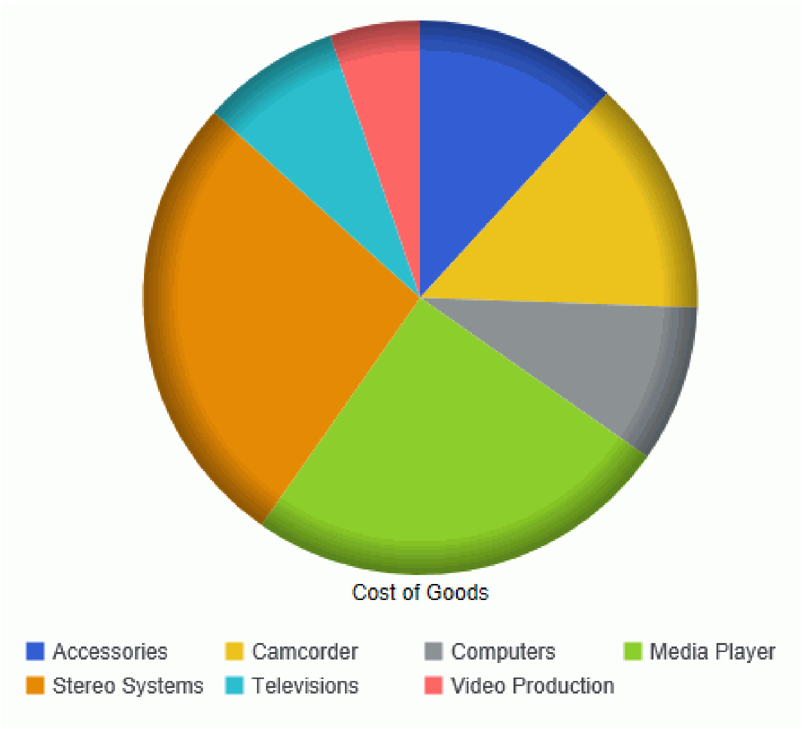
is the number of degrees, from 0 to 359, to rotate the pie chart. The default value is zero (0).

Example: Rotating A Pie Chart

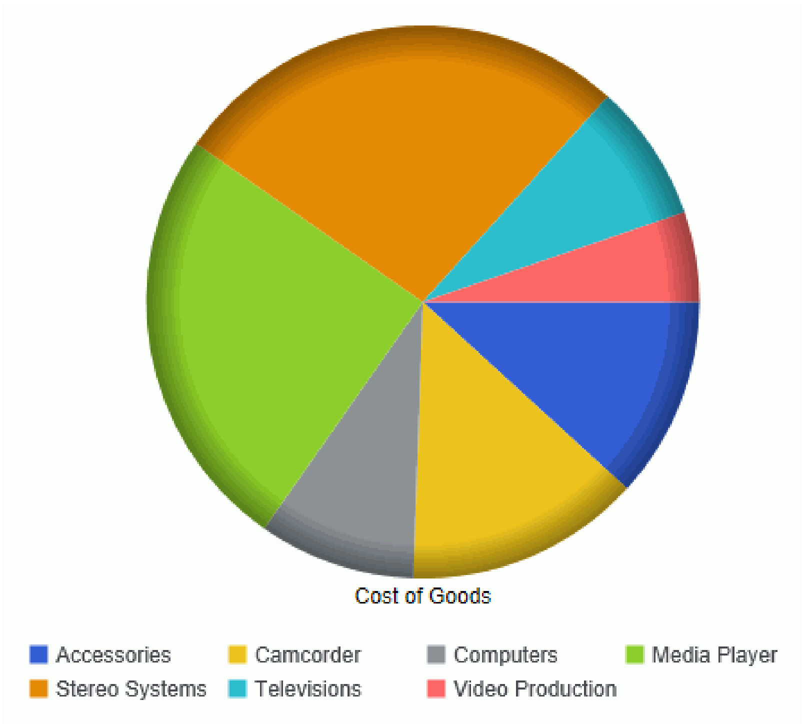
The following request generates an unrotated pie chart ({"rotation":0}) in which the blue slice starts at the zero degree position:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"pieProperties": {"rotation": 0}
*END
ENDSTYLE
END
```

On the output, the blue slice starts at the zero degrees point in the pie (12 o'clock position):



Changing the rotation value to 90 generates the following chart, in which the blue slice starts at the 90 degree point on the pie (3 o’clock position):



Skewing a Pie Chart

When depth is applied to a pie chart, the skew property controls the tile of the pie chart. For information about applying depth, see *Applying Depth to Charts* on page 237.

Syntax: How to Skew a Pie Chart

```
"pieProperties": {
   "skew": number   }
```

where:

`"skew": number`

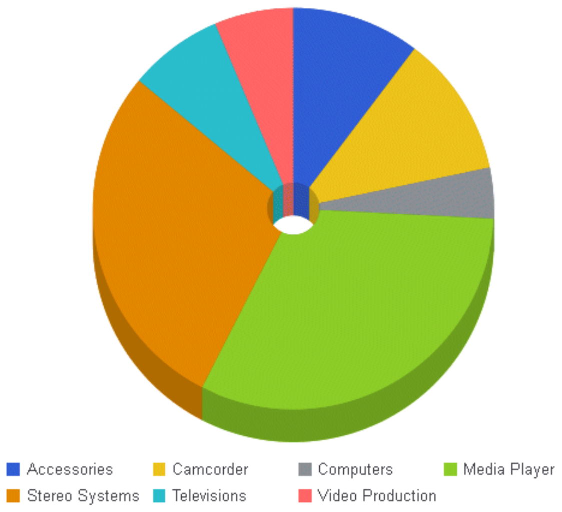
Is a number between 0 and 100. The default value is zero (0).

Example: Skewing a Pie Chart

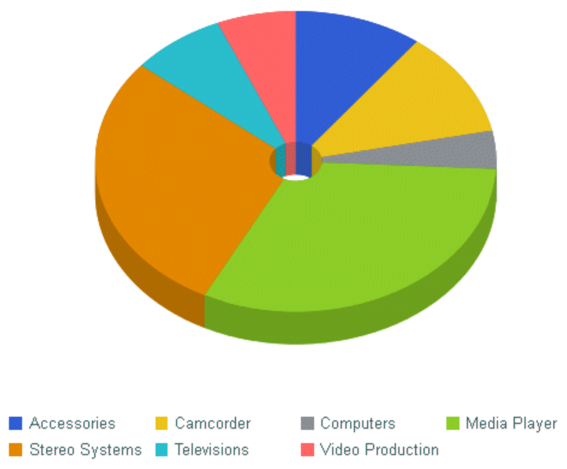
The following request generates an unskewed pie chart ({"skew":0}) with a depth of 25:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"depth": 25,
"pieProperties": {"holeSize": 20, "skew": 0}
*END
ENDSTYLE
END
```

The output is:



Changing the skew value to 25 generates the following chart:



Formatting the Total Label in a Pie Chart

The totalLabel property controls the visibility and format of the total label in a pie chart (the total value displayed in the center of the chart).

Syntax: How to Format the Total Label in a Pie Chart

```
"pieProperties": {
   "totalLabel": {
      "visible": boolean,
      "font": "string",
      "color": "string",
      "numberFormat": numformat   }
}
```

where:

`"visible": boolean`

Valid values are:

- true, which makes the label visible.
- false, which makes the label not visible. This is the default value.
- "auto", which makes the total label visible if there is a hole in the pie chart and it can fit the total label (the font size will be scaled, if possible).

`"font": "string"`

Is a string that defines the size, style, and, typeface of the label. The default value is "10pt Sans-Serif".

`"color": "string"`

Is a string that defines the color of the label, using a color name or numeric specification string. The default value is "black".

For information about specifying colors, see *Colors and Gradients* on page 85.

`"numberFormat": numformat`

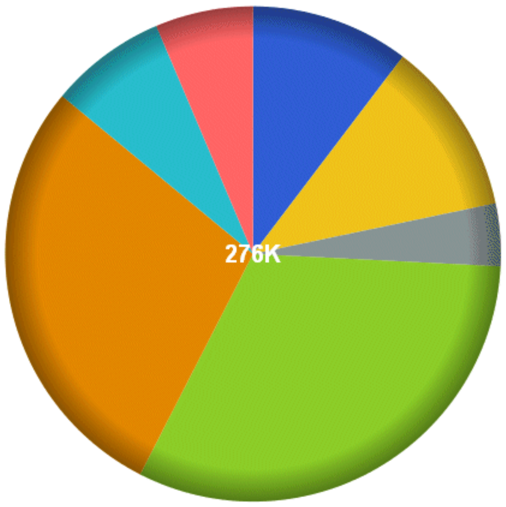
Can be specified as a JSON object, a format string, or a user-defined function. The default value is "auto". You can also use autoNumberFormats to apply a number format to all pie total labels. For information on number formats, see *Formatting Numbers* on page 108.

Example: Formatting a Pie Chart Total Label

The following request generates a pie chart with a total label that is white, in a font that is bold 14pt Sans-Serif:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"pieProperties": {
    "label": {"visible": false},
    "totalLabel": {
        "visible": true, "font": "Bold 14pt Sans-Serif", "color": "white"}}
*END
ENDSTYLE
END
```

The output is:



Polar Chart Properties (polarProperties)

These properties control the general appearance of polar and radar charts.

Syntax: How to Specify Properties for Polar and Radar Charts

```
"polarProperties": {
   "straightGridLines": boolean,
   "extrudeAxisLabels": boolean,
   "drawAsArea": boolean},
```

where:

`"straightGridLines": boolean`

Defines the grid line style. Valid values are:

- true, which draws straight grid lines with yaxis: majorGrid.
- false, which draws round grid lines with yaxis: majorGrid. The default value is false.

`"extrudeAxisLabels": boolean`

Defines where to draw y-axis labels. Valid values are:

- true, to draw y-axis labels extruded from the circular grid.
- false, to draw y-axis labels within the circular grid. The default value is false.

`"drawAsArea":` *`boolean`*

Applies to radar charts only. Valid values are:

- true, to draw a circular area chart.
- false, to draw a circular line chart. The default value is false.

Note:

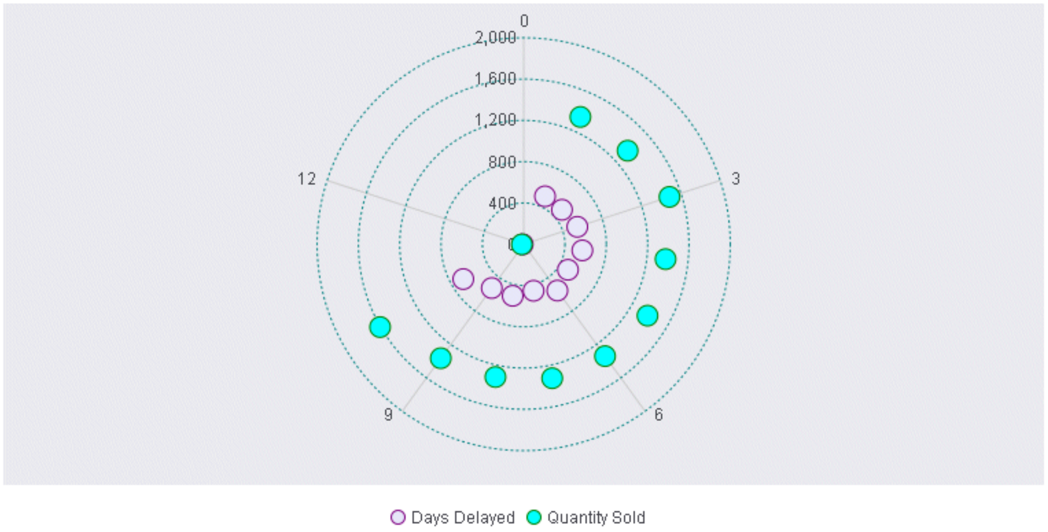
- A polar chart is a circular scatter chart.
- Like scatter charts, a polar chart requires two values to draw each marker.
- yaxis properties control the circular grid around polar chart markers. These properties also control the appearance of axis labels and gridlines.
- xaxis properties control the appearance of labels and values on the x-axis (around the outside edge of the circular grid) and x-axis gridlines.

Example: Setting Polar Chart Properties

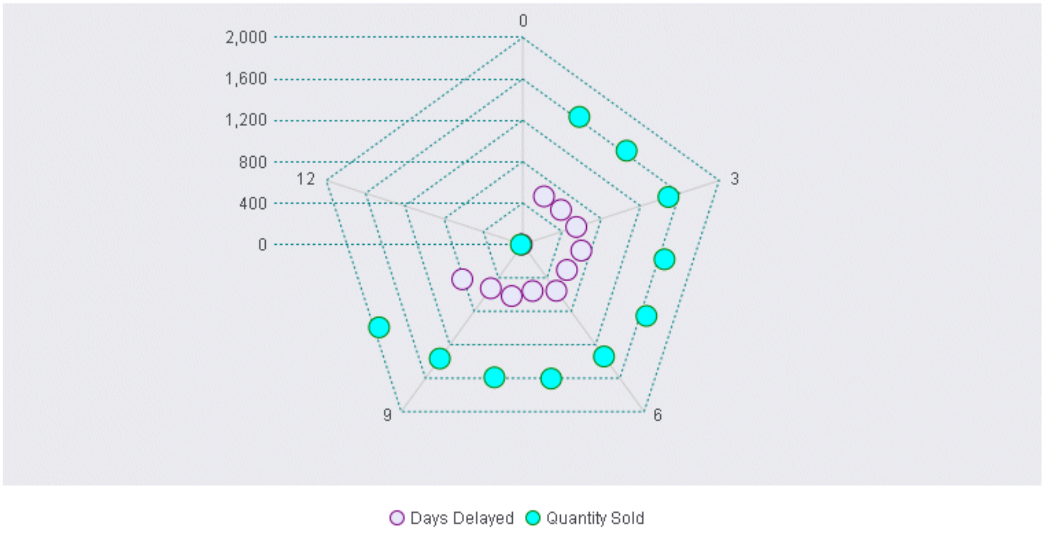
The following request generates a polar chart with the default properties. The major grid lines are dashed teal circular lines, and the series properties set the marker shapes, sizes, and colors:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
ACROSS TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH POLAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"polarProperties": {"straightGridLines": false, "extrudeAxisLabels": false},
"yaxis": {"majorGrid": {
    "lineStyle": {"width": 1, "color": "teal", "dash": "2 2"}}},
"series": [
    {"series": 0, "color": "lavender", "marker": {
        "size": 15, "shape": "circle", "border": {"width": 1, "color": "purple"}}},
    {"series": 1, "color": "cyan", "marker": {
        "size": 15, "shape": "circle", "border": {"width": 1, "color": "green"}}}]
*END
ENDSTYLE
END
```

The output is:



Changing straightGridLines to true, and extrudeAxisLabels to true, generates the following chart:

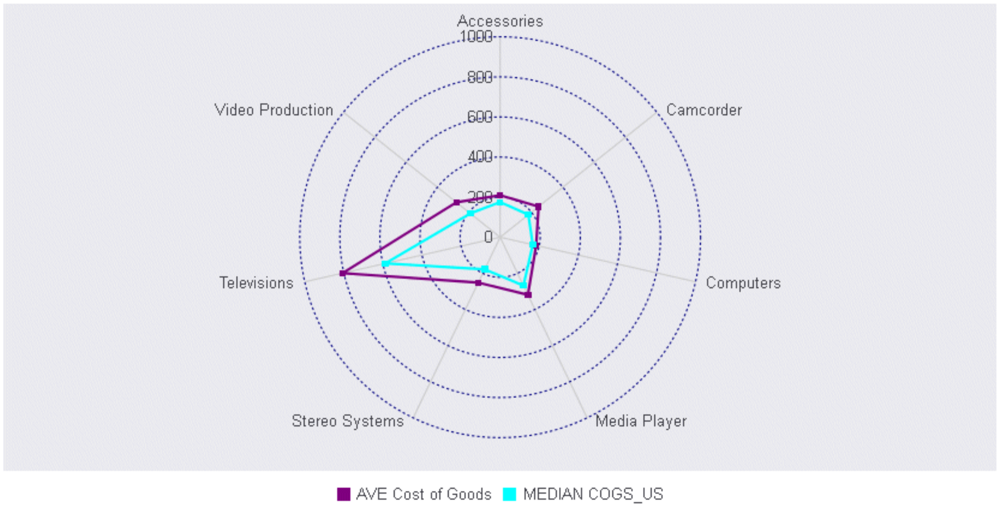


Example: Setting Radar Chart Properties

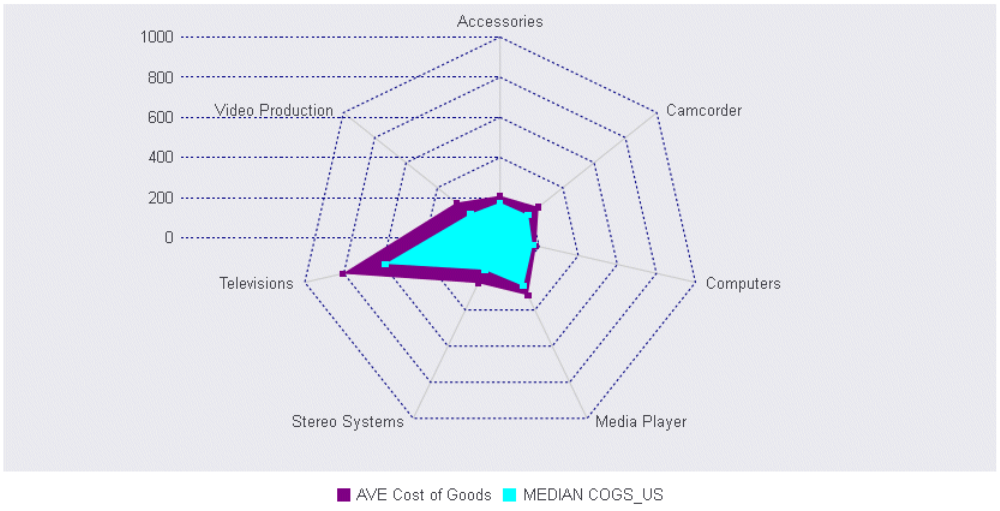
The following request generates a radar chart with the default properties:

```
GRAPH FILE WF_RETAIL_LITE
SUM AVE.COGS_US MDN.COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH RADARL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": true},
"polarProperties": {
    "straightGridLines": false, "extrudeAxisLabels": false, "drawAsArea": false},
"yaxis": {"numberFormat": "##","majorGrid": {"lineStyle": {
        "width": 1, "color": "navy", "dash": "2 2"}}},
"series": [
    {"series": 0, "color": "purple", "border": {"width": 2}},
    {"series": 1, "color": "cyan", "border": {"width": 2}},
    {"series": 2, "color": "grey", "border": {"width": 2}},
    {"series": 3, "color": "teal", "border": {"width": 2}}]
*END
ENDSTYLE
END
```

The output is:



Changing straightGridLines, extrudeAxisLabels, and drawAsArea to true generates the following chart:



Tagcloud Chart Properties (tagcloudProperties)

These properties control the engine used to draw the tagcloud, the font for the labels, and the maximum number of labels.

The following shows the properties and default values available for tagclouds:

```
"tagcloudProperties": {
    "engine": "old",
    "maxNumberOfTags": 50,
    "font": "bold 12pt Georgia"
    }
```

Syntax: How to Set Tagcloud Chart Properties

```
"tagcloudProperties": {
   "engine": "string",
   "maxNumberOfTags": number,
   "font": "string"
},
```

where:

`"engine": "string"`

Specifies which version of the tagcloud engine to use. Valid values are:

- "new", which uses the new tagcloud engine.

- "old", which uses the old tagcloud engine. This is the default value.

`"maxNumberOfTags": number`

Is the maximum number of labels to show in a tagcloud chart. The default value is 50.

`"font": "string"`

Is a string that defines the type face of labels. The default value is: "bold 12pt Georgia".

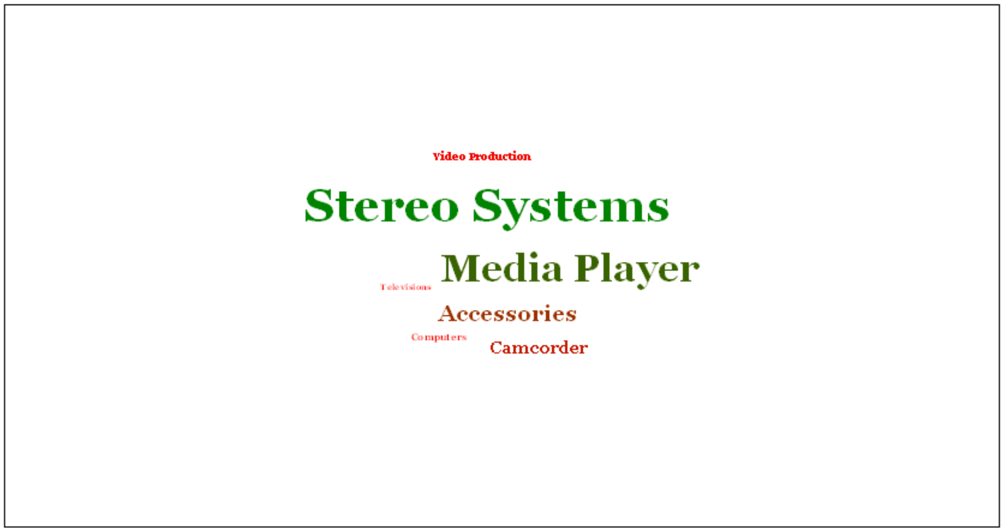
Note: Although a font specification string may include a point size, this value is not used. The data used to draw the tagcloud chart determines the size of the labels.

Example: Setting Tagcloud Chart Properties

The following request generates a tagcloud with the default properties:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
END
```

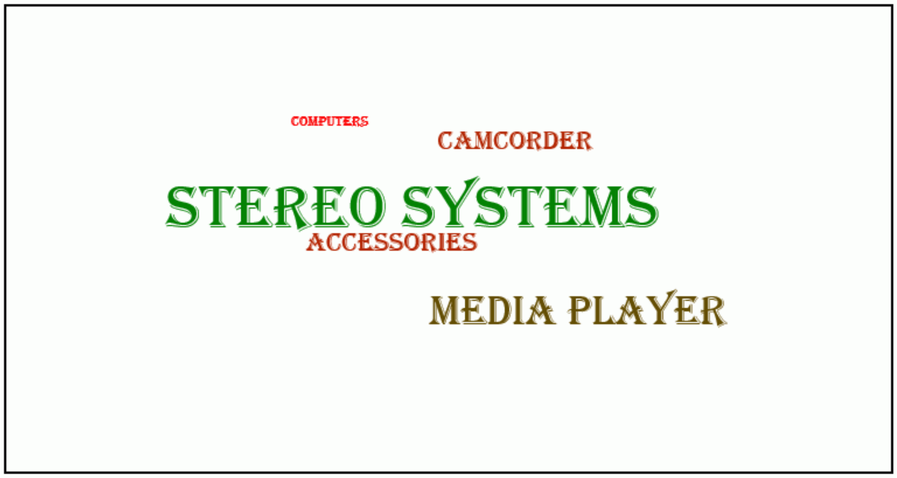
The output is:



The following version of the request uses the new tagcloud engine, limits the number of tags to 5 and sets the font to 14pt Algerian:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
ON GRAPH SET STYLE *
*GRAPH_JS
"tagcloudProperties": {
    "engine": "new",
    "maxNumberOfTags": 5,
    "font": "14pt Algerian"},
*END
ENDSTYLE
END
```

The output is:



Note: Although a font specification string may include a point size, this value is not used. The data used to draw the tagcloud chart determines the size of the labels.

3D Chart Properties (threedProperties)

These properties control the general appearance of 3D charts (chart types 'area3D', 'bar3D', and 'surface3D').

Syntax: **How to Set 3D Chart Properties**

```
"threedProperties": {
   "rotate": number,
   "tilt": number,
   "shadeSides": boolean
},
```

where:

`"rotate": number`

Is a number between 0 and 90 that defines the rotation of the 3D cube. The default value is 40.

`"tilt": number`

Is a number between 0 and 90 that defines the tilt of the 3D cube. The default value is 40.

`"shadeSides": boolean`

Valid values are:

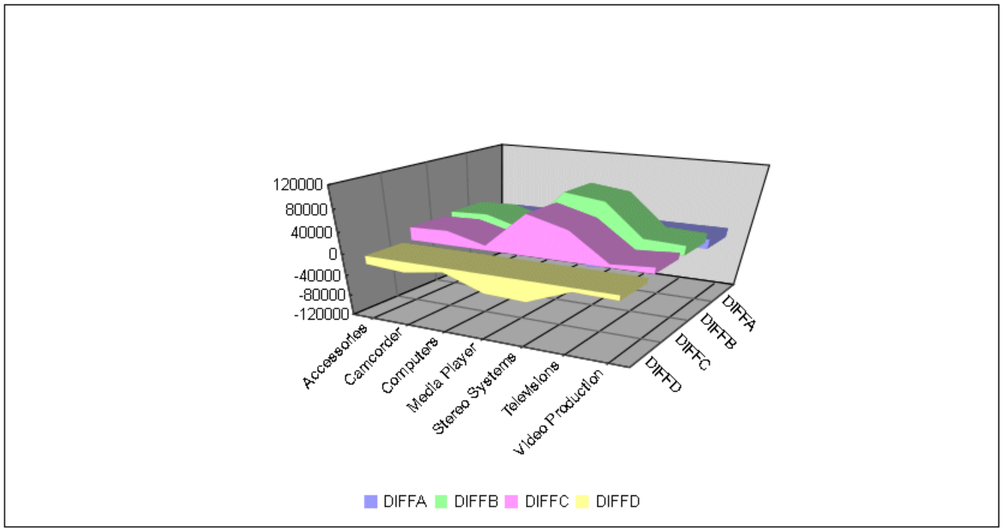
- true, to shade 3D risers. This is the default value.
- false, to not shade 3D risers.

Example: **Setting Properties for a 3D Area Chart**

The following request generates a 3D area chart with the default properties:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - (COGS_US * DISCOUNT_US)/100;
DIFFD = COGS_US - MSRP_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DAREAS
END
```

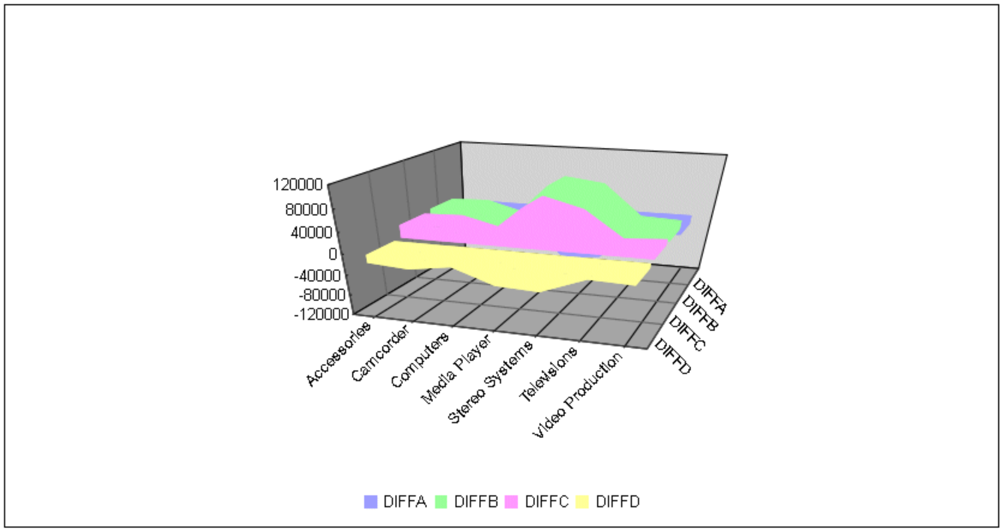
The output is:



The following version of the request sets the rotation to 20, the tilt to 30, and shadeSides to false:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - (COGS_US * DISCOUNT_US)/100;
DIFFD = COGS_US - MSRP_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DAREAS
ON GRAPH SET STYLE *
*GRAPH_JS
"threedProperties": {
    "rotate": 20, "tilt": 30, "shadeSides": false}
*END
ENDSTYLE
END
```

The output is:

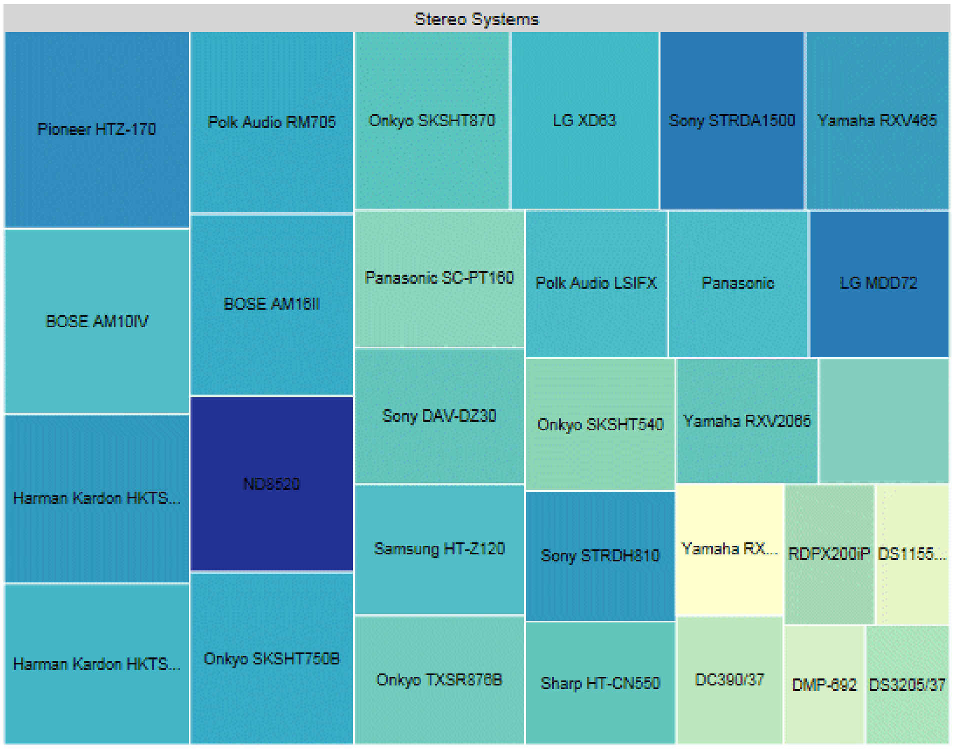


Treemap Chart Properties (treemapProperties)

These properties control the appearance of header cells and cell borders in a treemap chart.

Note: In a treemap, if "legend": {"visible": true} is in effect, you will get a color scale legend only if the chart includes color data. You will get a series legend only if the chart includes series IDs data.

If a data label in a treemap does not fit into its rectangle, it will be truncated and appended with ellipsis (...), as in the following image.



This code segment shows the default values.

```
"treemapProperties": {
    "scaleCellFonts": false,
    "header": {
        "height": undefined,
        "fill": "lightgrey",
        "border": {
            "width": 0,
            "color": "lightgrey",
            "dash": ""},
        "label": {
            "visible": true,
            "font": "8pt Sans-Serif",
            "color": "black"}},
    "cellBorder": {
        "width": 1,
        "color": "white",
        "dash": "",
        "outerCellWidth": 3}}
```

Scaling Fonts in Treemap Cells

The scaleCellFonts property controls how labels are drawn in the treemap cells.

Syntax: How to Scale Fonts in Treemap Cells

```
"treemapProperties": {
    "scaleCellFonts": boolean,
    }
```

where:

`"scaleCellFonts": boolean`

Valid values are:

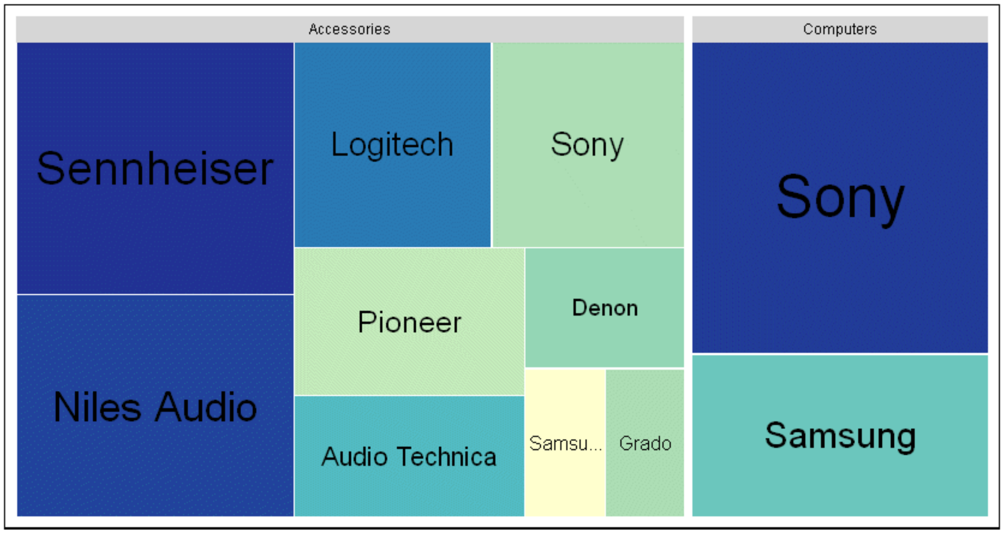
- true, which scales font sizes for labels drawn inside each cell according to the cell area.
- false, which uses the same font size for all cell labels and does not draw labels when the label size exceeds the cell size. The default value is false.

Example: Scaling Fonts in Treemap Cells

The following request generates a treemap chart. The scaleCellFonts property is set to true:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BRAND
WHERE PRODUCT_CATEGORY EQ 'Computers' OR 'Accessories'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"dataLabels": {"visible": true},
"treemapProperties": {"scaleCellFonts": true}
*END
ENDSTYLE
END
```

On the output, the cells with larger areas have labels with larger font sizes:



Formatting the Treemap Header

These properties format the header in a treemap chart.

Syntax: **How to Format a Treemap Header**

```
"treemapProperties": {
      "header": {
      "height": height,
      "fill": "string",
      "border": {
         "width": width,
         "color": "string",
         "dash": "string"
      },
      "label": {
         "visible": boolean,
         "font": "string",
         "color": "string"
      }
```

`"height": height`

Defines the height of the header. Valid values are:

- undefined, which automatically calculates the header height to be 33% larger than the header label height. This is the default value.

- A number that defines the height in pixels.
- A string that includes a percent symbol, enclosed in single quotation marks (for example, "5%"). When this property is set to a percent string, the header height will be the specified percentage of the overall height of the treemap.

`"fill": "`*`string`*`"`

Can be undefined, a color definition, or a gradient definition. The default value is "lightgrey". For information about defining colors and gradients, see *Colors and Gradients* on page 85.

`"border":`

Defines the properties of the header border.

`"width":` *`width`*

Is a number of pixels that defines the width of the header border. The default value is 0.

`"color": "`*`string`*`"`

Is a color or gradient definition string that defines the border color. The default value is "lightgrey".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the header border. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

`"label":`

Defines the properties of the header labels.

`"visible":` *`boolean`*

Controls the visibility of the header label. Valid values are:

- true, which makes the header label visible. This is the default value.
- false, which makes the header label not visible.

`"font": "`*`string`*`"`

Is a font string that defines the font of the header label. The default value is "8pt Sans-Serif".

"color": "*string*"

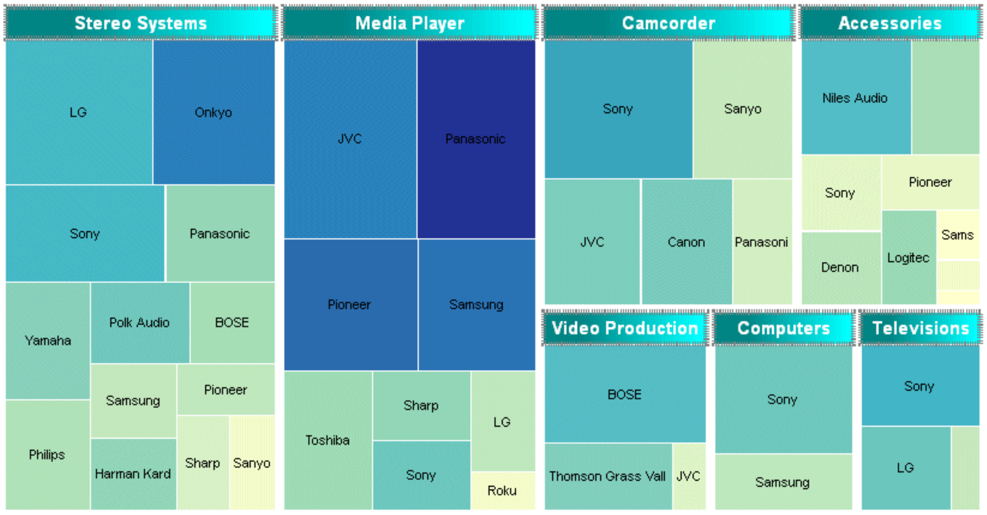
Is a color definition string that defines the color of the header label. The default value is "black".

Example: **Formatting a Treemap Header**

The following request generates a treemap chart. The header is 25 pixels high. Its border is a grey dashed line 4 pixels wide, filled with a linear gradient that transitions from teal to cyan. The label in the header is white and bold 10pt Sans-Serif:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"treemapProperties": {
    "header": {
        "height": 25,
        "fill": "linear-gradient(0%,0%,100%,0%, 20% teal, 95% cyan)",
        "border": {"width": 4, "color": "grey", "dash": "1 1"},
        "label": {"visible": true, "font": "bold 10pt Sans-Serif", "color": "white"}}}
*END
ENDSTYLE
END
```

The output is:



Formatting Treemap Cell Borders

These properties format the treemap cell borders.

Syntax: How to Format Treemap Cell Borders

```
"treemapProperties": {
    "cellBorder": {
      "width": number,
      "color": "string",
      "dash": "string",
      "outerCellWidth": number   }
},
```

where:

`"width": number`

Is a number that defines the width of the cell border in pixels. The default value is 1.

`"color": "string"`

Is a color or gradient definition string that defines the cell border color. The default value is "white".

`"dash": "*string*"`

Is a string that defines the dash style of the cell border. The default value is "", which produces a solid line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

`"outerCellWidth": *number*`

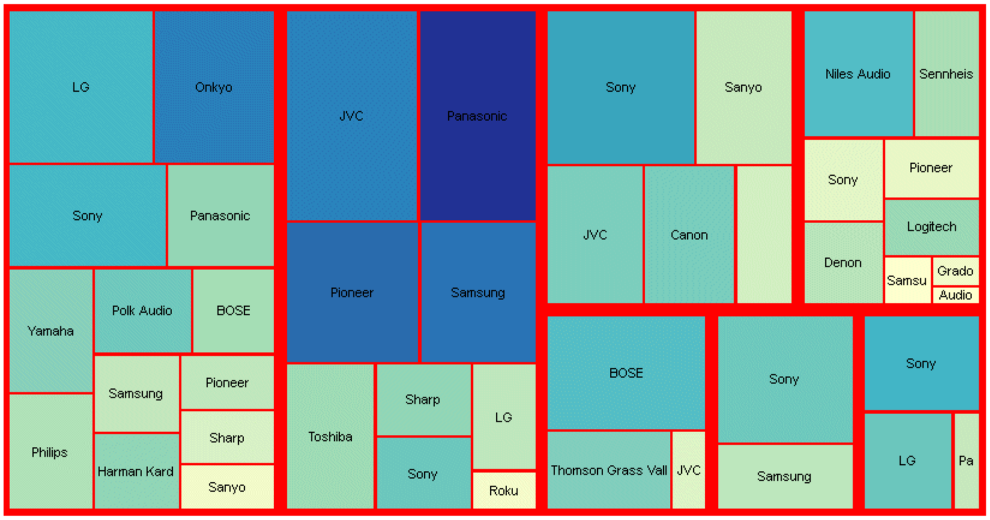
Is a number of pixels that defines the width of the border to draw around the top-level of cells. This property controls the width of the borders around only the top-level (root) nodes. The default value is 3.

Example: Formatting Treemap Cell Borders

The following generates a treemap chart. The headers are eliminated by making their height zero (0). The cell borders are red with a width of 2 pixels, and the outer border has a width of 4 pixels:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BRAND
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"treemapProperties": {
    "header": {"height": 0},
    "cellBorder": {"width": 2, "color": "red", "outerCellWidth": 4}}
*END
ENDSTYLE
END
```

The output is:



Waterfall Chart Properties (waterfallProperties)

Waterfall charts graphically illustrate the cumulative effect of sequentially introducing positive or negative values to an initial value. The initial and final (or total) values are represented by whole columns drawn from the ordinal axis baseline. Intermediate positive and negative values are drawn as floating columns.

These properties control the general layout of a waterfall chart.

The following code segment shows the default settings:

```
"waterfallProperties":
{
   "appendTotalRiser": true,
   "totalLabel: "Total",
   "positiveRiserColor": "#77b39a",
   "negativeRiserColor": "#e2675b",
   "zeroRiserColor": "#7593bd",
   "totalRiserColor: "#7593bd",
   "otherRiserColor": "#aaaaaa",
   "subtotalRisers": [],
   "otherRisers": [],
   "subtotalFormat":
   {
      "bold": undefined,
      "color": undefined,
      "numberFormat": undefined,
      "dataLabelPosition": undefined,
      "dividerLine":
        {
         "width": 0,
         "color": "black",
         "dash": ""
         }
    },
  "firstLastFormat":
  {
    "color": undefined,
    "border":
     {
      "width": 0,
      "color": undefined,
      "dash": ""
     }
  },
   "connectorLine":
     {
      "width": 1,
      "color": "black",
      "dash": ""
      }
},
```

Appending a Total Riser in a Waterfall Chart

This property generates a total riser as the last riser in a waterfall chart.

Syntax: How to Append a Total Riser in a Waterfall Chart

```
"waterfallProperties": {
   "appendTotalRiser": boolean   },
```

where:

`"appendTotalRiser": boolean`

Valid values are:

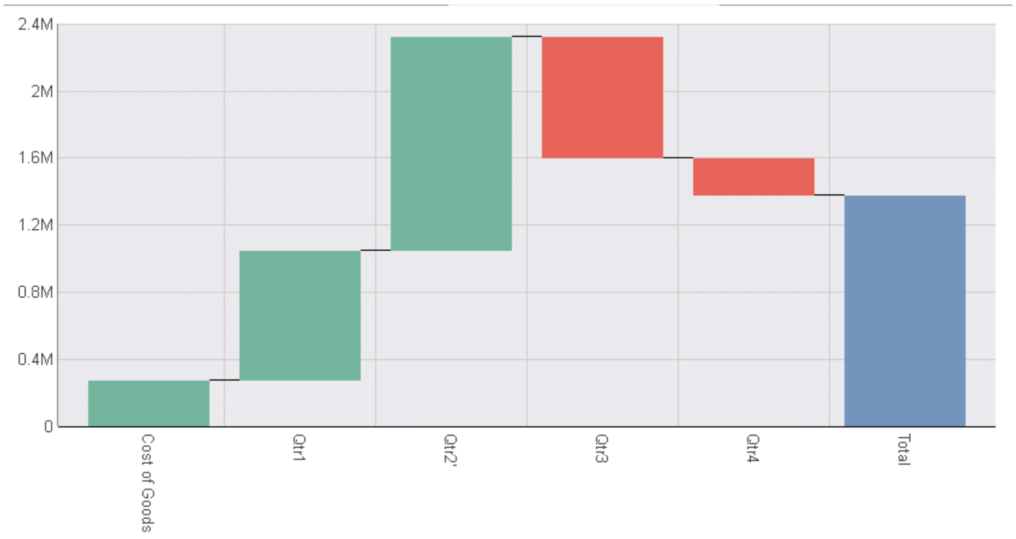
- ❏ true, which draws a total riser. This is the default value.
- ❏ false, which does not draw a total riser.

Example: Appending a Total Riser in a Waterfall Chart

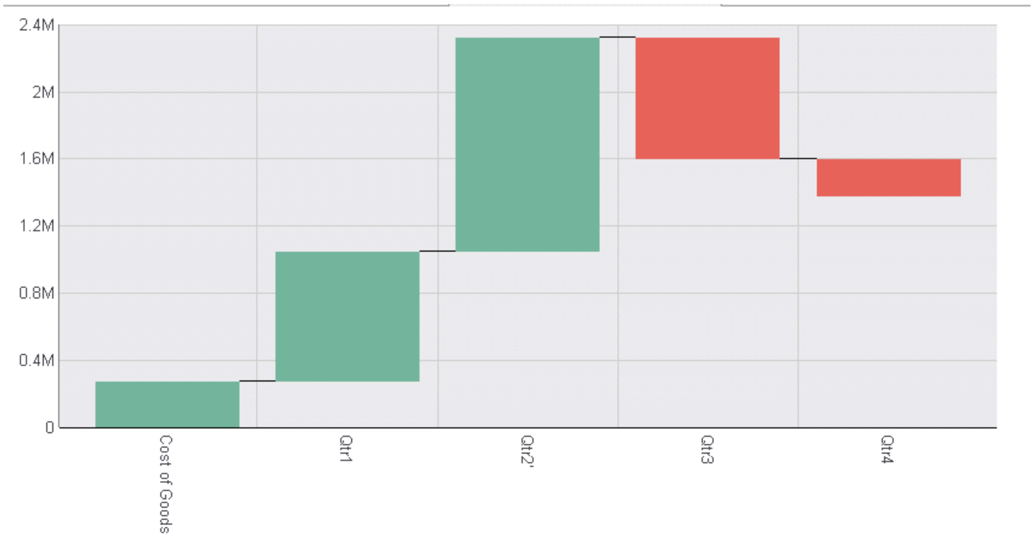
The following request generates a waterfall chart with a total riser (the default):

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
DECR1 = (COGS_US - 1000);
DECR2 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 AS 'Qtr1'
INCR2 AS 'Qtr2'
DECR1  AS 'Qtr3'
DECR2 AS 'Qtr4'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"waterfallProperties": {"appendTotalRiser": true}
*END
ENDSTYLE
END
```

The output is:



Changing appendTotalRiser to false generates the following chart:



Providing a Custom Label for the Total Riser

When you generate a total riser in a waterfall chart, this property enables you to customize the label for the total riser.

Syntax: How to Customize the Total Riser Label

```
"waterfallProperties": {
   "appendTotalRiser": true,
   "totalLabel": "string"   },
```

where:

`"appendTotalRiser": true`

Must be true in order to generate a total label. This is the default value, so this property can be omitted.

`"totalLabel": "string"`

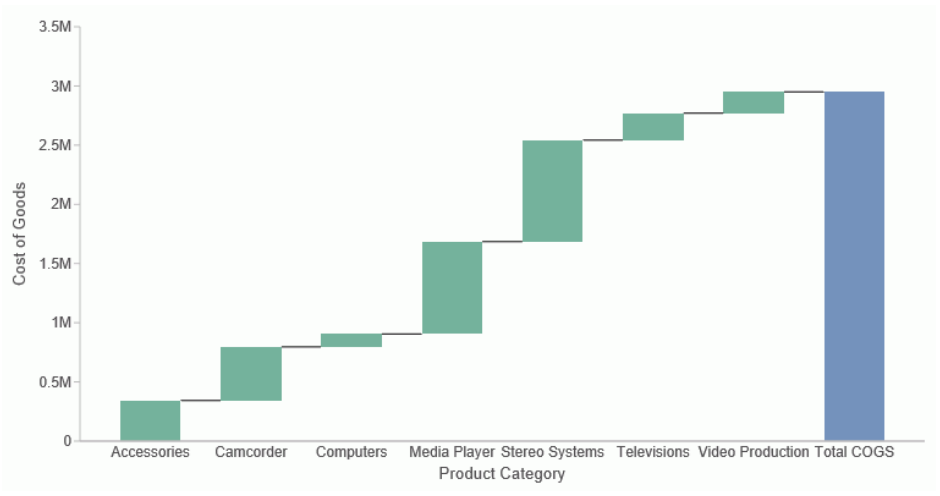
Is a string to use as the label for the total riser. The default label is Total.

Example: Customizing the Total Riser Label

The following request generates a total riser and customizes the label on the x-axis to *Total COGS*.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH WATERFALL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS
"waterfallProperties":
{
"appendTotalRiser": true,
"totalLabel": "Total COGS",
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Formatting Connector Lines in a Waterfall Chart

These properties control the appearance of connector lines between risers in a waterfall chart.

Syntax: How to Format Connector Lines in a Waterfall Chart

```
"waterfallProperties": {
   "connectorLine": {
      "width": number,
      "color": "string",
      "dash": "string"
   }
   },
```

where:

`"width": number`

Is the width of the connector line in pixels. The default value is 1.

`"color": "string"`

Is a color name or numeric specification string. The default value is "black".

`"dash": "string"`

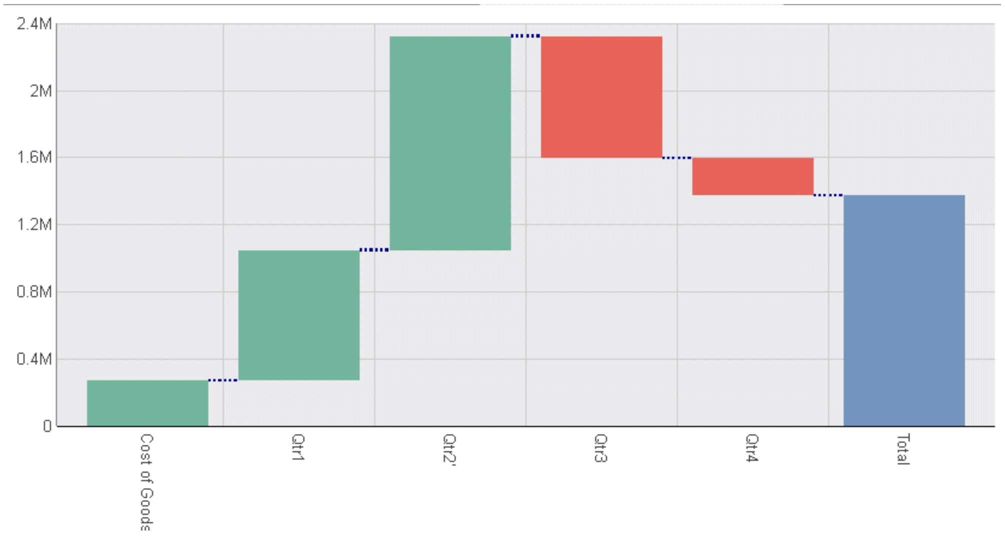
Is a string that defines the dash style of the connector lines. The default value is "", which produces a solid line. Use a string of numbers that specifies the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Formatting Waterfall Chart Connector Lines

The following request generates a waterfall chart in which the connector lines are 2 pixels wide, dark blue, and dashed:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
DECR1 = (COGS_US - 1000);
DECR2 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 AS 'Qtr1'
INCR2 AS 'Qtr2'
DECR1  AS 'Qtr3'
DECR2 AS 'Qtr4'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"waterfallProperties": {
    "connectorLine": {"width": 2, "color": "darkblue", "dash": "2 2"}}
*END
ENDSTYLE
END
```

The output is:



Controlling the Color of Negative Risers in a Waterfall Chart

This property controls the color of risers that represent negative values in a waterfall chart.

Note: You can use series-specific properties to format risers individually, by specifying a color for each group under series 0.

Syntax: How to Color Negative Risers in a Waterfall Chart

```
"waterfallProperties": {
   "negativeRiserColor": "color"   },
```

where:

`"negativeRiserColor": "color"`

Defines the negative riser color. Valid values are:

- ❑ A color string, enclosed in double quotation marks ("), that defines a color by name, numeric specification string, or gradient definition string. The default value is "#e2675b".
- ❑ A JSON object that defines a gradient.

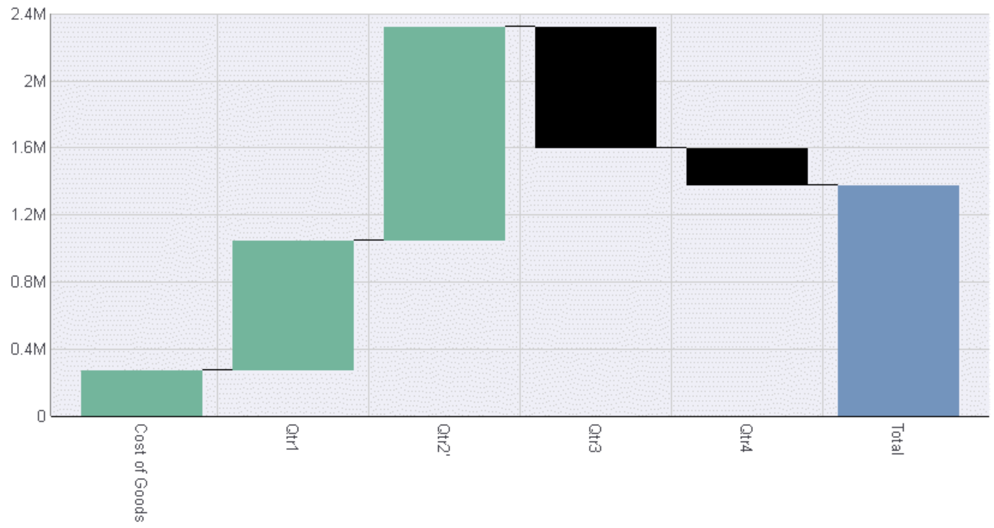
For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

Example: Setting the Waterfall Chart Negative Riser Color

The following request generates a waterfall chart in which the risers for negative values are black:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
DECR1 = (COGS_US - 1000);
DECR2 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 AS 'Qtr1'
INCR2 AS 'Qtr2'
DECR1  AS 'Qtr3'
DECR2 AS 'Qtr4'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"waterfallProperties": {"negativeRiserColor": "black"}
*END
ENDSTYLE
END
```

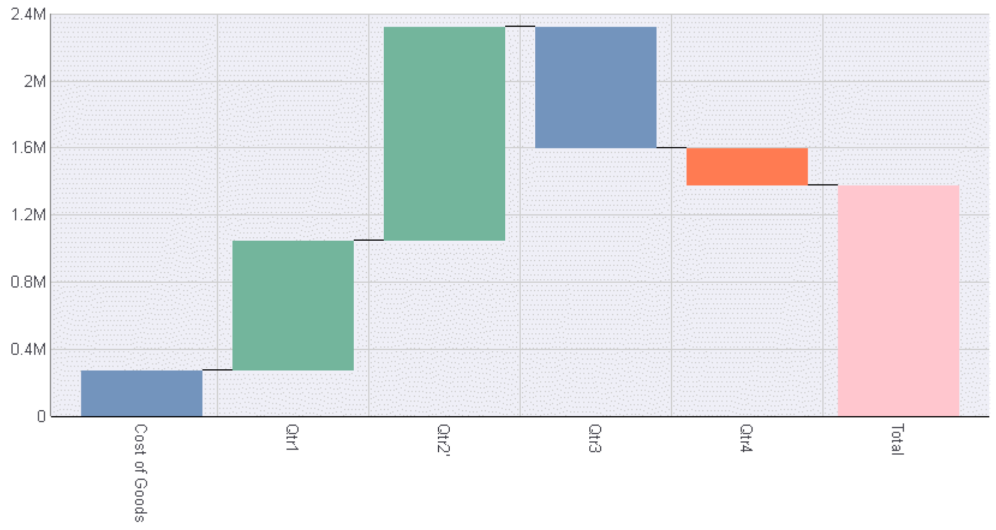
The output is:



Note: You can use series-specific properties to format risers individually, by specifying a color for each group under series 0. For example:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
DECR1 = (COGS_US - 1000);
DECR2 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 AS 'Qtr1'
INCR2 AS 'Qtr2'
DECR1  AS 'Qtr3'
DECR2 AS 'Qtr4'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"series": [
    {"series": 0, "group": 0, "color": "#7593bd"},
    {"series": 0, "group": 1, "color": "#77b39a"},
    {"series": 0, "group": 2, "color": "#77b39a"},
    {"series": 0, "group": 3, "color": "#7593bd"},
    {"series": 0, "group": 4, "color": "coral"},
    {"series": 0, "group": 5, "color": "pink"}]
*END
ENDSTYLE
END
```

On the output, each riser has the color specified for that group:



Specifying a Waterfall Chart Other Riser Color

The otherRiserColor property controls the color of risers that represent *other* values (if any) in a waterfall chart. Other values are defined in the otherRisers array.

Syntax: How to Specify a Waterfall Chart Other Riser Color

```
"waterfallProperties": {
   "otherRiserColor": "color"   },
```

where:

`"otherRiserColor": "color"`

Specifies a color for the *other* risers . Valid values are:

- ❑ A color string, enclosed in double quotation marks ("), that defines a color by name, by a numeric specification string, or by a gradient definition string. The default value is "#aaaaaa".
- ❑ A JSON object that defines a gradient.

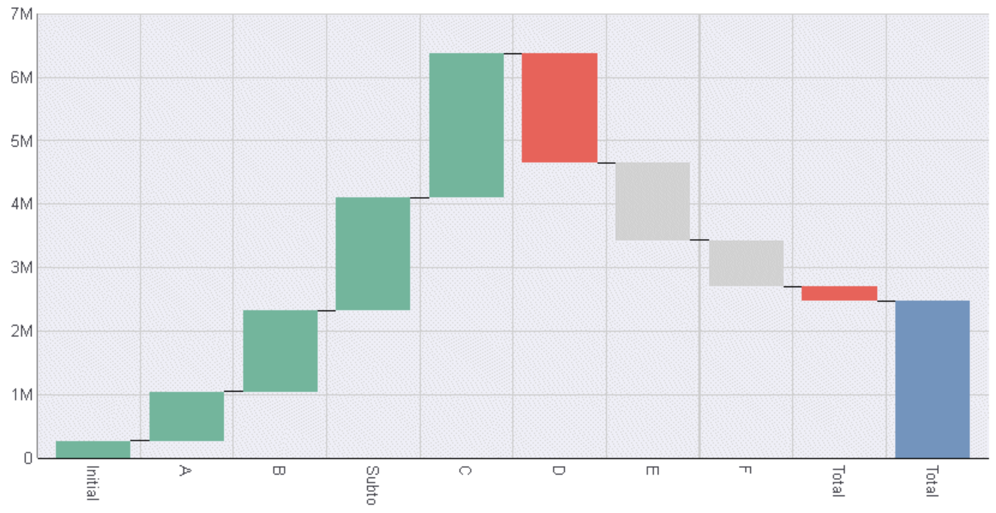
For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

Example: Specifying a Waterfall Chart Other Riser Color

The following request generates a waterfall chart in which the other risers are colored light grey:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +1500;
INCR4 = COGS_US +2000;
DECR1 = (COGS_US - 2000);
DECR2 = (COGS_US -1500);
DECR3 = (COGS_US -1000);
DECR4 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3 INCR4
DECR1 DECR2 DECR3 DECR4
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Initial", "A", "B", "Subtotal", "C", "D", "E", "F", "Total"],
"waterfallProperties": {
    "otherRiserColor": "lightgrey", "otherRisers": ["E", "F"]}
*END
ENDSTYLE
END
```

The output is:



Note: You can also use series-specific properties to format risers individually. For an example, see *Controlling the Color of Negative Risers in a Waterfall Chart* on page 689.

Specifying Other Risers

The otherRisers property defines one or more groups or values to be interpreted as *other* values. Risers representing these values will be assigned the color defined by the otherRiserColor property.

Syntax: How to Specify Other Risers

```
"waterfallProperties": {
   "otherRisers": [array]
   },
```

where:

`"otherRisers": [array]`

Can be an array of:

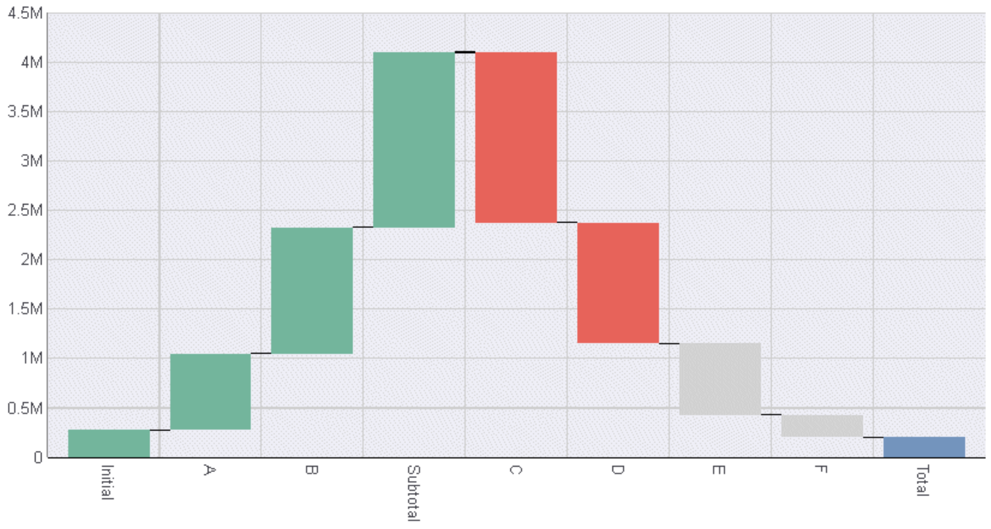
- ❑ numbers, that specify group indices.
- ❑ strings, enclosed in double quotation marks ("), that specify group names.

Example: Specifying Other Risers

The following request generates a waterfall chart. The groupLabels object assigns group label names to each group, and the waterfallProperties otherRisers property assigns groups E and F to the other risers category:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +1500;
DECR1 = (COGS_US - 2000);
DECR2 = (COGS_US -1500);
DECR3 = (COGS_US -1000);
DECR4 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3
DECR1  DECR2 DECR3 DECR4
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Initial", "A", "B", "Subtotal", "C", "D", "E", "F", "Total"],
"waterfallProperties": {
    "otherRiserColor": "lightgrey", "otherRisers": ["E", "F"]}
*END
ENDSTYLE
END
```

On the chart output, the other risers are drawn in light grey, the color assigned by the otherRiserColor property:



Specifying a Waterfall Chart Positive Riser Color

The positiveRiserColor property controls the color of risers that represent positive values in a waterfall chart.

Syntax: How to Color Waterfall Chart Positive Risers

```
"waterfallProperties": {
   "positiveRiserColor": "color"   },
```

where:

`"positiveRiserColor": "color"`

Defines a color for the positive risers. Valid values are:

- A color string, enclosed in double quotation marks ("), that defines a color by name, by a numeric specification string, or by a gradient definition string. The default value is "#77b39a".
- A JSON object that defines a gradient.

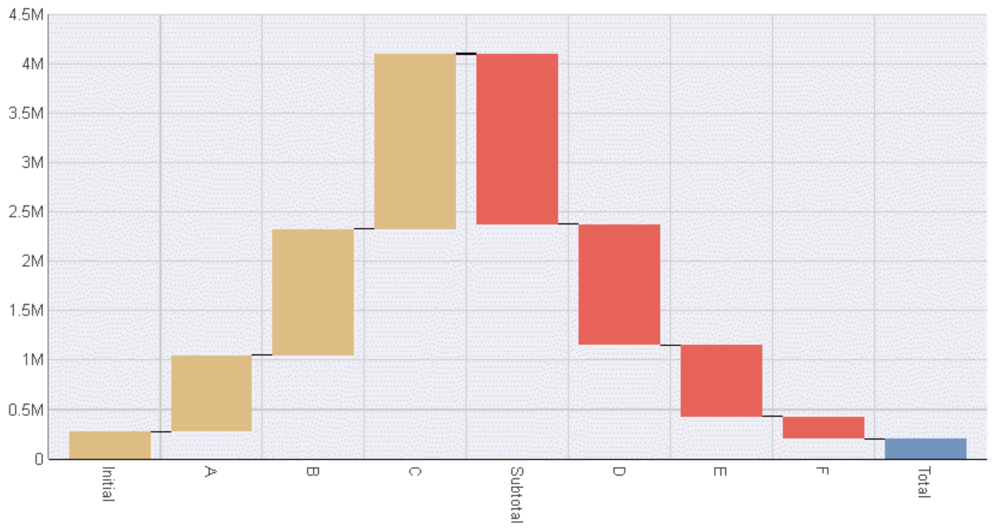
For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

Example: Specifying a Waterfall Chart Positive Riser Color

The following request generates a waterfall chart in which the risers for positive values are the color burlywood:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +1500;
DECR1 = (COGS_US - 2000);
DECR2 = (COGS_US -1500);
DECR3 = (COGS_US -1000);
DECR4 = (COGS_US -500);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3
DECR1  DECR2 DECR3 DECR4
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Initial", "A", "B", "C", "Subtotal", "D", "E", "F", "Total"],
"waterfallProperties": {"positiveRiserColor": "burlywood"}
*END
ENDSTYLE
END
```

The output is:



Note: You can also use series-specific properties to format risers individually. For an example, see *Controlling the Color of Negative Risers in a Waterfall Chart* on page 689.

Specifying Subtotal Values

This property defines one or more groups or values to be interpreted as *subtotal* values.

Syntax: How to Assign Subtotal Values

```
"waterfallProperties": {
   "subtotalRisers": [array]
   },
```

where:

`"subtotalRisers": [array]`

Can be an array of:

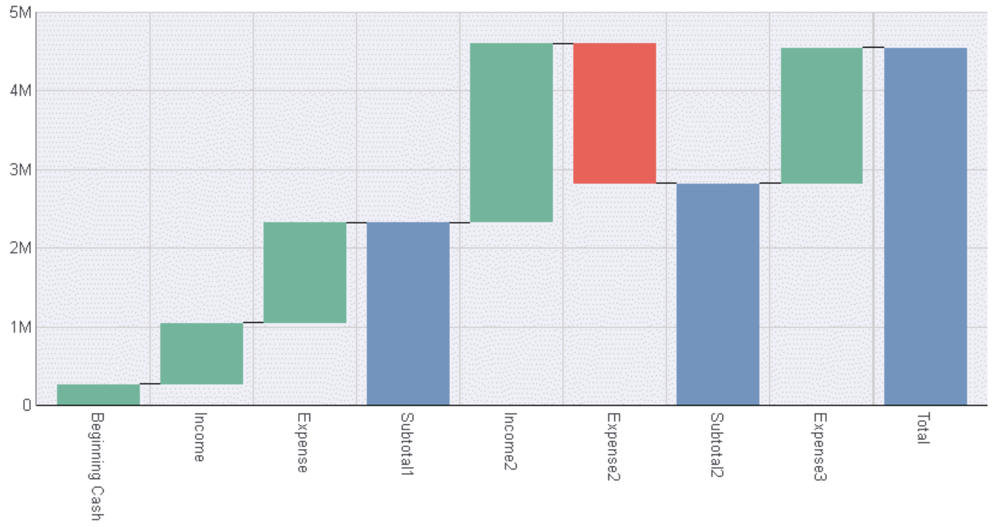
- numbers, that specify group indices.
- strings, enclosed in double quotation marks ("), that specify group names.

Example: Specifying Subtotal Risers

The following generates a waterfall chart with two subtotal risers. The groupLabels object assigns names to each riser, and the subtotalRisers object specifies those names in its array:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +INCR1 +INCR2;
INCR4 = COGS_US +2000;
DECR1 = -(INCR4- 500);
DECR2 = INCR3 +500;
DECR3 = -(COGS_US-2000);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3 INCR4
DECR1 DECR2  DECR3
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Beginning Cash", "Income", "Expense", "Subtotal1",
    "Income2", "Expense2", "Subtotal2", "Expense3", "TOTAL"],
"waterfallProperties": {"subtotalRisers": ["Subtotal1", "Subtotal2"]}
*END
ENDSTYLE
END
```

The output is:



Specifying the Color of Waterfall Chart Subtotal and Total Risers

The zeroRiserColor property controls the color of any riser placed in the subtotal array and the riser that represents the total value in a waterfall chart.

Syntax: How to Color Waterfall Chart Initial Value, Subtotal, and Total Risers

```
"waterfallProperties": {
   "zeroRiserColor": "color"   },
```

where:

`"zeroRiserColor": "color"`

Specifies the color for any riser placed in the subtotal array and the riser representing the total value. Valid values are:

- A color string, enclosed in double quotation marks ("), that defines a color by name, by a numeric specification string, or by a gradient definition string. The default value is "#7593bd".
- A JSON object that defines a gradient.

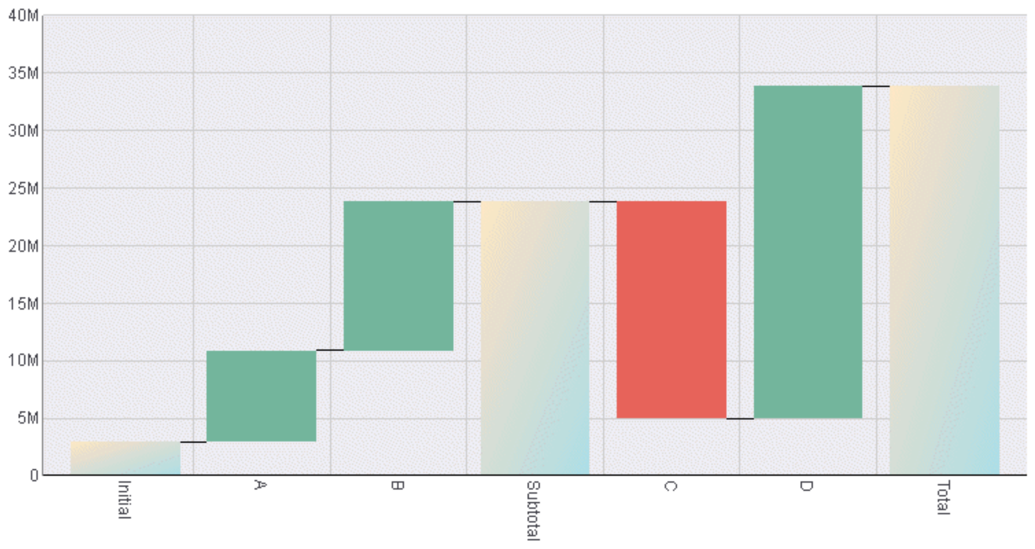
For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

Example: Setting the Waterfall Chart zeroRiserColor Property

The following request generates a waterfall chart and sets the zeroRiserColor property to a linear gradient that transitions from bisque to light blue. It adds the initial value riser to the subtotal array, so the initial value riser also displays using the specified gradient:

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +INCR1 +INCR2;
DECR1 = -(INCR3- 500);
DECR2 = INCR3 +500;
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3
DECR1 DECR2
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Initial", "A", "B", "Subtotal", "C", "D", "Total"],
"waterfallProperties": {
    "subtotalRisers": ["Initial", "Subtotal"],
    "zeroRiserColor": {
        "type": "linear",
        "start": {"x": "0%", "y": "0%"},
        "end": {"x": "100%", "y": "100%"},
        "stops": [[0, "bisque"], [1, "lightblue"]]}}
*END
ENDSTYLE
END
```

The output is:



Note: You can also use series-specific properties to format risers individually. For an example, see *Controlling the Color of Negative Risers in a Waterfall Chart* on page 689.

Formatting Axis and Data Label Properties of Subtotal Risers

The subtotalFormat properties enable you to customize the axis labels, data labels, and divider lines for subtotal risers.

Syntax: How to Format Axis and Data Label Properties of Subtotal Risers

```
"waterfallProperties":
{
  "subtotalFormat":
     {
        "bold": boolean,
        "color": "string",
        "numberFormat": format,
        "dataLabelPosition": "string",
        "dividerLine":
          {
           "width": number,
           "color": "string",
           "dash": "string"
           }
      }
}
```

where:

`"bold": `*`boolean`*

Can be one of the following values.

- true, which makes the axis labels of subtotal risers bold.
- false, which does not bold the axis labels of subtotal risers.

The default value is *undefined*.

`"color": "`*`string`*`"`

Is a color specification string for the data labels of subtotal risers. The default value is *undefined*.

`"numberFormat": `*`format`*

Is a number format string or JSON object that defines the number format of data labels for subtotal risers. For information, see *Introduction to JSON Properties for HTML5 Charts* on page 83.

The default value is *undefined*.

`"dataLabelPosition": "`*`string`*`"`

Is a string that defines the position of the data labels for subtotal risers. For information, see *Showing and Formatting Data Text Labels* on page 430.

The default value is *undefined*.

`"dividerLine"`

Defines the properties of an optional divider line that can be drawn before each subtotal riser.

`"width": `*`number`*

Is the width of the divider lines, in pixels. The default value is 0 (zero).

`"color": "`*`string`*`"`

Is a color specification string for the divider lines. The default value is "black".

`"dash": "`*`string`*`"`

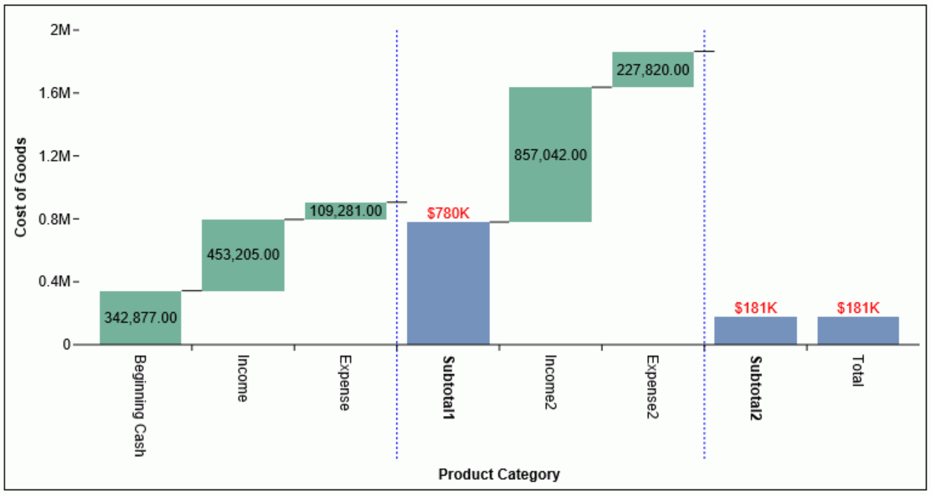
Is a string that defines the dash style of the divider lines. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, "dash": "1 1" draws a dotted line).

Example: Formatting Axis and Data Label Properties of Subtotal Risers

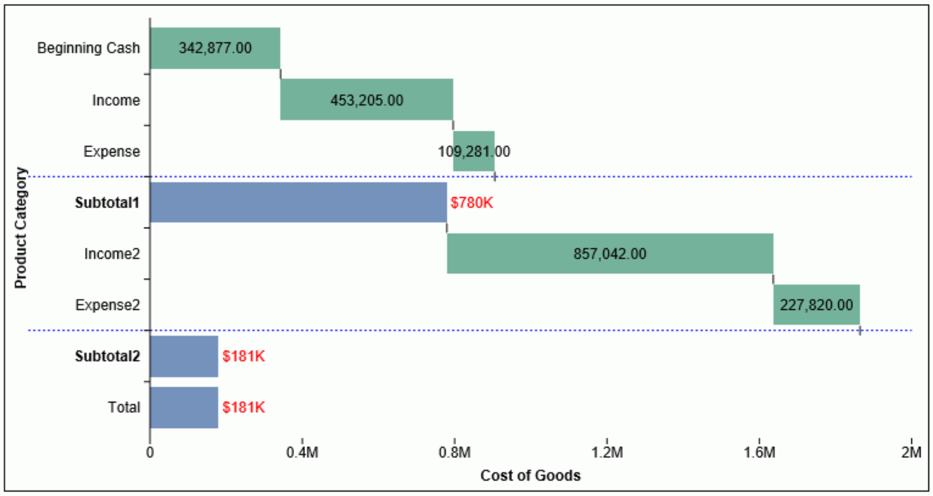
The following request makes the axis labels of subtotal risers bold, generates blue dashed divider lines, and formats the data labels to be red, to be placed above the risers, and to have a number format defined by a JSON object.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US

BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH WATERFALL
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [{"series":0, "dataLabels": {"visible":true}}],
"groupLabels": ["Beginning Cash", "Income", "Expense", "Subtotal1",
    "Income2", "Expense2", "Subtotal2", "Expense3", "TOTAL"],
"waterfallProperties":
{
"subtotalRisers": ["Subtotal1", "Subtotal2"],
"subtotalFormat":
   {
    "bold": true,
    "color": "red",
    "numberFormat": {"mode": "currency", "decimalPlaces": 0,
                     "grouping": "K"},
    "dataLabelPosition": "top",
    "dividerLine":
      {
       "width": 1,
       "color": "blue",
       "dash": "2 2"
       }
   }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Changing the LOOKGRAPH value to HWATERFL produces the following horizontal waterfall chart output.



Formatting the First and Last Risers in a Waterfall Chart

The firstLastFormat properties enable you to customize the formats of the first and last risers in a waterfall chart.

Syntax: How to Format the First and Last Risers in a Waterfall Chart

```
"firstLastFormat":
 {
   "color": color,
   "border":
     {
      "width": number,
      "color": "string",
      "dash": "string"
     }
 }
```

where:

`"color": color`

Specifies the color for the first and last risers. The default value is *undefined*, which does not distinguish the first and last risers from the other risers. Valid values are:

- ❑ A color string, enclosed in double quotation marks ("), that defines a color by name, by a numeric specification string, or by a gradient definition string.
- ❑ A JSON object that defines a gradient.

For information about specifying colors and gradients, see *Colors and Gradients* on page 85.

`"border"`

Defines the properties of the border of the first and last risers.

`"width": number`

Is the width of the border line in pixels. The default value is 0.

`"color": "string"`

Is a color name or numeric specification string. The default value is *undefined*.

`"dash": "string"`

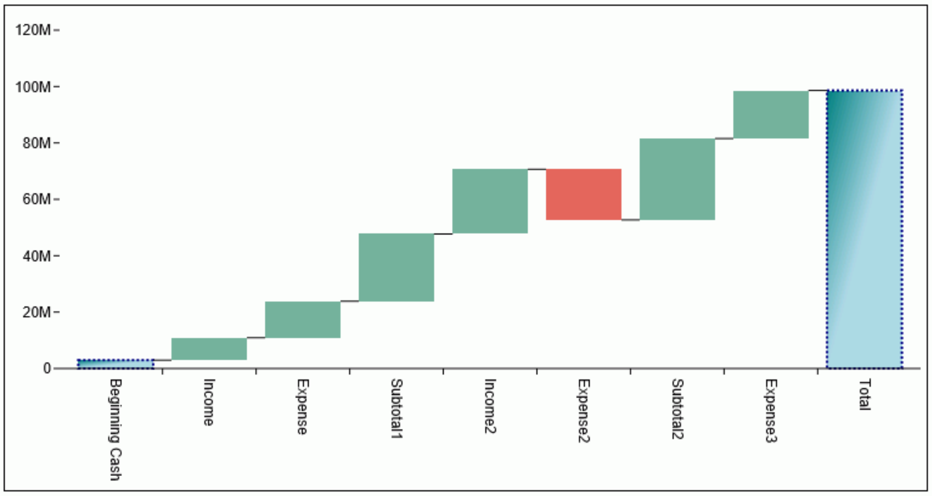
Is a string that defines the dash style of the border. The default value is "", which produces a solid line. Use a string of numbers that specifies the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line).

Example: Formatting the First and Last Risers

The following request formats the first and last risers have a gradient fill that transitions from teal to light blue, and to have a navy dashed border.

```
DEFINE FILE WF_RETAIL_LITE
INCR1 = COGS_US + 500;
INCR2 = COGS_US +1000;
INCR3 = COGS_US +INCR1 +INCR2;
INCR4 = COGS_US +2000;
DECR1 = -(INCR4- 500);
DECR2 = INCR3 +500;
DECR3 = -(COGS_US-2000);
END
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
INCR1 INCR2 INCR3 INCR4
DECR1 DECR2  DECR3
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VWATERFL
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": false},
"groupLabels": ["Beginning Cash", "Income", "Expense", "Subtotal1",
    "Income2", "Expense2", "Subtotal2", "Expense3", "TOTAL"],
"waterfallProperties":
{
  "firstLastFormat":
  {
   "color":
      {"type": "linear",
             "start": {"x": "0%", "y": "0%"},
             "end": {"x": "100%", "y": "100%"},
             "stops": [[0, "teal"], [0.5, "lightblue"]]},
    "border":
      {
       "width": 2,
       "color": "navy",
       "dash": "2 2"
      }
    }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Chapter 11

Generating Narrative Charts

Narrative charts use a text generation server to generate a natural language summary of and insight into a chart.

Natural Language Generation (NLG) is a branch of artificial intelligence (AI) that produces language as output on the basis of data input. This helps you to get an easily understandable explanation of a chart that adds value by identifying the most relevant information and conveying it through conversational language.

To generate a narrative chart, you add narrative chart properties to a chart request. Using these properties, you can control some of the features of the narrative that is generated.

In this chapter:

Configuring WebFOCUS to Generate Narrative Charts

In order to configure WebFOCUS to generate narrative charts, you must install the WebFOCUS Narrative Charts Server and, optionally, configure the URL to the WebFOCUS Narrative Charts Server in the WebFOCUS Administration Console.

Note: If you do not configure the URL to the WebFOCUS Narrative Charts Server in the WebFOCUS Administration Console, the Narrative charts button will not be activated in InfoAssist+. If you do configure the URL in the WebFOCUS Administration Console, you will find the *Narrative* button on the *Format* tab under *Features* when you create an HTML5 (FORMAT JSCHART) bar, line, area, or pie chart.

The WebFOCUS Narrative Charts Server is a separately licensed product that can be installed on Windows or Linux. Once you have licensed this feature, you can install the server using the instructions in the *WebFOCUS Narrative Charts Installation and Configuration* manual.

In order generate WebFOCUS narrative charts, you must either include the narrative text in your chart procedure or StyleSheet, configure the URL to the WebFOCUS Narrative Charts Server in the WebFOCUS Administration Console, or include the URL in the narrative chart properties in your chart procedure or StyleSheet. The order of precedence is:

1. **The URL configured in the WebFOCUS Administration Console.** If neither the content nor the url property is specified in your procedure or StyleSheet, the URL configured in the WebFOCUS Administration Console will be used.
2. **A URL specified in the procedure or StyleSheet.** If your procedure or the StyleSheet included in your procedure has a url property pointing to the WebFOCUS Narrative Charts Server, and the content property is omitted or set to *undefined*, the last URL found when parsing the procedure or StyleSheet will be used.
3. **Your own content.** If you supply narrative text in the content property of your procedure or StyleSheet, it will be used.

Reference: Usage Notes for Narrative Charts

Narrative charts can have a maximum of 1,024 data points. This limit is imposed by the WebFOCUS Narrative Charts Server and will be removed in a future release. If the limit is exceeded, the message returned from the WebFOCUS Narrative Charts Server is

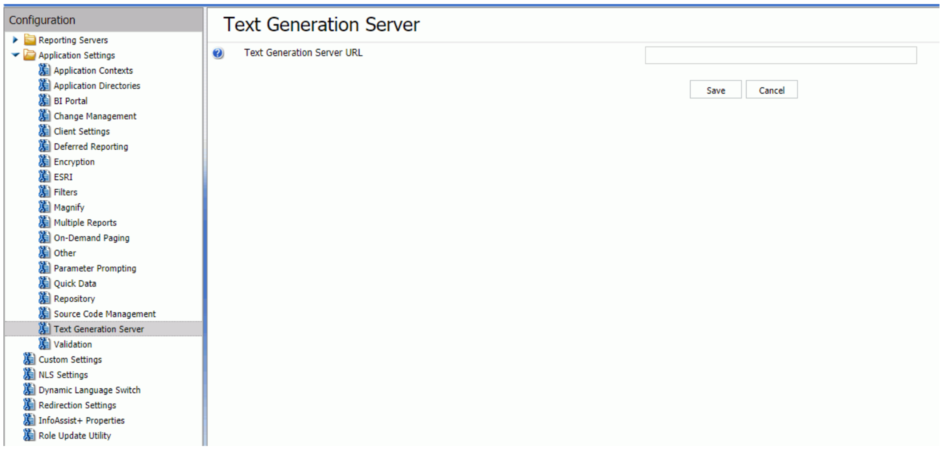
```
Failed to retrieve narrative text
```

Procedure: How to Configure the URL to the WebFOCUS Narrative Charts Server in the WebFOCUS Administration Console

1. From the WebFOCUS Home Page, click *Administration* at the top right of the page, and click *Administration Console*.

   The WebFOCUS Administration Console opens.

2. Select *Text Generation Server* under Application Settings in the left pane, as shown in the following image.



3. Enter the URL to the WebFOCUS Narrative Charts Server in the *Text Generation Server* URL text box.
4. Click *Save*.
5. Clear the cache in the WebFOCUS Administration Console by clicking *Clear Cache* at the top right of the console page.

Chart Types Supported for Narration

The following chart types are supported for narrative text generation when creating an HTML5 (FORMAT JSCHART) chart.

- **Bar.** Horizontal and vertical bars are supported, including stacked bars, and percent bars.
- **Line.** Horizontal and vertical lines are supported, including stacked lines and absolute lines.

  Note: Line charts require an ordered axis, such as date.
- **Area.** Horizontal and vertical area charts are supported, including stacked lines and absolute lines.

  Note: Area charts require an ordered axis, such as date.
- **Pie.** Both pie and ring pie charts are supported, including percent pies.

Narrative chart requests can include up to two dimensions as long as they are concatenated and if there is only one measure, or up to two measures if there is only one dimension.

Adding Narrative Chart Properties

Note: For JSON formatting, object names, property names, and string property values should be enclosed in double quotation marks.

For an introduction to JSON syntax for WebFOCUS HTML5 charts and the supported methods for specifying colors, see *Colors and Gradients* on page 85.

Syntax: How to Add Narrative Chart Properties to a Chart Request

The following properties can be used to add a narrative to a chart, format the narrative, and control features of the generated narrative.

```
*GRAPH_JS
"narrativeText":
   {
     "enabled": boolean,
     "hideChart": boolean,
     "position": "string",
     "maxSize": "string",
     "url": "string",
     "content": "string",
     "label": {
             "font": "string",
             "color": "string",
             "align": "string"
            },
     "backgroundColor": "string",
     "border": {
              "width": number,
              "color": "string",
              "dash": "string"
             },
     "dockButton": {
                  "enabled": boolean,
                  "animationDelay": number,
                  "size": number,
                  "color": "string",
                  "hoverColor": "string"
                },
  "textGenerationAPI":
            {
              "inputChart":
               {
                "mainDimensionOrdered": boolean,
                "multiSeriesType": "string",
                "disableInterpolation": boolean,
                "singularExceptions": ["string", "string"...],
                "pluralExceptions": ["string", "string"...],
               },
               "outputText": {
                           "levelOfDetail": number,
                           "lang": "string",
                           "useBulletPoints": boolean
                           },
              "measures":   [
                           {
                            "defaultValue": "string",
                            "meaningOfUp": "string",
                            "unit": "string"
                           },
                           {
                            "defaultValue": "string",
                            "meaningOfUp": "string",
                            "unit": "string"
                           }
                          ]
            }
   }
*END
```

where:

`"narrativeText":`

Starts the narrative chart properties section of a *GRAPH_JS block in the StyleSheet.

`"enabled":` *boolean*

Enables or disables the narrative text generation. Valid values are true, which generates a narrative on the chart output, and false, which does not generate a narrative on the chart output. The default value is false.

`"hideChart":` *boolean*

Controls whether the chart displays along with the narrative. Valid values are true, which displays only the narrative, not the chart, and false, which displays both the narrative and the chart. The default value is false.

`"position": "`*string*`"`

Defines the position of the narrative text in relation to the chart. Valid values are "top", "bottom", "left", or "right". The default value is "top".

`"maxSize": "`*`string`*`"`

Is a string that includes a percent symbol (%) that defines the maximum percent of the chart draw area that will be dedicated to the narrative text. This value takes position into account. If position is "left" or "right", this defines the maximum width of the text box for the narrative. If position is "top" or "bottom", this value defines the maximum height of the text box for the narrative. The default value is "20%".

`"url": "`*`string`*`"`

Identifies the URL to the text generation server. Optional if the URL is configured in the WebFOCUS Administration Console. Omit both the property name and value if you want to supply content using the *content* property.

`"content": "`*`string`*`"`

Should be *undefined*, if the text will be generated by the WebFOCUS Charts Narrative Server. If the url property is missing or commented out and the URL is not configured in the WebFOCUS Administration Console, provide a string containing the text of the narrative you want to display. Simple HTML tags such as < br > and < ol > can be used. An amper variable can be used if the Narrative Chart properties are defined in the chart request and not in an external WebFOCUS StyleSheet The default value is *undefined*.

Note: All of the content string must be entered as a single line in the procedure. Any breaks that you want to appear on the narrative output should be entered as HTML tags.

`"label":`

These properties define the format and alignment of the generated narrative text.

`"font": "`*`string`*`"`

Is a string that defines the font of the narrative text. The default value is "11pt Sans-Serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the narrative text using a color name, an RGB color string, or a hex color string. The default value is "black".

`"align": "`*`string`*`"`

Is a string that defines the alignment of the narrative text within the text box. Valid values are "left", "middle", "center", and "right". The default value is "left".

`"backgroundColor": "`*`string`*`"`

Is a string that defines the background color of the text using a color name, an RGB color string, or a hex color string. The default value is *undefined*.

`"border":`

These properties define the border of the text box.

`"width":` *`number`*

Is a number that defines the width in pixels of the border around the narrative text box. The value zero (0) does not draw a border around the text box. The default value is 0.

`"color": "`*`string`*`"`

Is a string that defines the color of the border around the text box using a color name, an RGB color string, or a hex color string. The default color is "black".

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the border around the text box. The default value is "" (a solid line). Use a string of numbers that defines the width of a dash in pixels followed by the width of the gap between dashes in pixels (for example, dash: "1 1" draws a dotted line).

`"dockButton":`

These properties define a button that can be drawn pointing to the chart that, when clicked, expands the narrative text box as much as required, up to the size of the entire chart frame. After expansion, the button becomes a collapse button that collapses the text box to its original size.

`"enabled":` *`boolean`*

Controls whether a dock button will be drawn. Valid values are true, which draws a dock button, and false, which does not draw a dock button. The default value is false.

`"animationDelay":` *`number`*

Is a number that defines the length of time (in ms) to animate the sliding transition when expanding and collapsing the text box. The default value is 300.

`"size":` *`number`*

Is a number that defines the diameter of the dock button in pixels. The default value is *undefined*.

`"color": "`*`string`*`"`

Is a string that defines the color of the dock button using a color name, an RGB color string, or a hex color string. The default value is "rgb(50, 50, 50)".

`"hoverColor": "`*`string`*`"`

Is a string that defines the color of the dock button when the mouse hovers over it. the default value is "black".

`"textGenerationAPI":`

These properties define the parameters sent to the WebFOCUS Narrative Charts Server to be used in generating the narrative text.

`"inputChart":`

These properties define aspects of the data being sent to the WebFOCUS Narrative Chart Server.

- `"mainDimensionOrdered":` *`boolean`*

  Specifies whether the dimension represents a range of values or a categorical list. To have the type of ordering automatically detected, omit this property. Valid values are:

  - true, which indicates that the dimension values can be expressed as a range of values. This type of ordering is appropriate for time, date, or numeric dimensions. The generated text will describe the dimension values as a from-to range of values. This is the default value for bar, line, and area charts.
  - false, which indicates that the dimension values do not represent a range. This type of ordering is appropriate for alphanumeric dimensions. The generated text will express the dimension values as a categorical list. This is the default value for pie charts.

- `"multiSeriesType": "`*`string`*`"`

  Indicates if the measures are related to each other. Valid values are:

  - "UNRELATED", which means the measures are not related to each other. This is the default value.
  - "VALUE_VS_REFERENCE", which means that the measures are related. Value refers to the first measure, say, Actual Sales, and the second measure is the Reference (that is, the target or objective). This option requires a date axis, and the format must be YYMD.

- `"disableInterpolation":` *`boolean`*

  Determines whether missing values will be approximated or replaced with a default value. Valid values are:

  - true, which uses a default value for missing values. The default value is specified in the defaultValue property for each measure. This is the default option.

- false, which computes an approximate value when a data value is missing.

- `"singularExceptions": ["`*`string`*`", "`*`string`*`" ...]`

  Is an array of strings that identify nouns that seem plural but should be treated as singular. Examples are the nouns mathematics, physics, and ethics. The default value is not to have singular exceptions.

- `"pluralExceptions": ["`*`string`*`", "`*`string`*`" ...]`

  Is an array of strings that identify nouns that seem singular but should be treated as plural. Examples are the nouns pliers, tongs, and tweezers. The default value is not to have plural exceptions.

`"outputText":`

These properties control some of the aspects of the returned narrative text.

- `"levelOfDetail":` *`number`*

  Is an integer from 1 to 7 that controls how much text is generated, with 7 generating the most detailed text and 1 generating the simplest text. The default value is 7.

- `"lang": "`*`string`*`"`

  Defines the language for the generated text. At this time, the supported values are "en" (English) or "fr" (French). You can be licensed to use either English or French, but not both. The default value is "en".

- `"useBulletPoints":` *`boolean`*

  Defines whether key points in the narrative should be displayed as bullets or paragraphs. Valid values are true, which displays key points as bullets, or false, which uses paragraphs. The default value is false.

`"measures":`

Is an array that defines the properties of up to two measures.

Note: If your chart request has only one measure, do not include properties for a second measure or the chart will not draw.

- `"defaultValue": "`*`string`*`"`

  Is a string that specifies default value to be used for missing data values if the disableInterpolation property is set to *true*. The default value is "0".

- `"meaningOfUp": "`*`string`*`"`

  Identifies whether high values represent positive information (for example, sales or profit) or negative information (for example, expenses). Valid values are:

  - "GOOD", which means higher values represent positive information. This is the default value.
  - "BAD", which means higher values represent negative information.

- `"unit": "`*`string`*`"`

  Defines the units to be applied to measure values. Valid values are:

  - "$". Should be used for currency values.
  - "%". When "%" is used, an automatic scale is applied to the data (.1 will be displayed as 10%).
  - "". This is the default value.

Note:

- For all examples, the application containing the Master File must be on the APP PATH or be included in the GRAPH FILE command. For example:

  ```
  GRAPH FILE app1/wf_retail_lite
  ```

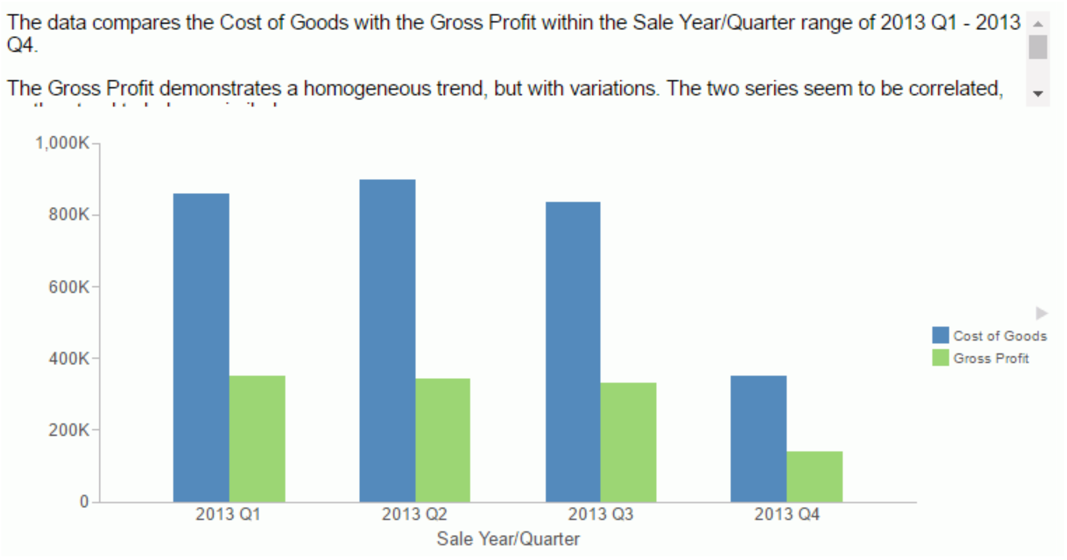
- The text generation server purposely creates varying narratives, even for the same chart output. Therefore, the narrative output you get if you run the following examples may differ from the narrative output displayed here.

Example: Generating a Bar Chart With Narrative Text

The following request generates a vertical bar chart with narrative text enabled. Default values are accepted for all other properties.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DATE_QTR_COMPONENT
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_DATE_QTR_COMPONENT, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
"textGenerationAPI": {
                     "measures": [
                                 {"unit": "$"},
                                 {"unit": "$"}
                                 ],
                     }
                }
*END
ENDSTYLE
END
```

The output is shown in the following image.


The data compares the Cost of Goods with the Gross Profit within the Sale Year/Quarter range of 2013 Q1 - 2013 Q4.

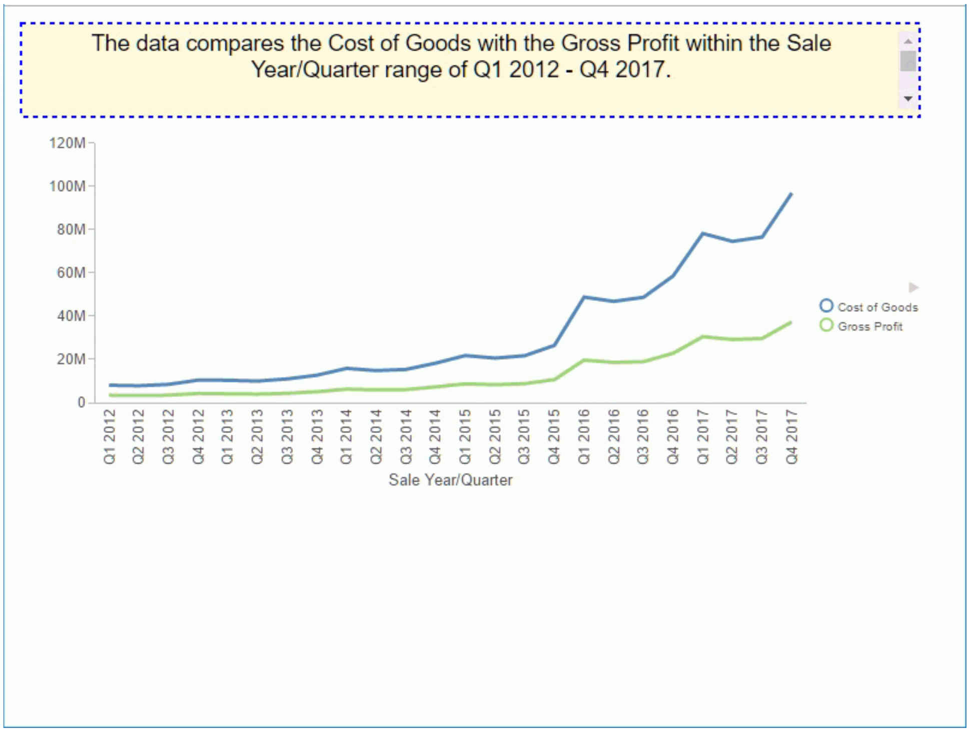
The Gross Profit demonstrates a homogeneous trend, but with variations. The two series seem to be correlated,

Example: Formatting the Narrative Text and Border

The following request generates a line chart and makes the text larger and center aligned, draws a blue dashed border around the text, and makes the background color corn silk.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DATE_QTR_COMPONENT
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH LINE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_DATE_QTR_COMPONENT, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
                "label": {
                                "font": "14pt Sans-Serif",
                                "color": "black",
                                "align": "center"
                      },
                "backgroundColor": "cornsilk",
                "border": {
                                "width": 2,
                                "color": "blue",
                                "dash": "4 4"
                        },
              }
*END
ENDSTYLE
END
```

The output is shown in the following image.

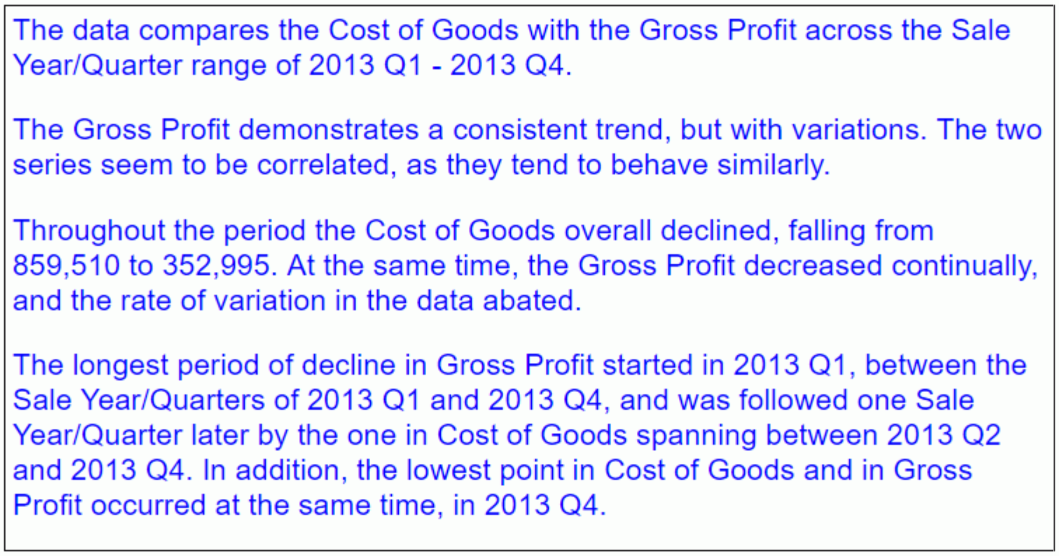

The data compares the Cost of Goods with the Gross Profit within the Sale Year/Quarter range of Q1 2012 - Q4 2017.

Example: Generating a Narrative-Only Chart

The following request hides the chart output so that only the narrative displays.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY TIME_DATE_QTR_COMPONENT
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_DATE_QTR_COMPONENT, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
                "hideChart": true,

                "label": {
                                "font": "16pt Sans-Serif",
                                "color": "blue",
                                "align": "left"
                      },

             }
*END
ENDSTYLE
END
```

The output is shown in the following image.



The data compares the Cost of Goods with the Gross Profit across the Sale Year/Quarter range of 2013 Q1 - 2013 Q4.

The Gross Profit demonstrates a consistent trend, but with variations. The two series seem to be correlated, as they tend to behave similarly.

Throughout the period the Cost of Goods overall declined, falling from 859,510 to 352,995. At the same time, the Gross Profit decreased continually, and the rate of variation in the data abated.

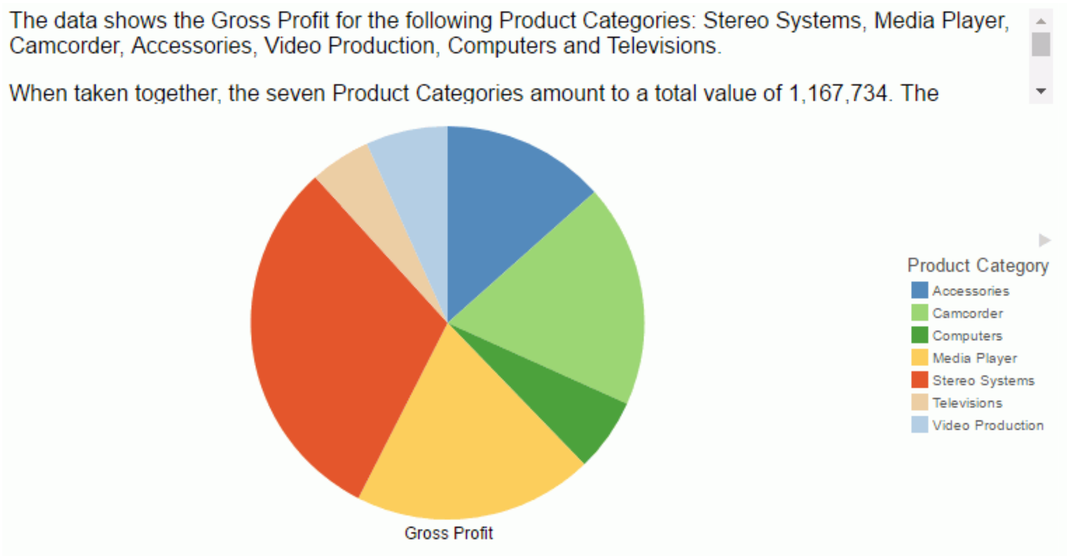
The longest period of decline in Gross Profit started in 2013 Q1, between the Sale Year/Quarters of 2013 Q1 and 2013 Q4, and was followed one Sale Year/Quarter later by the one in Cost of Goods spanning between 2013 Q2 and 2013 Q4. In addition, the lowest point in Cost of Goods and in Gross Profit occurred at the same time, in 2013 Q4.

Example: Generating a Narrative Using an Unordered Dimension

The following request generates a pie with a dimension (PRODUCT_CATEGORY) that should not be expressed as a range.

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=measure, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
                "label": {
                        "font": "12pt Sans-Serif",
                        "color": "black",
                        "align": "left"
                      },
                "textGenerationAPI":
                      {
                        "inputChart": {
                        "mainDimensionOrdered": false
                                    },
                      }
              }
*END
ENDSTYLE
END
```

The output is shown in the following image. The dimension values are expressed as a list, not a range.



The data shows the Gross Profit for the following Product Categories: Stereo Systems, Media Player, Camcorder, Accessories, Video Production, Computers and Televisions.

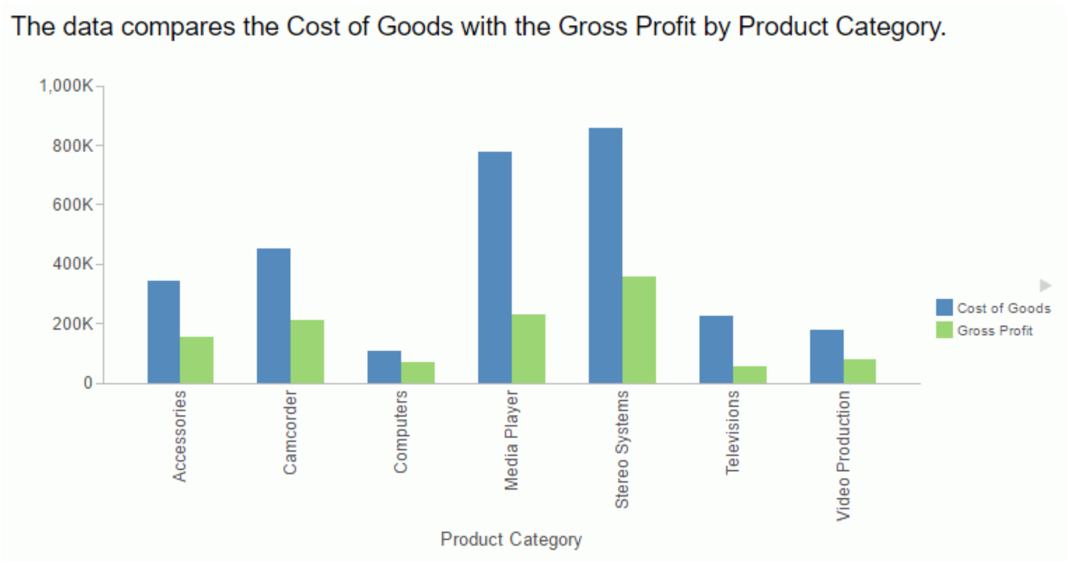
When taken together, the seven Product Categories amount to a total value of 1,167,734. The

Example: Generating a Narrative With Less Detail

The following request generates a bar chart with a less detailed narrative (levelOfDetail:1).

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
                "label": {
                        "font": "14pt Sans-Serif",
                        "color": "black",
                        "align": "left"
                      },
                "textGenerationAPI":
                      {
                       "outputText": {
                                    "levelOfDetail": 1
                                   },
                      }
              }
*END
ENDSTYLE
END
```

The output is shown in the following image.

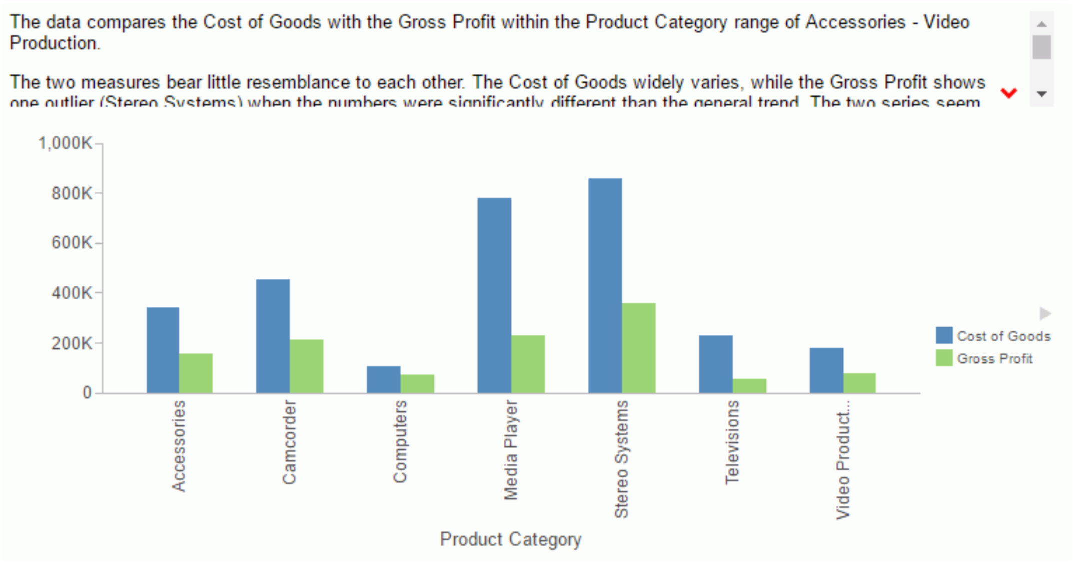


Example: Displaying a Dock Button

The following request generates a chart with a red dock button that has a diameter of 25 pixels.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText":
      {
       "enabled": true,
       "label": {
               "font": "10pt Sans-Serif",
               "color": "black",
               "align": "left"
              },
       "backgroundColor": "white",
       "dockButton":
               {
                "enabled": true,
                "size": 10,
                "color": "red"
               },
        }
*END
ENDSTYLE
END
```

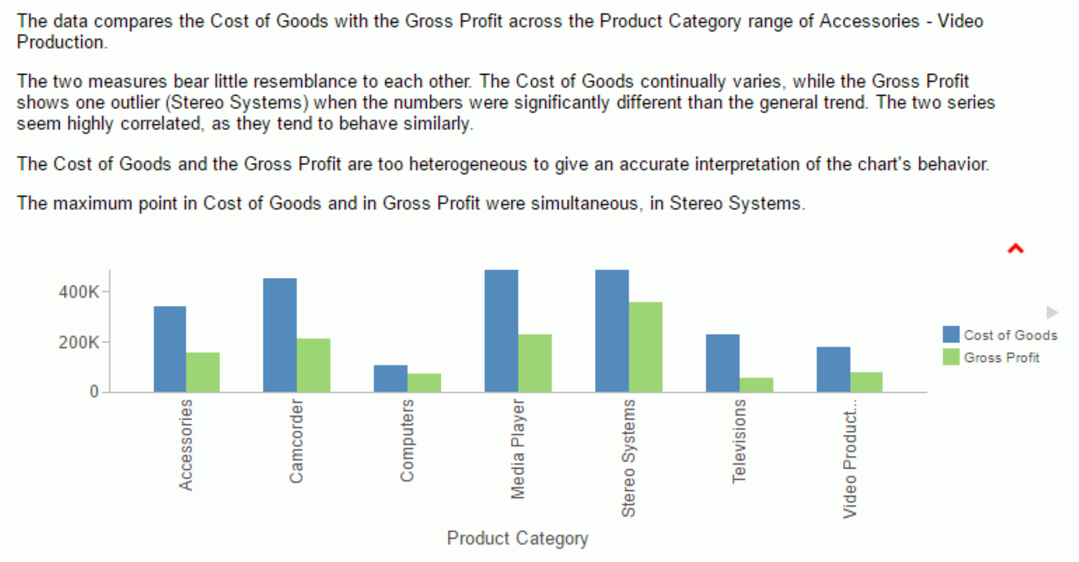
The initial output is shown in the following image, where the button is pointing toward the chart.



The data compares the Cost of Goods with the Gross Profit within the Product Category range of Accessories - Video Production.

The two measures bear little resemblance to each other. The Cost of Goods widely varies, while the Gross Profit shows one outlier (Stereo Systems) when the numbers were significantly different than the general trend. The two series seem

The following image shows the text box expanded after the dock button is clicked.



The data compares the Cost of Goods with the Gross Profit across the Product Category range of Accessories - Video Production.

The two measures bear little resemblance to each other. The Cost of Goods continually varies, while the Gross Profit shows one outlier (Stereo Systems) when the numbers were significantly different than the general trend. The two series seem highly correlated, as they tend to behave similarly.

The Cost of Goods and the Gross Profit are too heterogeneous to give an accurate interpretation of the chart's behavior.

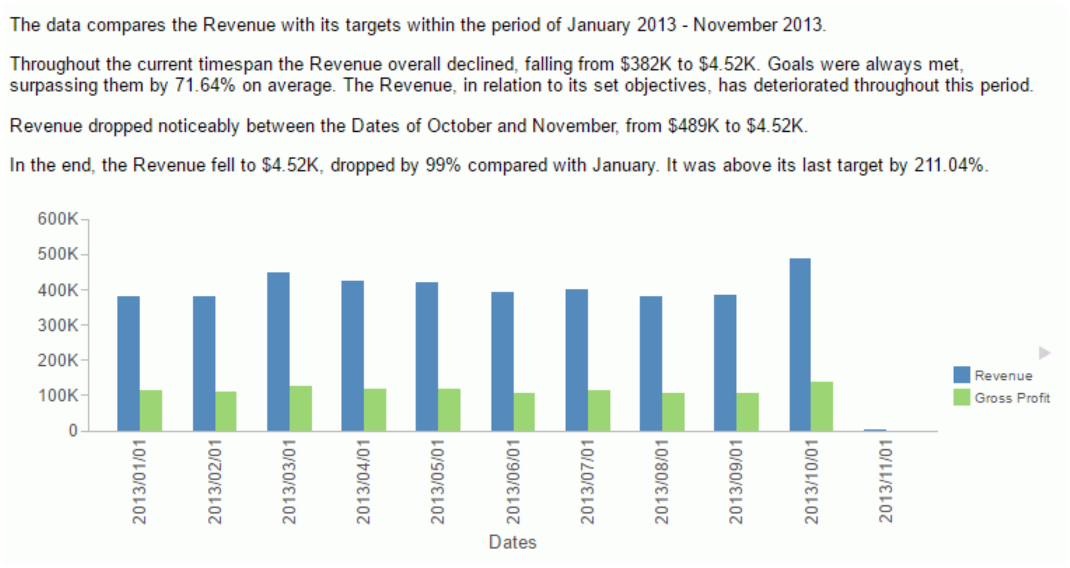
The maximum point in Cost of Goods and in Gross Profit were simultaneous, in Stereo Systems.

Example: Generating a Narrative for Related Measures

The following request describes revenue for the date period, using gross profit as its target values.

```
DEFINE FILE WF_RETAIL_LITE
Dates/YYMD=TIME_DATE_MONTH_COMPONENT;
END
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY Dates
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=REVENUE_US , BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=Dates, BUCKET=x-axis, $
*GRAPH_JS
  "narrativeText":
   {
    "enabled": true,
    "maxSize": "40%",
    "label": {"font": "10pt Arial"},
    "textGenerationAPI":
        {
         "inputChart": {
             "mainDimensionOrdered": true,
             "multiSeriesType":  "VALUE_VS_REFERENCE",
             "disableInterpolation": true,
                     },
         "measures":
            [
             {"unit": "$"},
             {"unit": "$"}
            ],
        }
      }
*END
ENDSTYLE
END
```

The output is shown in the following image.



The data compares the Revenue with its targets within the period of January 2013 - November 2013.

Throughout the current timespan the Revenue overall declined, falling from $382K to $4.52K. Goals were always met, surpassing them by 71.64% on average. The Revenue, in relation to its set objectives, has deteriorated throughout this period.

Revenue dropped noticeably between the Dates of October and November, from $489K to $4.52K.

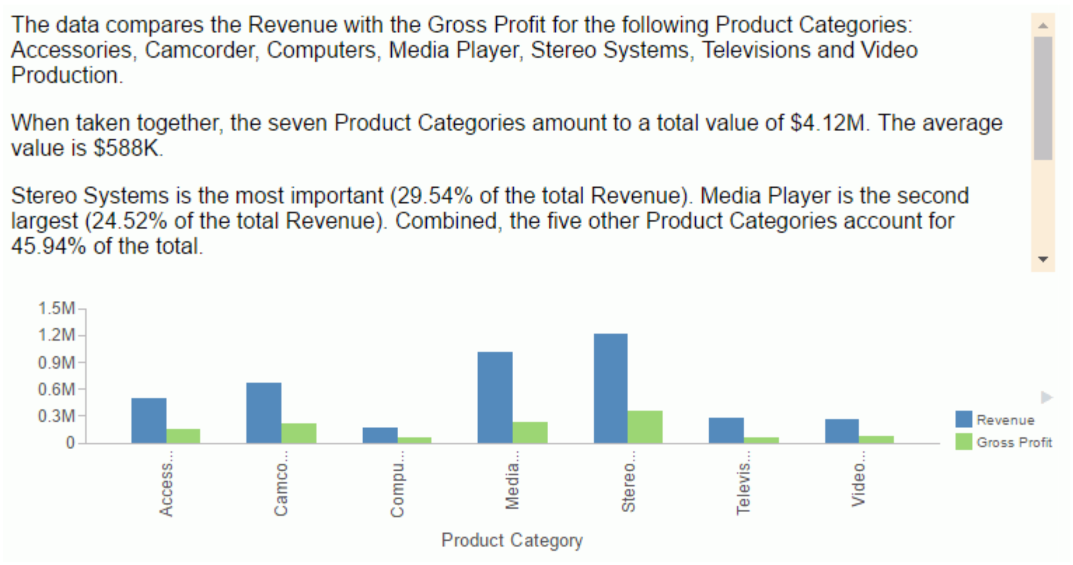
In the end, the Revenue fell to $4.52K, dropped by 99% compared with January. It was above its last target by 211.04%.

Example: Setting the Unit for Measures in the Narrative

The following request sets the dollar sign ($) as the unit for both measures and makes the text box larger.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=REVENUE_US , BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
  "narrativeText": {
    "enabled": true,
    "maxSize": "50%",
    "label": {"font": "12pt Arial"},
    "textGenerationAPI": {
        "inputChart": {
            "mainDimensionOrdered": false,
            "multiSeriesType":  "UNRELATED",
            "disableInterpolation": true,
                    },
        "measures": [
               {"unit": "$"},
               {"unit": "$"}
               ],
                       }
                 }
*END
ENDSTYLE
END
```

The output is shown in the following image.



The data compares the Revenue with the Gross Profit for the following Product Categories: Accessories, Camcorder, Computers, Media Player, Stereo Systems, Televisions and Video Production.

When taken together, the seven Product Categories amount to a total value of $4.12M. The average value is $588K.

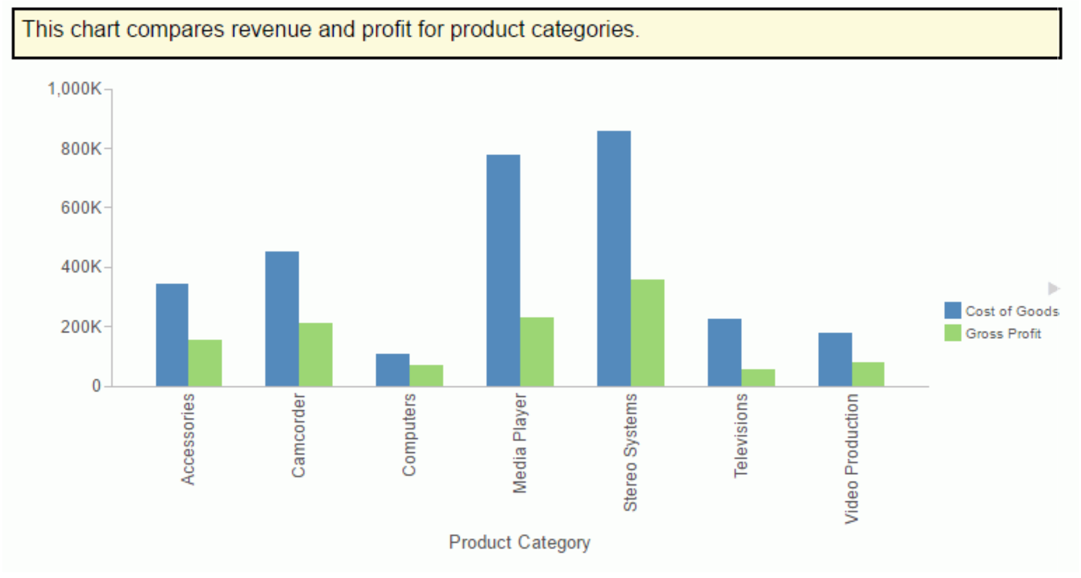
Stereo Systems is the most important (29.54% of the total Revenue). Media Player is the second largest (24.52% of the total Revenue). Combined, the five other Product Categories account for 45.94% of the total.

Example: Providing Your Own Narrative

The following request provides a narrative as text in the content property. Note that all content must be supplied as a single line in the request.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET=y-axis, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
*GRAPH_JS
"narrativeText":
  {
    "enabled": true,
    "content": "This chart compares revenue and profit for product categories.",
    "label": {
             "font": "12pt Sans-Serif",
             "color": "black",
             "align": "left"
           },
    "backgroundColor": "cornsilk",
    "border": {
              "width": 2,
              "color": "black",
              "dash": ""
            },
    "textGenerationAPI":
     {
             "inputChart":
             {
               " mainDimensionOrdered": false,
                "multiSeriesType": "UNRELATED",
             },

     }
  }
*END
ENDSTYLE
END
```

The output is shown in the following image.



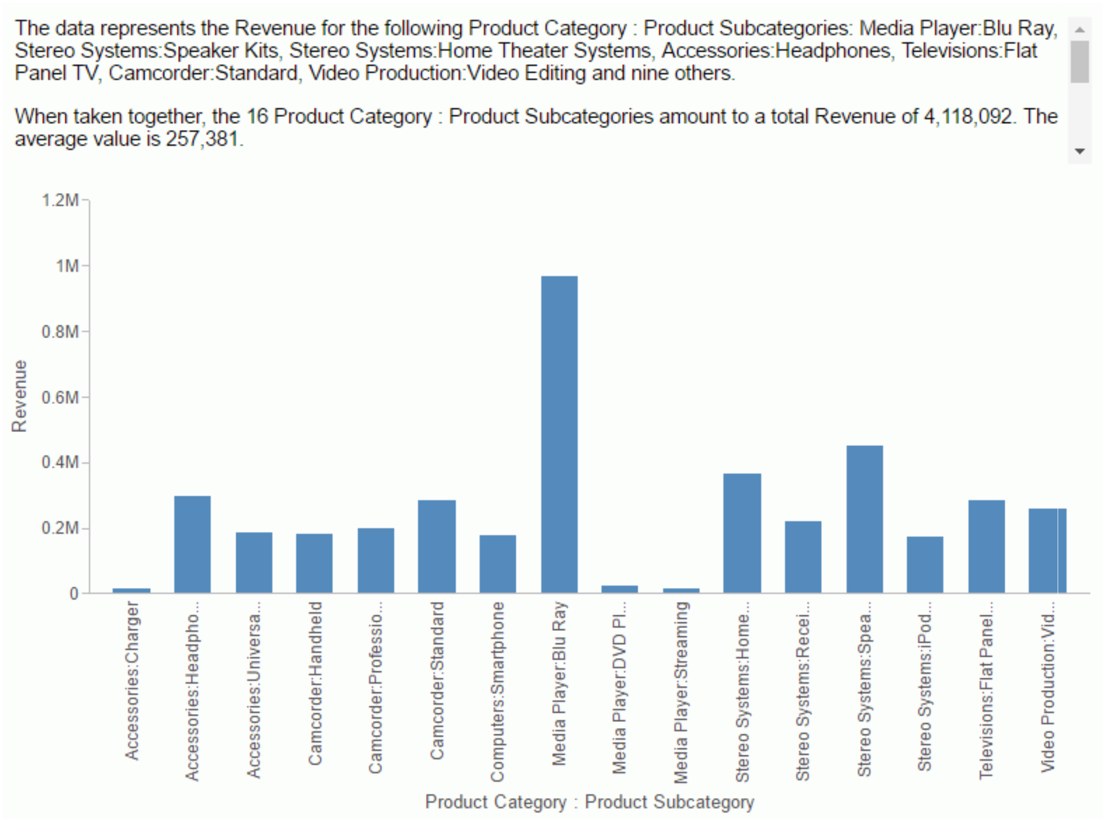
Example: Using Concatenated Dimensions

The following request uses two dimensions and one measure. The dimensions are required to be concatenated:

```
GRAPH FILE wfretail82/wf_retail_lite
SUM REVENUE_US
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, TITLETEXT='WebFOCUS Report', ORIENTATION=LANDSCAPE, $
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis, $
TYPE=DATA, COLUMN=PRODUCT_SUBCATEG, BUCKET=x-axis, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis, $
*GRAPH_JS
"narrativeText": {
                "enabled": true,
               "textGenerationAPI":
                      {
                        "inputChart": {
                        "mainDimensionOrdered": false
                                    },
                      }
                  },

"xaxis":{"labels":{"nestingConcatSymbol":":"}}
*END

ENDSTYLE
END

-RUN
```

Note: Concatenated labels are the default label option for the x-axis, and the default concatenation symbol is the colon (:). To concatenate dimensions manually, you can supply a concatenation symbol for the x-axis labels using the nestingConcatSymbol property, as shown in this example.

The output is shown in the following image.


The data represents the Revenue for the following Product Category : Product Subcategories: Media Player:Blu Ray, Stereo Systems:Speaker Kits, Stereo Systems:Home Theater Systems, Accessories:Headphones, Televisions:Flat Panel TV, Camcorder:Standard, Video Production:Video Editing and nine others.

When taken together, the 16 Product Category : Product Subcategories amount to a total Revenue of 4,118,092. The average value is 257,381.

Chapter 12

Special Topics

This chapter describes special features for HTML5 charts.

In this chapter:

Animation (introAnimation)

The introAnimation property enables or disables animation in drawing the chart and defines the duration of animation. When enabled, risers and markers are animated when the chart is drawn. Bar, line, and area risers enter the chart from the base of the frame and become static when they reach their assigned values. Circular risers (for example, bubble markers) are animated from the inside to the outer edge. Pie slices are drawn clockwise. Animation can be applied to all chart types except 3D charts and charts where 2.5D depth has been applied with the depth property.

Syntax: **How to Enable or Disable Animation When Drawing a Chart**

```
"introAnimation": {
   "enabled": boolean,
   "duration": number},
```

where:

`"enabled": boolean`

Enables or disables animation when drawing the chart. Valid values are:

- true, which animates the drawing of the chart.
- false, which does not animate the drawing of the chart. This is the default value.

`"duration": number`

Is a number that defines the duration of the animation in milliseconds. Larger numbers produce slower animation. The default value is 1000 (1 second).

Morph Animation Duration (morphAnimation)

When a user selects from multiple filters in Visualization mode, the risers and markers morph to show the transition to the new values generated by the filter. You can control the duration of this effect using the chart-wide morphAnimation:duration property.

Syntax: How to Control the Duration of Morph Animation

```
"morphAnimation": {
  "duration": number
}
```

where:

`"duration": number`

Is the length of time to spend animating the morph effect, measured in milliseconds. The default value is 1500.

Controlling the Speed of a Data Page Slider

When you add a sort field to the slider attribute category or, in the tools, add one to the animate category, a data page slider displays on the chart output. If you click the play button, the control moves along the slider bar with a default delay of 1000 milliseconds. You can increase or decrease this animation speed.

For complete information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

Syntax: How to Control the Speed of a Data Page Slider

```
"dataPageSlider": {
    "animateButton": {
      "delay": number                   }
                           },
```

where:

`"delay": number`

Is the delay in milliseconds with which the slider moves from frame to frame. The default value is 1000.

Using JSON to Register Chart Events

On an HTML5 chart, a DOM (Document Object Model) event, such as clicking an object on the chart, can be detected and used to trigger a response, such as invoking a callback function or going to a URL.

Syntax: How to Register Chart Events Using JSON

You can describe an array of events to the event dispatcher. Each event will trigger a corresponding response when the event occurs on the DOM nodes that match the properties listed in the event description.

```
"eventDispatcher": {
 "events" = [
  {response, event: 'string', object: 'string',
   userInfo: {custom: 'string'}, series: number, group: number,
   row: number, col: number,
   misc: 'string', userInfo: {custom: 'string'},
   context:  context, target:  'string' },
   .
   .
   .

 ]
}
```

where:

response, . . .

Required

Is the response to the event that matches the remaining parameters described by this entry in the array of events. Valid responses can be:

- ❑ Invoking a callback function.

  `callback: function(parm){function_code;}`

  where:

`function(parm){function_code;}`

Are the function parameters and the code invoked by the function.

- ❑ Going to a URL.

  `"url": "`*`link_to_url`*`"`

  where:

`"url": "`*`link_to_url`*`"`

  Is the URL to which you want to send the user.

`event: '`*`string`*`'`

Required

Is a DOM event string such as 'mouseover', 'click', or 'keyup', that triggers the response for this event in the array of events.

`object: '`*`string`*`'`

Required

Is a string matching the object ID of the chart object (such as 'riser') on which to register the event.

`series:` *`number`*

Optional

Is the series ID number of the riser series to match. If undefined, match all series.

`group:` *`number`*

Optional

Is the group ID number of the riser groups to match. If undefined, match all groups.

`row:` *`number`*

Optional

Is the matrix row ID to match. If undefined, match all rows.

`col:` *`number`*

Optional

Is the matrix column ID to match. If undefined, match all columns.

`misc: '`*`string`*`'`

Optional

Is a miscellaneous object ID to match, for example, 'marker'. If undefined, match all misc IDs.

`userInfo: {custom: '`*`string`*`'}`

Optional

Is an extra, custom user-defined object to be passed back into the event callback.

`context:` *`context`*

Optional

Defines 'this' in the event callback. If undefined, 'this' refers to the calling chart instance.

`target: '`*`string`*`'`

Optional

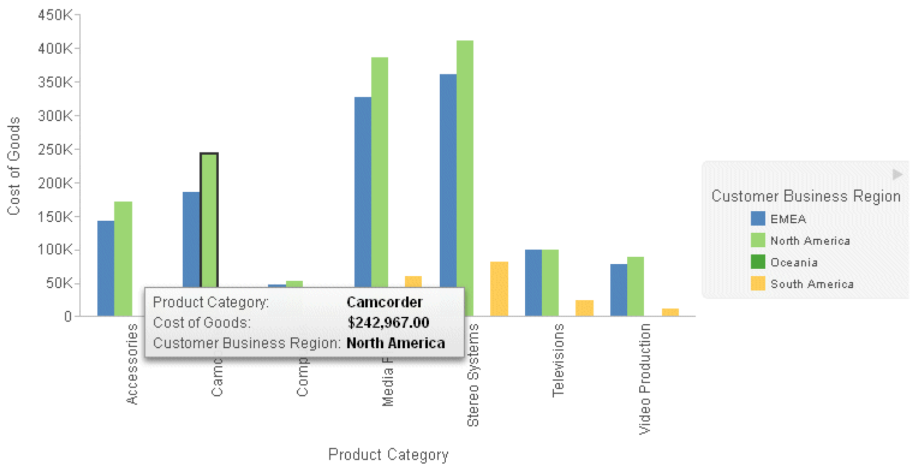
Is the target associated with a "url" property, used when triggering a URL click event. The default is '_blank', which instructs the browser to create a new browser tab or window when the user clicks the link.

Example: Registering Chart Events

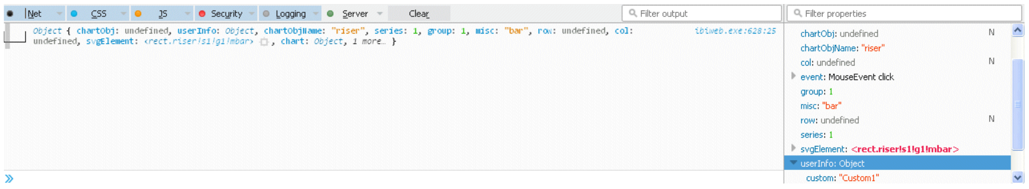
The following request registers two events. The first event logs the event parameters to the browser Web Console when the user clicks the riser for series 1 in group 1. The second event goes to the Information Builders web site when the user clicks a riser for series 2.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis,$
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=color,$
TYPE=DATA, COLUMN=cogs_us, BUCKET=y-axis,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"eventDispatcher": {
"events": [
    {callback: function(a){console.log(a);},
   event: 'click', object: 'riser',
   userInfo: {custom: 'Custom1'}, series: 1, group:1},
 {"url": "http://www.informationbuilders.com", target: '_blank',
    event: 'click', object: 'riser', series: 2},
    ]
}
*END
ENDSTYLE
END
```

The following image shows the chart output with the riser for series 1 and group 1 clicked.



The following shows the parameters logged to the browser Web Console in response to the click event.



```
Object { chartObj: undefined, userInfo: Object, chartObjName: "riser", series: 1, group: 1, misc: "bar", row: undefined, col:
undefined, svgElement: <rect.riser!s1!g1!mbar> , chart: Object, 1 more… }                                  ibiweb.exe:628:25
```

```
chartObj: undefined          N
chartObjName: "riser"
col: undefined               N
event: MouseEvent click
group: 1
misc: "bar"
row: undefined               N
series: 1
svgElement: <rect.riser!s1!g1!mbar>
userInfo: Object
  custom: "Custom1"
```

Annotations on Charts

An annotation is text, a marker, or both that you can place anywhere on a chart. Multiple annotations can be defined within the annotations array.

Chart annotations support the ability to specify an arbitrary SVG object and define events and tooltips for the SVG object, enabling you to create your own interactive buttons and icons on the chart canvas.

You have to add a separate eventDispatcher object to the request in order to add interactivity to an annotation. For information, see *Using JSON to Register Chart Events* on page 741.

The following shows the properties and default values for annotations:

```
"annotations": [
 "position": {
   "x": undefined,
   "y": undefined,
   "points": undefined,
   "parent": "background"
        },
 "marker": {
   "visible": true,
   "color": "red",
   "size": 8,
   "shape": "circle",
   "rotation": 0,
   "border": {
     "width": 1,
     "color": "black",
     "dash": ""
           }
     "tooltip": undefined
        },
 "line": {
              "width": 1,
              "color": undefined,
              "dash: ""
            },
 "label": {
    "visible": false,
    "text": "Annotation Label",
    "font": "10pt Sans-Serif",
    "color": "black",
    "position": "center"
     }
 },
]
```

Syntax: How to Place an Annotation on a Chart

```
"annotations": [
 "position": {
   "x": x,   "y": y,
   "points": [{"x":x, "y":y},...},
   "parent": "string"
     },
 "marker": {
   "visible": boolean,
   "color": "string",
   "size": number,
   "shape": "string",
   "position": "string",
   "rotation": number,
   "border": {
     "width": number,
     "color": "string",
     "dash": "string"
         }
     "tooltip": "string"
     },
 "line": {
               "width": number,
               "color": "string",
               "dash": ""
            },
 "label": {
         "visible": boolean,
         "text": "string",
         "font": "string",
         "color": "string",
         "position": "string"
     }
 },
   .
   .
   .
]
```

where:

`"position":`

Defines the position of the annotation marker.

`"x":` *`x`*

Is the horizontal position for the annotation icon. A synonym for "x" is "left", which measures the horizontal position from the left edge of the frame. You can also use "right", which measures the position from the right edge of the frame.

The default value is *undefined*. Can be:

- A number. If the parent chart component is "chart", the number represents a value on a numeric axis. Otherwise, it represents a number of pixels from the left corner of the parent component for "x" or "left", or the right corner for "right".
- A series and group label for a horizontal chart (in the form {series:*n*,group:*m*}), which places the annotation on that series or group. This requires the parent component to be "chart".
- A percent string (for example, "40%"), which places the annotation at a point the specified percentage horizontally from the left corner of the parent component for "x" or "left", or the right corner for "right".
- A pixel string (for example, "40px"), which places the annotation at a point that number of pixels horizontally from the left corner of the parent component for "x" or "left", or the right corner for "right".

If you set multiple horizontal position properties, "x" takes precedence over either "right" or "left", and "left" takes precedence over "right".

`"y":` *`y`*

Is the vertical position for the annotation icon. A synonym for "y" is "bottom", which measures the horizontal position from the bottom edge of the frame. You can also use "top", which measures the position from the top edge of the frame.

The default value is *undefined*. Can be:

- A number. If the parent chart component is "chart", the number represents a value on a numeric axis. Otherwise, it represents a number of pixels from the bottom edge of the parent component for "y" or "bottom", or the top edge for "top".
- A series and group label for a vertical chart (in the form {series:*n*,group:*m*}), which places the annotation on that series or group. This requires the parent component to be "chart".

- A percent string (for example, "40%"), which places the annotation at a point the specified percentage vertically from the bottom edge of the parent component for "y" or "bottom", or the top edge for "top".
- A pixel string (for example, "40px"), which places the annotation at a point that number of pixels vertically from the bottom edge of the parent component for "y" or "bottom", or the top edge for "top".

If you set multiple vertical position properties, "y" takes precedence over either "bottom" or "top", and "bottom" takes precedence over "top".

`"points": [{"x":`*`x`*`, "y":`*`y`*`},...]`

Is an array of {x, y} points, used to draw arbitrary lines or shapes. Only used if "position": "x" and "y" are undefined.

`"parent": "`*`string`*`"`

Can be one of the following chart components:

- <u>"background"</u>, which measures the position from the lower left corner of the chart frame and places the annotation behind the chart. This is the default value.
- "chart", which measures the position from the lower left corner of the chart draw area and places the annotation on the chart.
- "legend", which measures the position from the lower left corner of the legend area and places the annotation in the legend area.
- "xaxis", which measures the position along the x-axis.
- "yaxis", which measures the position along the y-axis.

`"marker":`

Defines the properties of the annotation marker.

`"visible":` *`boolean`*

Defines whether the annotation marker is visible. Valid values are:

- <u>true</u>, which makes the annotation marker visible. This is the default value.
- false, which makes the annotation marker not visible.

`"color": "`*`string`*`"`

Defines the color for the annotation marker, defined by a color name or numeric specification string, or a gradient defined by a string. The default value is "red".

`"size": number`

Is the size, in pixels, of the diameter of the annotation marker. This is ignored if "shape" represents an SVG path. The default value is 8.

`"shape": "string"`

Is a shape for the annotation marker. All valid marker shapes are supported.

One marker shape, "axisArrow", is specific to annotations anchored to the parent "xaxis" or "yaxis". It draws an arrow along the axis, which will automatically be rotated to point in the positive direction of the axis. The axisArrow color and style are set using the border properties. If the axisArrow marker does not include a border, the relevant axis body line style is used instead

The marker can also be an SVG path data string ('d' attribute) that defines the icon shape and size. The default shape is "circle".

For a list of marker shapes, see *Series-Specific Properties* on page 413.

`"position": "string"`

Defines the marker position relative to data text labels for bullet charts as "top", "bottom", or "middle", Not supported for other types of charts.

`"rotation": number`

Is a number between 0 and 360 that defines the rotation angle of the marker, in degrees.

`"border":`

Defines the properties of the marker border.

`"width": number`

Is a number that defines the width of the border in pixels. The default value is 1.

`"color": "string"`

Is a color for the marker border defined by a color name or numeric specification string. The default value is "black".

`"dash": "string"`

Is a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes. The default is no dash ("").

`tooltip`

Is a tooltip for the annotation marker. The default value is undefined.

`"line":`

Defines the properties of the line drawn if points are specified instead of one x and y value. This is only used if "position" includes multiple points. These properties will specify the line style used to connect those points.

`"width":` *`number`*

Is the width of the line in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a string that specifies the color of the line. The default value is undefined.

`"dash": "`*`string`*`"`

Is a string that defines the line dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes. The default is no dash ("").

`"label":`

Defines the properties of the annotation label.

`"visible":` *`boolean`*

Defines whether the annotation label is visible. Valid values are:

- true, which makes the annotation label visible. This is the default value.
- false, which makes the annotation label not visible.

`"text": "`*`string`*`"`

Defines the annotation label text. The default value is no label text.

`"font": "`*`string`*`"`

Defines the annotation label font. The default value is "10pt Sans-Serif".

`"color": "`*`string`*`"`

Defines the annotation label color defined by a color name or numeric specification string. The default value is "black".

`"position": "`*`string`*`"`

Defines a position for the annotation label relative to the annotation marker. Valid values are:

- "top", which places the annotation label above the annotation marker.
- "bottom", which places the annotation label below the annotation marker.
- "left", which places the annotation label to the left of the annotation marker.
- "right", which places the annotation label to the right of the annotation marker.

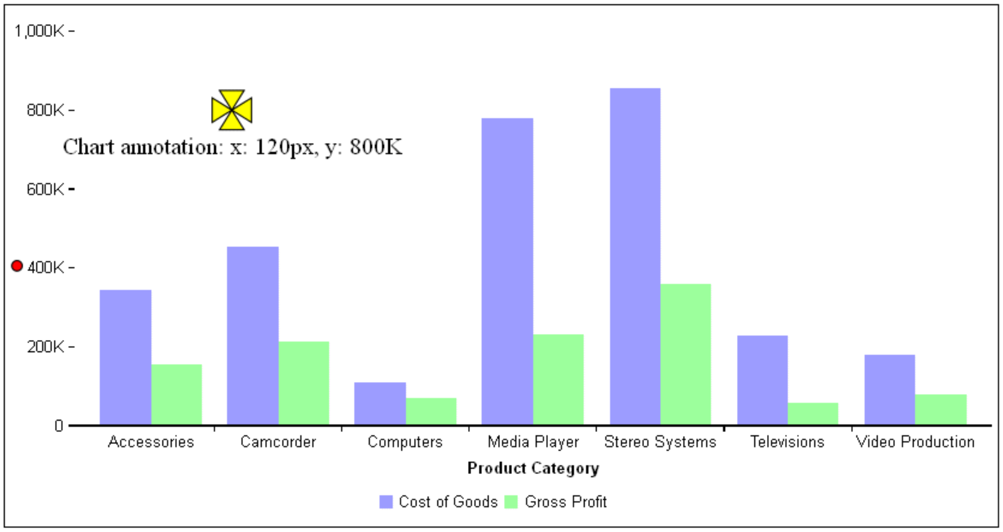
- "center", which places the annotation label centered on the annotation marker. This is the default value.

Example: Adding Annotations

The following request specifies two annotations. The first annotation is on the chart component. Its marker is a yellow pirate cross with a black border that is 120 pixels horizontally from the left edge of the x-axis and at the 800K point along the y-axis (which is numeric). The label is positioned below the marker. The second annotation only has a position (representing pixels), so there is no label, and it has the default marker, a red circle that is 8 pixels in size.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *

*GRAPH_JS
"annotations": [
    {"position": {"x": "120px", "y": 800000, "parent": "chart"},
        "marker": {"visible": true, "color": "yellow",
            "size": 30, "shape": "piratecross", "rotation": 0,
            "border": {"width": 1, "color": "black", "dash": ""}},
        "label": {"visible": true, "text": "Chart annotation: x: 120px, y: 800K",
            "font": "14pt Times New Roman", "color": "black", "position": "bottom"}},
        {"position": {"x": 10, "y": 200}}]
*END
ENDSTYLE
END
```

The output is:

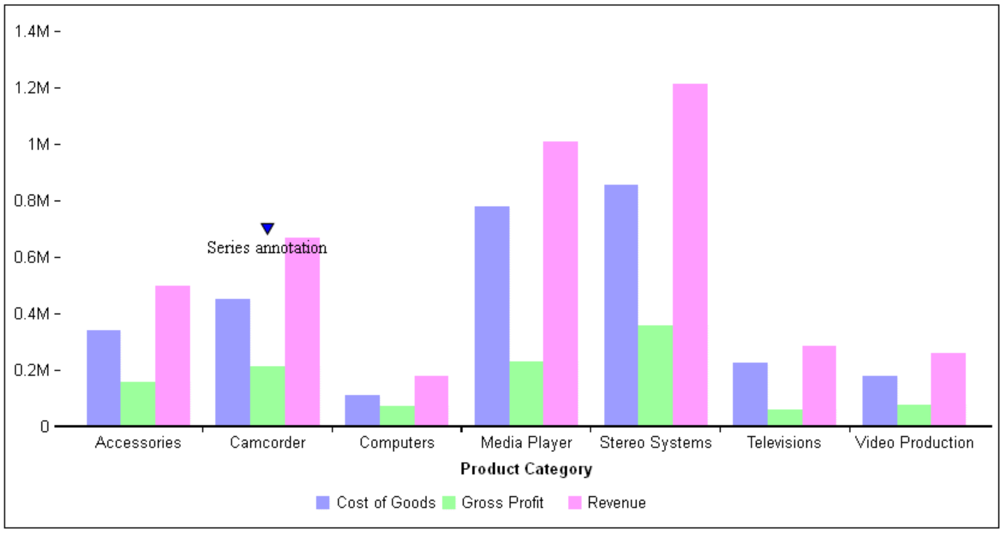


Example: **Placing Annotations Relative to a Series and Group**

The following request has one annotation, on series 1 and group 1. The y value is at 50% along the y-axis. The marker is a blue triangle.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"annotations": [{
    "position": {"x": {"series": 1, "group": 1},
        "y": "50%", "parent": "chart"},
    "marker": {"visible": true, "color": "blue",
        "shape": "triangle", "rotation": 0,
        "border": {"width": 1, "color": "black", "dash": ""}},
    "label": {"visible": true, "text": "Series annotation",
        "font": "10pt Times New Roman", "color": "black", "position":
"bottom"}
    }]
*END
ENDSTYLE
END
```

The output is:

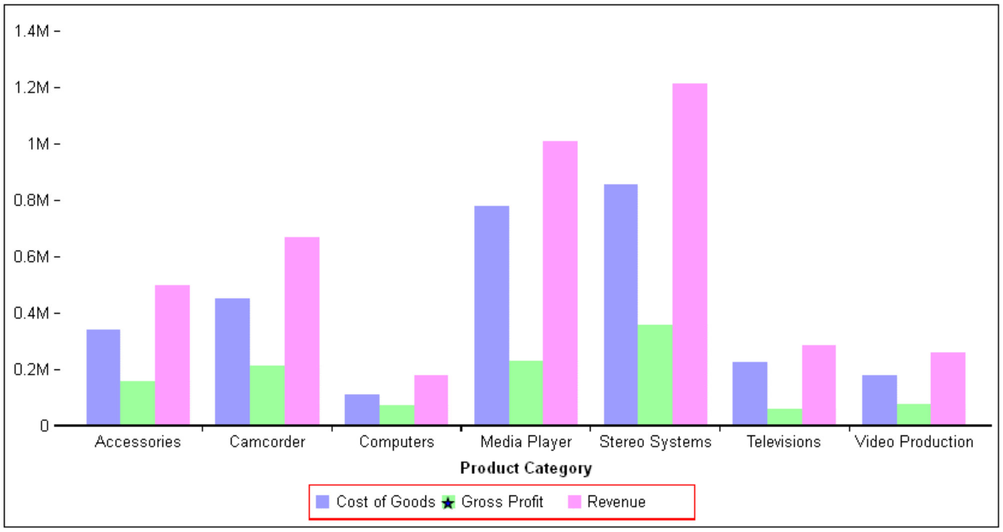


Example: **Placing an Annotation in the Legend Area**

The following request places a blue star over the green legend marker in the legend area.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"lineStyle": {"color": "red"}},
"annotations": [{
    "position": {
        "x": "106px",
        "y": ".14px",
        "parent": "legend"},
    "marker": {"visible": true, "color": "blue",
        "shape": "fiveStar", "rotation": 0,
        "border": {"width": 1, "color": "black", "dash": ""}}
    }]
*END
ENDSTYLE
END
```

The output is:

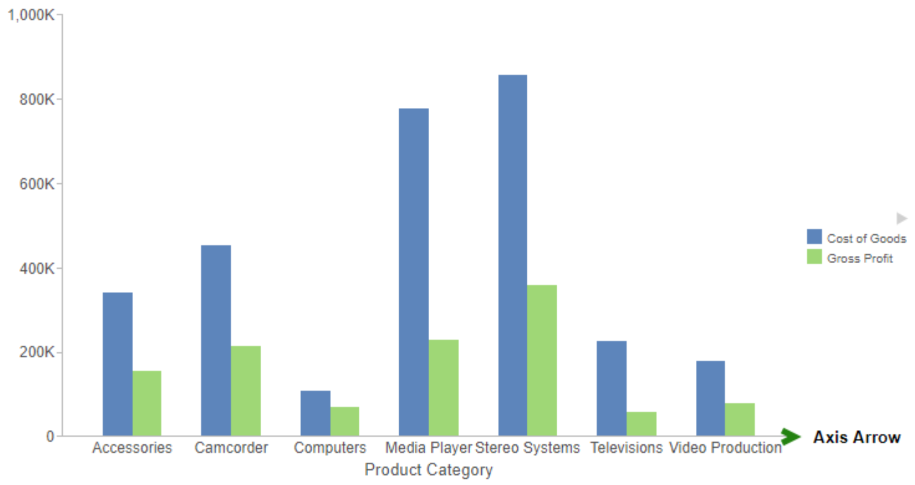


Example: Using an Axis Annotation Anchor

The following request anchors the annotation on the x-axis. The annotation marker is a 20-pixel green axisArrow marker with a 5-pixel wide border, which is placed at the end of the x-axis. The label text *Axis Arrow* is positioned to the right of the marker.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=gross_profit_us, bucket=y-axis,$
type=data, column=n1, bucket=x-axis,$
*GRAPH_JS
"annotations": [{
  "position": {
     "x": "100%",
     "parent": "xaxis"},
  "marker":
    {"visible": true,
    "shape": "axisArrow", "rotation": 0,
    "size": 20,
    "border": {"width": 5, "color": "green", "dash": ""}
},
  "label": {"visible": true, "text": "Axis Arrow",
    "font": "bold 10pt Arial", "color": "black", "position": "right"},
}]
*END
ENDSTYLE
END
```

The output is shown in the following image.



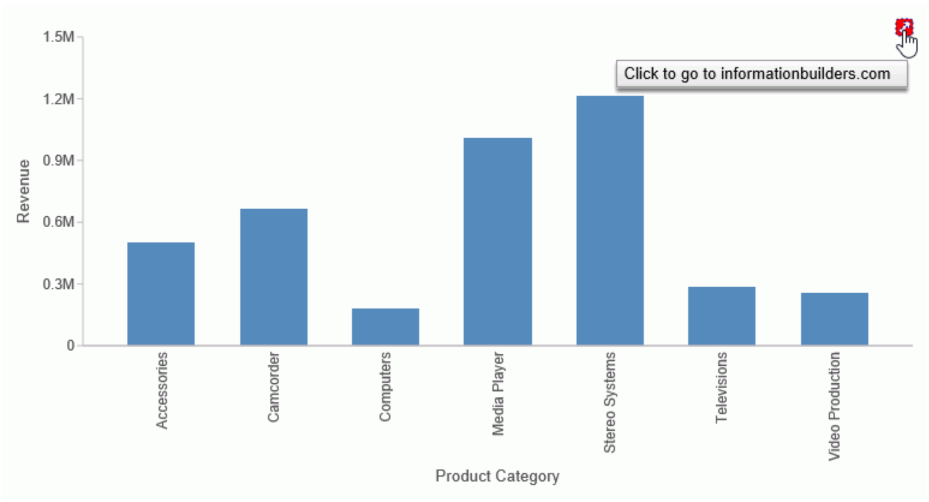
Example: **Defining an Interactive Annotation for a Chart**

The following request creates an annotation that consists of a red icon with a navy dashed border. The event associated with clicking this icon opens the Information Builders home page. The tooltip displays the following text:

```
Click to go to informationbuilders.com
```

```
SET PAGE-NUM=NOLEAD
SET HTMLENCODE=OFF
SET EMBEDHEADING=ON
GRAPH FILE wf_retail_lite
SUM REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, $
*GRAPH_JS
   "eventDispatcher": {
        "events": [
            {
                // This will add a click  event handler to the annotation
                "event": "click", object: "annotation", series: 0, url:
"http://informationbuilders.com", target: "_blank"
            }
        ]
    },

annotations:[{
      "position": {
            "x": 750,
            "y": 12,
        "parent": "background"
    },
    "marker": {
    "visible": true,
    "color": "red",
    "border": {"width": 1, "color": "navy", "dash": "2 2"},
       "shape": "m 12.666667,8.291667 0,-4.375 c 0,-0.319084
-0.26425,-0.583334 -0.583334,-0.583334 l -4.3749997,0 C 7.4715,3.333333
7.2615,3.479167 7.1705,3.688583 7.0795,3.907333 7.125,4.16225
7.29825,4.32675 l 1.3125,1.3125 -4.8673333,4.867333 c -0.2280834,0.228084
-0.2280834,0.592667 0,0.820167 L 4.67325,12.256583 c 0.2280833,0.228084
0.5926667,0.228084 0.8201667,0 L 10.36075,7.38925 l 1.3125,1.3125 c
0.109083,0.118417 0.2555,0.17325 0.410083,0.17325 0.07292,0
0.155167,-0.01808 0.228084,-0.0455 0.209416,-0.091 0.35525,-0.301
0.35525,-0.537833 z M 15,3.625 l 0,8.75 C 15,13.824 13.824,15 12.375,15 l
-8.75,0 C 2.176,15 1,13.824 1,12.375 L 1,3.625 C 1,2.176 2.176,1 3.625,1 l
8.75,0 C 13.824,1 15,2.176 15,3.625 Z",
        "tooltip": "Click to go to informationbuilders.com"
    }
}]
*END
 ENDSTYLE
END
```

The output is shown in the following image.



Color Modes (colorMode)

The colorMode property defines the color mode for drawing chart risers and markers. In the default configuration (bySeries), risers and markers in a series are colored according to the values specified by the series:color property. The default values are series: 0, color: red, series: 1, color: green, series: 2, color: orange. This property can be used to apply different color modes to the chart risers and markers. Some color modes are not applicable or sensible for some chart types. For example, the waterfall charts properties define colors for positive risers and negative risers.

Syntax: How to Define Color Modes

```
"colorMode": {
   "mode": "string",
   "colorList": ["string",..."string"]
 },
```

where:

`"mode": "string"`

Is a string that defines the chart color mode. Valid values are:

- "byGroup", in which risers and markers in a group a colored according to values specified in the series:color property.

- "byHeight", in which the colors defined in the colorList array create a gradient such that the first color is at the numeric axis minimum, the last color is at the numeric axis maximum, and the remaining colors are interpolated between these. This gradient is then used to color the risers. The byHeight mode is only valid for charts with an ordinal axis and a single numeric axis.
- "byInterpolation", in which the first series is colored the first color in the colorList array, the last series is colored the last color, and the series in between are interpolated accordingly.
- "byInterpolationAlt", which is identical to byInterpolation, except it does one additional step of mixing up the order of the chosen series colors. This mode is useful because colors between adjacent risers are different and easier to distinguish.
- "bySeries", in which risers and markers in a series are colored according to the values specified by the series:color property. This is the default, except for pie charts.
- "continuous", which is the default color mode for heatmaps and choropleths. This mode blends the colors in the color scale to form a gradient.
- "discrete", which can be used for heatmaps and choropleths to visualize the colorScale in discrete color bands. You use this with the colorScale object to define an array of color bands that provide start and stop values for each color. For more information, see *Defining a Color Scale Color Mode* on page 771.
- "bin", which can be used for choropleths and bubblemaps to visualize the legend for the color bands using markers. You use this with the colorScale object to define an array of color bands that provide start and stop values for each color. For more information, see *Defining a Color Scale Color Mode* on page 771.

`"colorList": ["`*`string`*`",..."`*`string`*`"]`

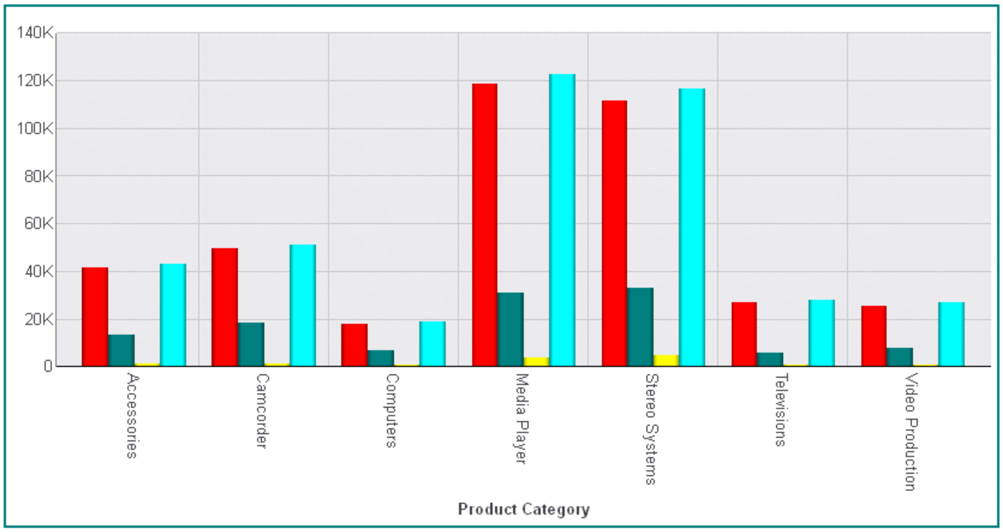
For byHeight, byInterpolation, and byInterpolationAlt color modes, is an array of colors defined by a color name or numeric specification string. If only one color is specified for byHeight, byInterporation, or byInterpolationAlt, a lighter version of that color is used as an implicit second color. The default value is undefined.

Example: Setting Color Modes

The following request uses the default color mode (bySeries):

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"legend": {"visible": false},
"series": [
    {"series": "all", "border": {"width": 1, "color": "grey"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "teal"},
    {"series": 2, "color": "yellow"},
    {"series": 3, "color": "cyan"}]
*END
ENDSTYLE
END
```

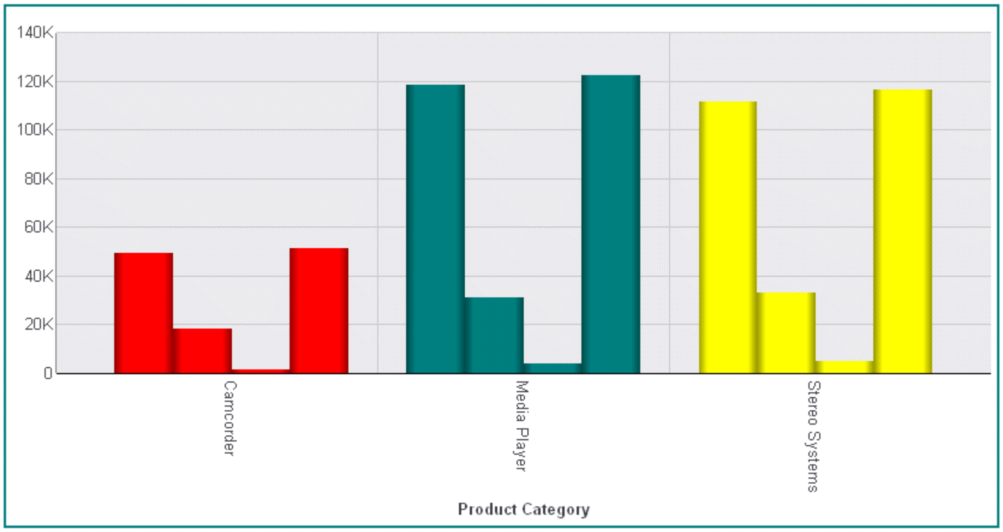
On the output, each series has its own color, and that color is repeated for each group (PRODUCT_CATEGORY):



The following version of the request changes the color mode to byGroup:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"legend": {"visible": false},
"colorMode": {"mode": "byGroup"},
"series": [
    {"series": "all", "border": {"width": 1, "color": "grey"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "teal"},
    {"series": 2, "color": "yellow"},
    {"series": 3, "color": "cyan"}]
*END
ENDSTYLE
END
```

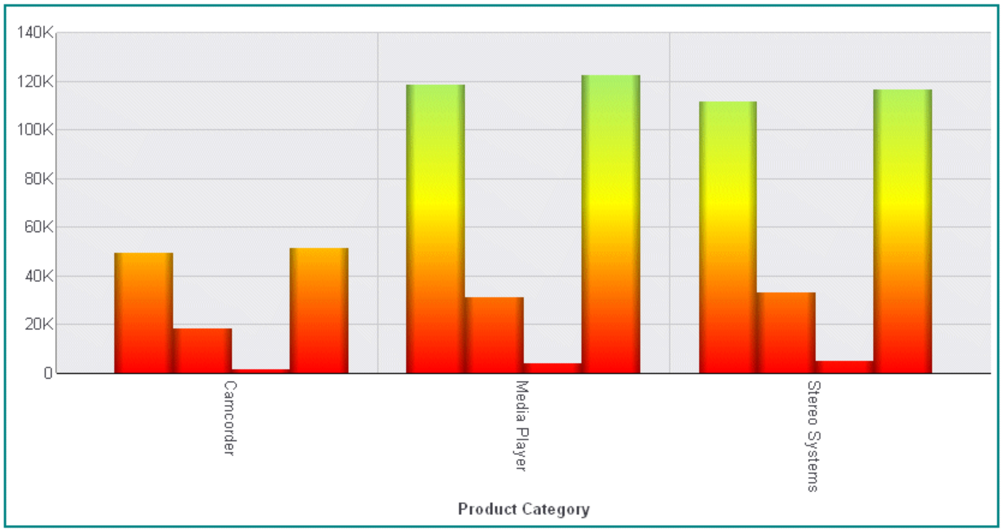
On the output, all series in each group (PRODUCT_CATEGORY) are colored the same. Group 0 uses the color set for series 0, group 1 uses the color set for series 1, and so on. When the series colors are used up, default colors are used:



Next, change the color mode to byHeight and add a color list:

```
"colorMode": {
    "mode": "byHeight",
    "colorList": ["red","yellow","lightgreen"]}
```

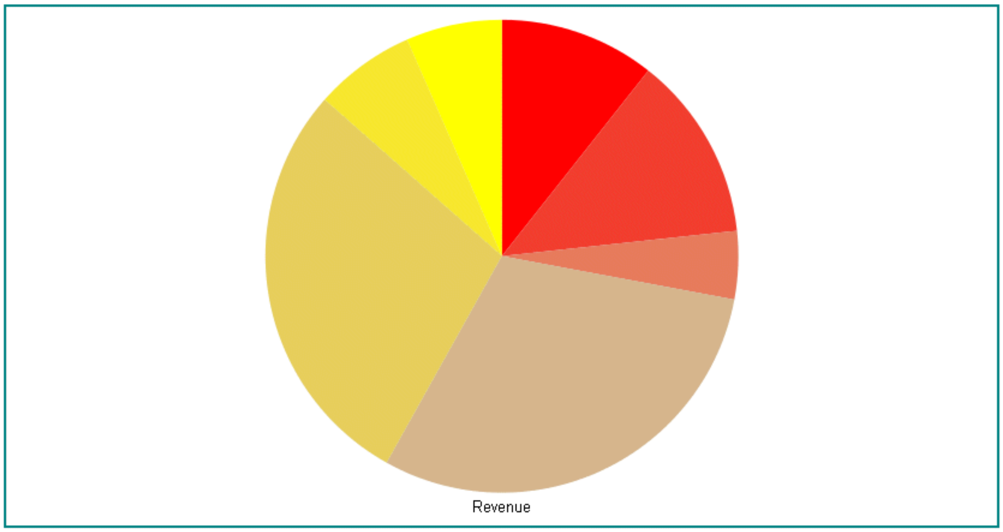
On the output, the risers are colored using a gradient such that the first color is at the numeric axis minimum, the last color is at the numeric axis maximum, and the remaining colors are interpolated between these:



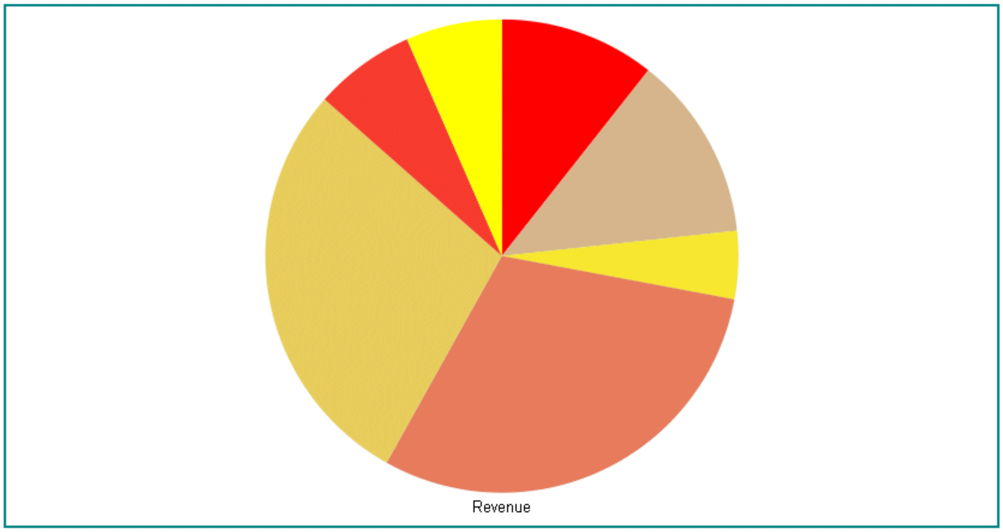
The next request generates a pie chart with the color mode byInterpolation with a color list:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"legend": {"visible": false},
"colorMode": {
    "mode": "byInterpolation",
    "colorList": ["red", "tan", "yellow"]}
*END
ENDSTYLE
END
```

On the output, the first series is colored the first color in the colorList array, the last series is colored the last color, and the series in between are interpolated accordingly:



Changing the color mode to byInterpolationAlt generates the following chart, in which the colors have been interpolated and shuffled:



Defining Colors for Color Modes

When the colorMode property is set to "byInterpolation" or "byHeight", the colorModeColors property defines the range of colors to apply to risers and markers.

- ❏ When colorMode is "byInterpolation", the first series is colored the first color in this array, the last series is colored the last color, and the series in between are interpolated accordingly.
- ❏ When colorMode is "byHeight", the colors defined here create a gradient such that the first color is at the numeric axis minimum, the last color is at the numeric axis maximum, and the remaining colors are interpolated between these. This gradient is then used to color the risers.

Syntax: How to Define Colors for Color Modes

```
"colorModeColors": ["string",...,"string"]
```

where:

`["string",...,"string"]`

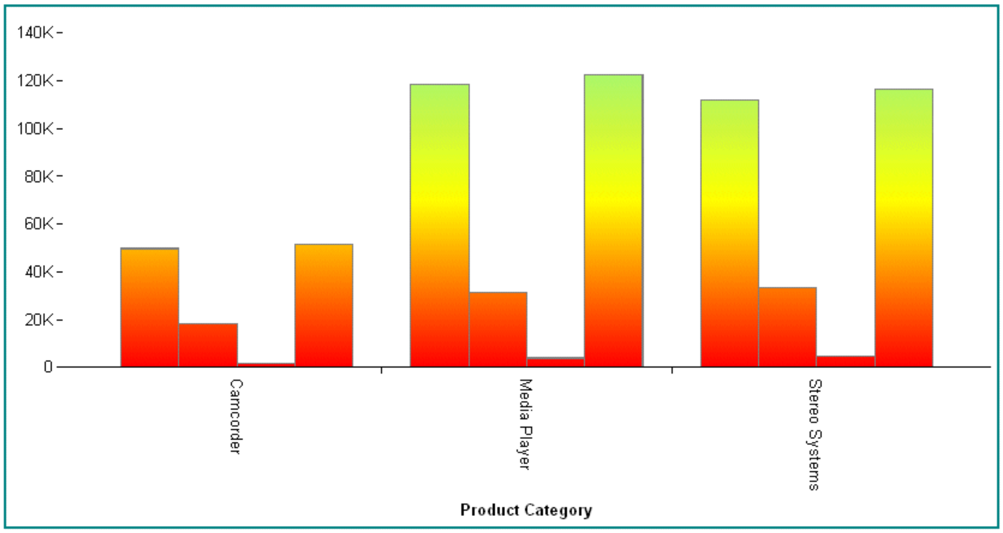
Is an array of color strings defined by a color name or numeric specification string. The default value is undefined.

Example: Defining Color Mode Colors

The following request generates a vertical bar chart with color mode "byHeight" and colorModeColors red, yellow, and light green:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"legend": {"visible": false},
"colorMode": "byHeight",
"colorModeColors": ["red", "yellow", "lightgreen"],
"series": [
    {"series": "all", "border": {"width": 1, "color": "grey"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "teal"},
    {"series": 2, "color": "yellow"},
    {"series": 3, "color": "cyan"}]
*END
ENDSTYLE
END
```

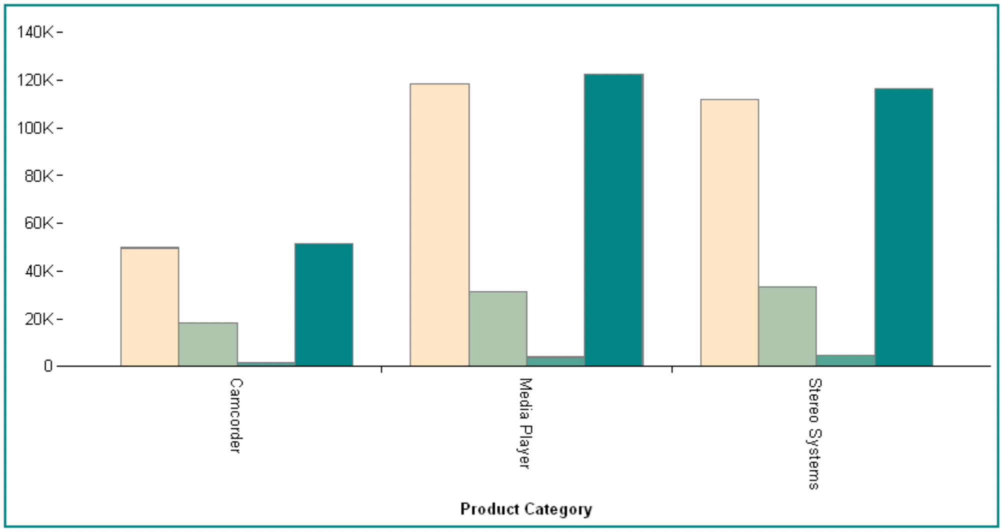
On the output, the colors are used to generate a gradient fill for the risers in which the first color is at the numeric axis minimum, the last color is at the numeric axis maximum, and the remaining colors are interpolated between these:



In the following request, the color mode is 'byInterpolation', and the color mode colors are bisque and teal:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"border": {"width": 2, "color": "teal"},
"legend": {"visible": false},
"colorMode": "byInterpolation",
"colorModeColors": ["bisque", "teal"],
"series": [
    {"series": "all", "border": {"width": 1, "color": "grey"}},
    {"series": 0, "color": "red"},
    {"series": 1, "color": "teal"},
    {"series": 2, "color": "yellow"    },
    {"series": 3, "color": "cyan"}]
*END
ENDSTYLE
END
```

On the output, the first series is colored the first color in the array, the last series is colored the last color, and the series in between are interpolated accordingly:



Defining a Color Scale (colorScale)

The colorScale property defines the colors to use for drawing the color scale for any chart that has a field assigned to the color attribute category.

You can also visualize the color transitions in a heatmap or choropleth as discrete bands. For map charts, you can also visualize the legend for map charts using bin markers. For information, see *Defining a Color Scale Color Mode* on page 771.

Syntax: How to Define a Color Scale

```
  "colorScale": {
    "colors": ["string", "string", ..., "string"]
},
```

or:

```
"colorScale": {
       "colors":[
                 {
                  "start": value,
                  "stop": value,
                  "color": "string"
                  },
                  .
                  .
                  .
                ]
          }
```

where:

`"colors"`

Can be an array of colors or an array or start values, stop values, and colors.

`["string", "string", ..., "string"]`

Is an array of colors defined by a color name or numeric specification string. The default value is: ["#253494", "#2C7FB8", "#41B6C4", "#A1DAB4"].

`"start": value or "pin": value`

Is a number or percent string that defines where to start the color gradient.

Note that "pin" is a synonym for "start".

`"stop": value`

Is a number or percent string that defines where to stop the color gradient.

The stop value is optional. If it is not specified, the color scale at the start value will be the defined start color. It will then smoothly transition from this color to the next start or pin color value.

If two color entries have matching start and stop values, they will be drawn with discrete solid colors, even in continuous mode. For example, the following properties will not draw gradient colors, just solid red from 0 to 50 then green from 50 to 100.

```
[{"start": 0, "stop": 50, "color": "red"},
{"start": 50, "stop": 100, "color": "green"} ]
```

`"color": "`*`string`*`"`

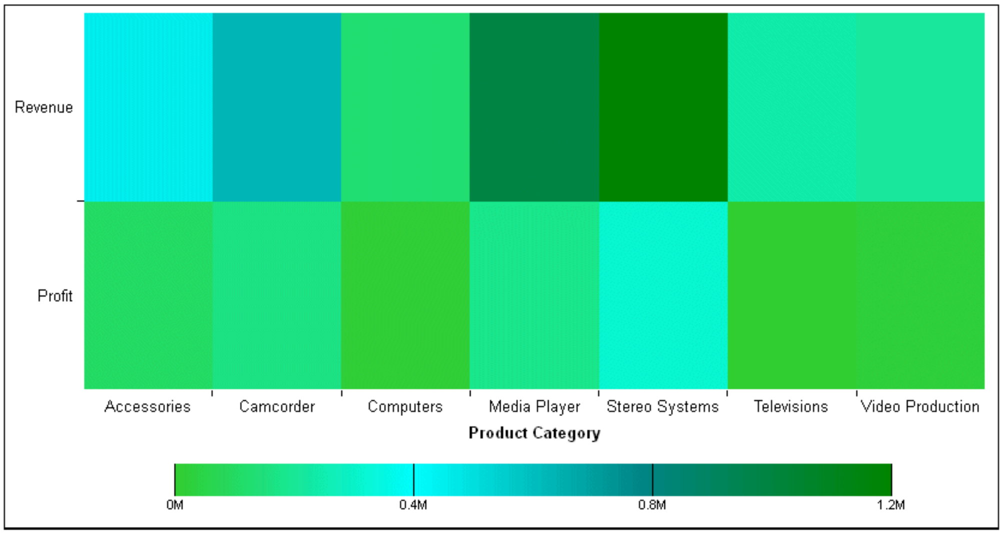
Defines the color for the interval.

Example: Defining a Color Scale for a Heatmap Chart

The following request generates a heatmap chart with a color scale consisting of the colors lime green, cyan, teal, and green:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US AS 'Revenue'
        GROSS_PROFIT_US AS 'Profit'
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SPECTRAL
ON GRAPH SET STYLE *
*GRAPH_JS
"colorScale": {"colors": ["limegreen", "cyan", "teal", "green"]}
*END
ENDSTYLE
END
```

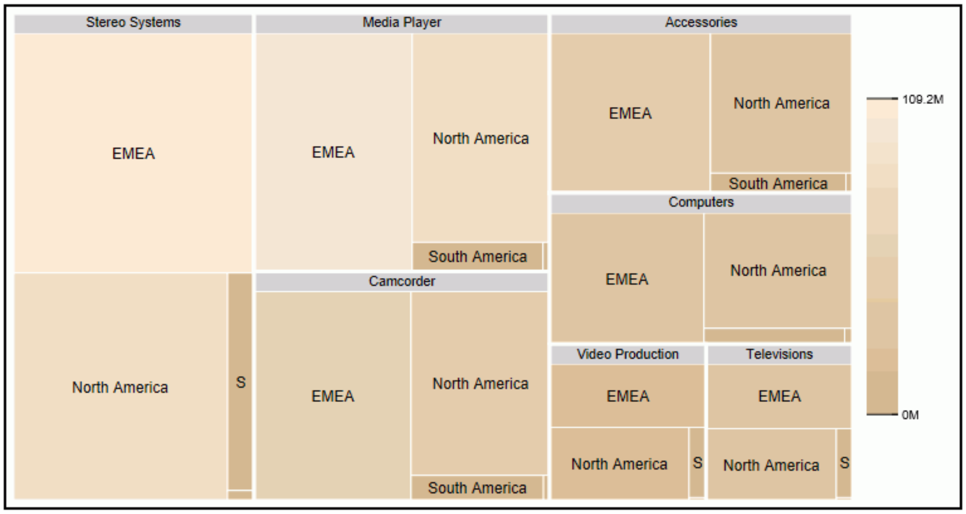
The output is:



The following request generates a treemap chart with a color scale consisting of the colors tan and antique white:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY PRODUCT_CATEGORY
BY BUSINESS_REGION
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"legend": {"visible": true},
"colorScale": {"colors": ["tan", "antiquewhite"]}
*END
ENDSTYLE
END
```

The output is:

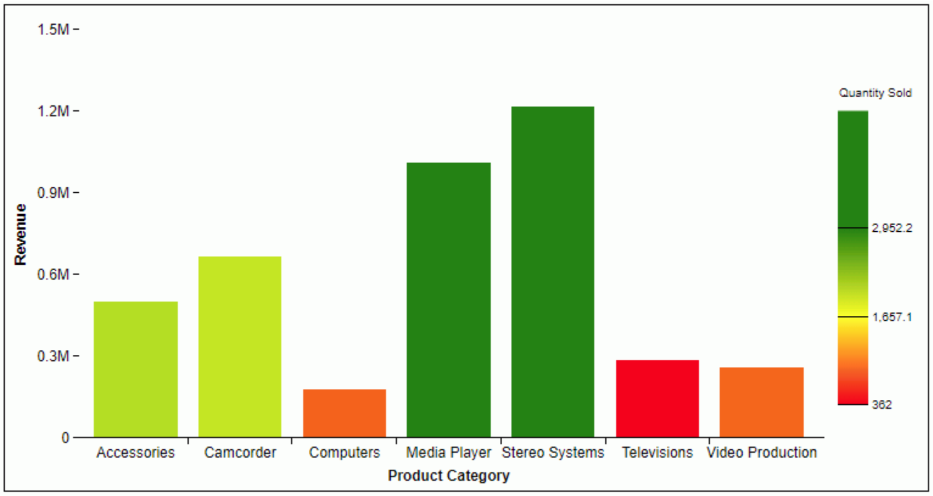


Example: Defining a Color Scale for a Bar Chart

The following request uses continuous color mode (the default), and defines a color scale.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US  QUANTITY_SOLD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=N1, BUCKET=x-axis,$
TYPE=DATA, COLUMN=N2, BUCKET=y-axis,$
TYPE=DATA, COLUMN=N3, BUCKET=color,$
*GRAPH_JS_FINAL
"yaxis": {"majorGrid": {"visible": false}},
"xaxis": {"majorGrid": {"visible": false}},
"colorScale": {
    "colors": [
         {"pin": 0, "color": "red"},
         {"start": "30%", "color": "yellow"},
         {"pin": "60%", "color": "green"}
     ] }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Defining a Color Scale Color Mode

Three color modes define the way color transitions are visualized on the chart and in the legend.

- Continuous mode is the default visualization. In continuous mode, the colors in the color scale form a blended gradient.
- Discrete mode visualizes the colorScale in discrete color bands. You define an array of color bands that provide start and stop values for each color. The color bands must be defined within the colorScale object.
- Bin mode creates discrete markers on the legend for each entry in the colorScale array. This is useful when the color bands generate an unappealing legend that is hard to read.

Syntax: How to Define a Discrete Color Scale

For discrete mode, the syntax follows.

```
"colorScale": {
   "colorMode":"discrete",
   "invert": boolean,
   "colors":[
                 {
                  "start": value,
                  "stop": value,
                  "color": "string"
                  },
                  .
                  .
                  .
                ]
          }
```

where:

`"invert": boolean`

Defines the order of the color bands. Valid values are:

- ❑ true, which lists the bands from high to low.
- ❑ false, which lists the bands from low to high. This is the default value.

`"colors":`

Is an array of start values, stop values, and colors for each color band.

`"start": value`

Is a number or percent string that defines where to start the color band.

`"stop": value`

Is a number or percent string that defines where to stop the color band.

`"color": "string"`

Defines the color of the band.

For continuous mode, the syntax follows.

```
"colorScale": {
  "colorMode": "continuous",
  "colors": ["string","string", ..., "string"]
        }
```

where:

`"colors": ["string","string", ..., "string"]`

Specify the colors to be blended.

Note: If you set the color mode to "continuous" but define color bands, the chart will be visualized in discrete mode.

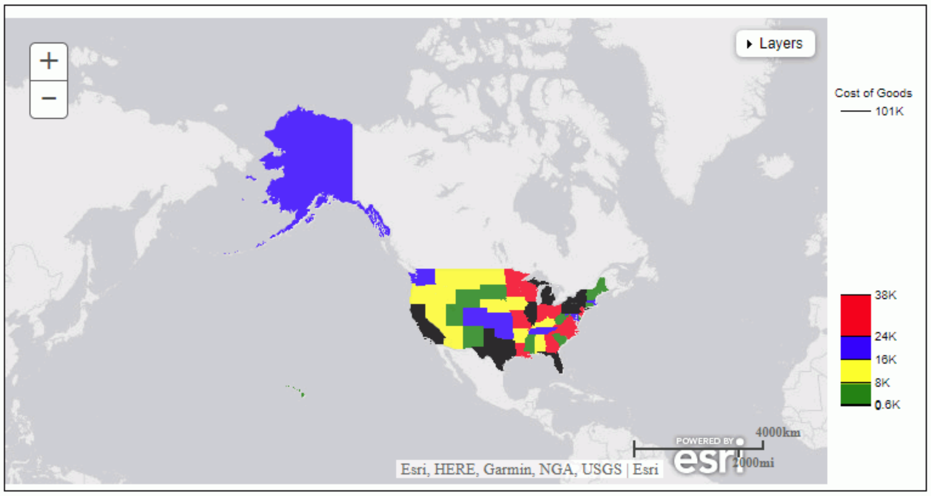
Example: Defining a Discrete Set of Color Bands for a Choropleth

The following request defines a choropleth with discrete color bands.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or url references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN= COGS_US, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
"com.esri.map": {
"overlayLayers":
[
 {
 "ibiDataLayer": {
   "map-metadata": {
      "map_by_field": "STATE_PROV_NAME"
                    }
                 }
 }
],
"baseMapInfo":
 {
   "customBaseMaps":
   [
    {
     "ibiBaseLayer": "gray"
    }
   ]
   }
 }
},
"legend":{"visible":true},
"colorScale":
{
 "colorMode":"discrete",
  "colors": [
   {"start": 0, "stop": 8000,"color":"green"},
   {"start": 8000, "stop": 16000,"color":"yellow"},
   {"start": 16000, "stop": 24000,"color":"blue"},
   {"start": 24000, "stop": 38000,"color":"red"}
        ]
}
*END
ENDSTYLE
END
```

The chart and legend are visualized using the discrete color bands defined in the color scale, as shown on the following image:



Syntax: How to Define Color Mode Bins

For color mode bin, the syntax follows.

```
"colorScale":
  {
    "colorMode": "bin",
    "invert": boolean,
      "binMarkers":
         {
           "visible": boolean,
           "concatSymbol": "string",
           "size": number,
           "shape": "string",
           "rotation": number,
           "position": "string",
           "border":
                {
                  "width": number,
                  "color": "string",
                  "dash": "string"
                }
         }
   }
```

where:

`"invert": boolean`

Defines the order of the color bands. Valid values are:

- true, which lists the bands from high to low.
- false, which lists the bands from low to high. This is the default value.

`"visible": boolean`

Specifies whether to show the bin markers. Valid values are:

- true, which shows the bin markers. This is the default value.
- false, which does not show the bin markers.

`"concatSymbol": "string"`

Is a string that defines the symbol used in each legend label between the minimum value and maximum value for the bin. The default concatenation symbol is " ... ".

`"size": number`

Is the diameter of the bin marker in pixels. The default value is 8.

`"shape": "string"`

Is a string that specifies any standard marker shape. The default value is "square".

`"rotation": number`

Is a number that defines the rotation of the marker in degrees. The default value is 0 (zero).

`"position": "string"`

Specifies the position of the markers relative to the labels. Valid values are:

- "left", which places the markers to the left of the labels. This is the default value.
- "right", which places the markers to the right of the labels.
- "bottom", which places the markers below the labels.
- "top", which places the markers on top of the labels.

`"border":`

Defines the properties of the marker border.

`"width": number`

Is a number that defines the width of the border in pixels. The default value is 1.

`"color": "string",`

Is a string that defines the color of the marker border. The default value is "black".

`"dash": "`*`string`*`"`

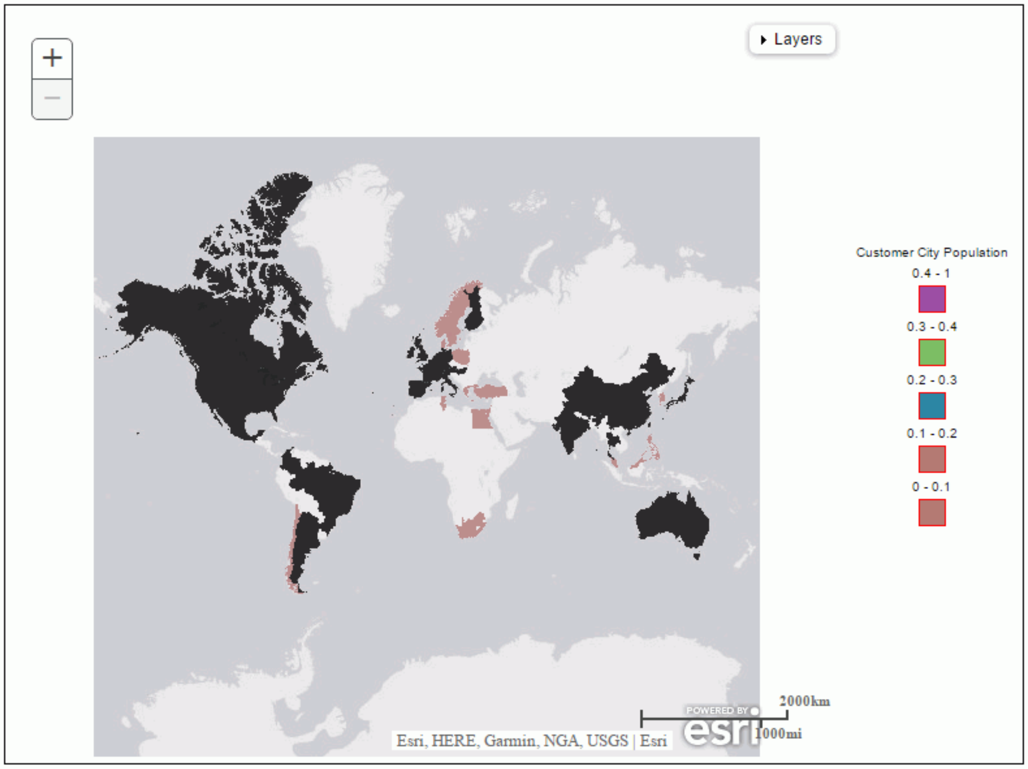
Is a string that defines the dash style of the marker border. Specify the length of a dash in pixels followed by the length of the space between dashes in pixels. For example, dash: "1 1" generates a dotted line. The default value is " ", which generates a solid border.

Example: Using Bin Color Mode for a Choropleth

The following request uses color mode bin. The bin markers are square and 20 pixels in diameter. The concatenation symbol between the start and stop values is " -". The bin markers are beneath the bin labels and have a red border one pixel wide. The color bands are inverted so that the band representing the highest percentages is on top.

```
GRAPH FILE WF_RETAIL_LITE
SUM CITY_POPULATION
BY COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=CITY_POPULATION, BUCKET=color, $

*GRAPH_JS
"legend": {"visible": true},
"colorScale":
      {
      "colorMode": "bin",
      "invert": true,
      "binMarkers":
           {
              "visible": true,
              "concatSymbol": " - ",
              "size": 20,
              "shape": "square",
              "rotation": 0,
              "position": "bottom",
                 "border": {
                     "width": 1,
                     "color": "red",
                           }
           },
      "colors":[
            {"start":"0%",  "stop":"10%",  "color":"#B57877"},
            {"start":"10%", "stop":"20%",  "color":"#2887A6"},
            {"start":"20%", "stop":"30%",  "color":"#7ABD66"},
            {"start":"30%", "stop":"40%",  "color":"#984EA3"},
            {"start":"40%", "stop":"100%", "color":"#4DAF4A"},
                   ]
      },
"extensions": {
"com.esri.map": {
"overlayLayers":
   [
    {
     "title": "WebFOCUS Countries Request",
     "layerType": "choropleth",
     "geometrySourceType": "esri",
     "geometryLocateField": ["Country"],
     "geometryDataField": "name",
     "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
    }
   ],
                }
              }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Support for Pictograms

Using the chart type "pictogram", you can use images as markers on the chart. The chart consists of bars, for each series, that are comprised of either repeated images that are stacked to the height of the bar or a single image that is stretched or shrunk to represent the height of the bar. The image for each series is specified in the marker:shape property as either a URL reference or a standard marker shape.

Syntax: How to Use Pictograms as Chart Markers

```
*GRAPH_JS
"chartType": "pictogram",
"pictogramProperties": {
       "fillMode": "string",
       "backFill": "string",
       "lockToGrid": boolean
},
"series":
[
  {
      "series": number,
      "color": "string",
      "marker":
     {
            "shape": "string",
            "color": "string",
            "border": {"width": number, "color": "string"}
         }
   }
]
```

where:

`"pictogramProperties"`

Defines the properties of the pictogram images.

`"fillMode": "string"`

Controls how each bar is filled. Valid values are:

- "repeatClip", which repeatedly stacks the image to the height of the bar. This is the default value.
- "stretch", which stretches or shrinks the image to the height of the bar.

`"backFill": "string"`

Defines a color specification string to use with a standard marker shape and the property "fillMode": "repeatClip". It repeats the marker shape all the way to the top of the axis space instead of to the height of the bar, but fills the markers above the height of the bar with this color. The default value is "undefined".

`"lockToGrid":` *`boolean`*

If true, shrinks the chart so that exactly one pictogram glyph fits between each y-axis gridline. The default value is false.

`"series"`

Defines the properties of the series that will be represented by the images.

`"series":` *`number`*

Is the number of the series to be represented by the pictogram.

`"color": "`*`string`*`"`

Is a color specification string for the series. To see the pictograms without the series bars, use the property "color": "transparent".

`"marker"`

Defines the properties of the marker for the series.

`"shape": "`*`string`*`"`

Can be either a URL reference to an image file, or a standard marker shape. A URL reference must be in the following format.

```
"url(url_to_image_file)"
```

Image types supported include .gif, .jpg, .png, and .svg.

`"color": "`*`string`*`"`

Defines the color for the marker when it is a standard marker shape. This property is ignored for an image file.

`"border"`

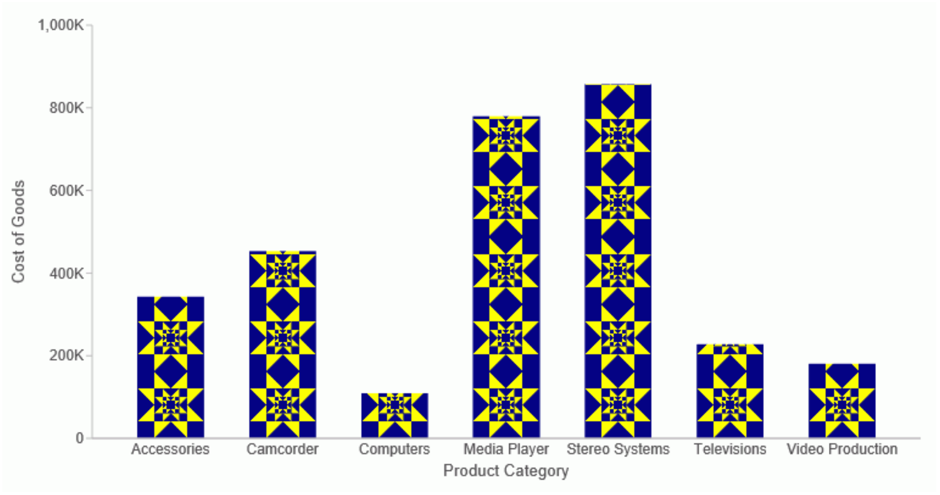
Defines the width in pixels and a color specification string for the marker border. Border properties are ignored when the marker shape is an image file.

Example: Using a Repeated Image File as a Pictogram

The following request uses a repeated quilt block image to generate the bars on the chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=N1, bucket=x-axis, $
type=data, column=n2, bucket=y-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS
"chartType": "pictogram",
"pictogramProperties": {
    "fillMode": "repeatClip",
    "backFill": "undefined",
},
"series": [{
 "series": 0,
  "color": "transparent",
  "marker": {
  "shape": "url(http://ecl.informationbuilders.com/jschart/quilt.gif)",
  "color": "undefined",
  "border": {"width": 1, "color": "#DDDDDD"}
    }
}]
*END
ENDSTYLE
END
```

The output is shown in the following image.

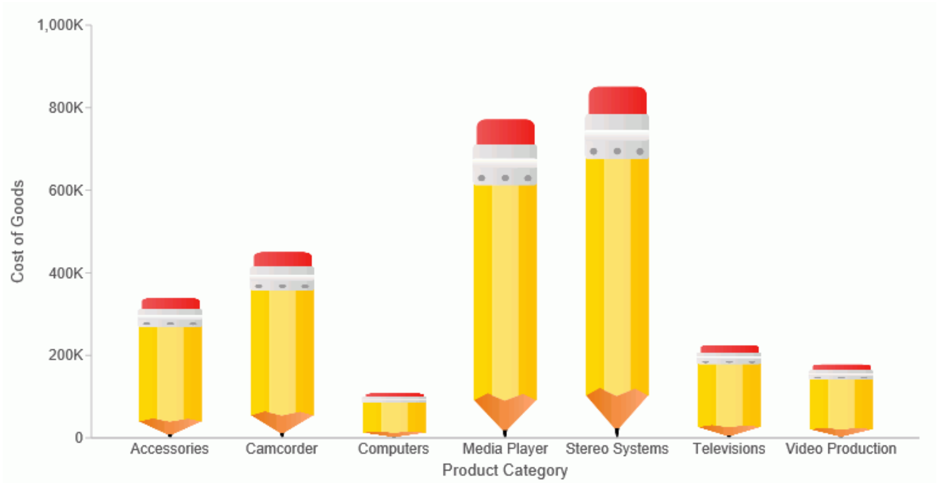


Example: Using a Stretched Image File as a Pictogram

The following request uses a stretched image file to generate the bars on the chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=N1, bucket=x-axis, $
type=data, column=n2, bucket=y-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS_FINAL
"chartType": "pictogram",
"pictogramProperties": {
    "fillMode": "stretch",
    "backFill": "undefined",
},
"series": [{
 "series": 0,
    "color": "transparent",
    "marker": {
    "shape": "url(http://ecl.informationbuilders.com/jschart/pencil.gif)",
        "color: "red",
        "border": {"width": 1, "color": "#DDDDDD"}
    }
}]
*END
ENDSTYLE
END
```

The output is shown in the following image.

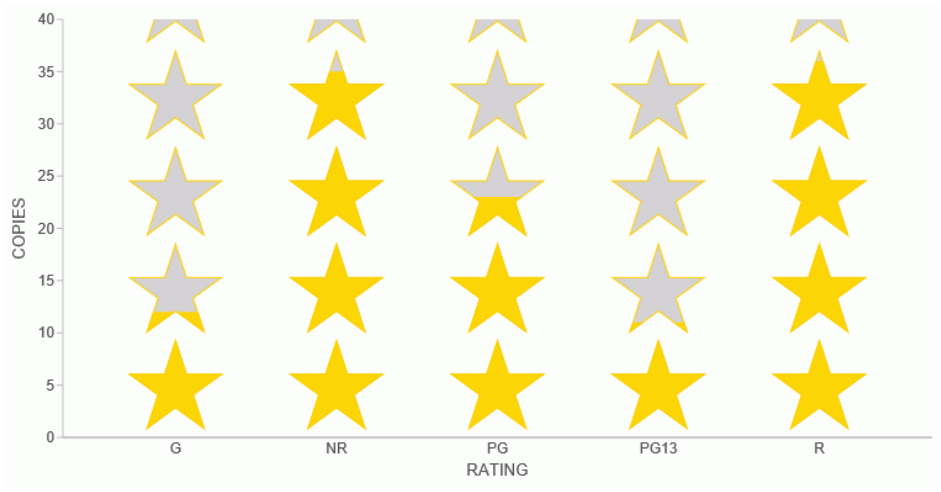


Example: Using a Standard Marker Shape With Backfill as a Pictogram

The following request uses the fivestar maker shape with light gray backfill to generate the chart.

```
GRAPH FILE MOVIES
SUM COPIES
BY RATING
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
type=data, column=N1, bucket=x-axis, $
type=data, column=n2, bucket=y-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
*GRAPH_JS
"chartType": "pictogram",
"pictogramProperties": {
    "fillMode": "repeatClip",
    "backFill": "lightgray",
},
"series": [{
    "series": 0,
    "color": "transparent",
    "marker": {
        "shape": "fivestar",
        "color": "gold",
        "border": {"width": 1, "color": "#FECF01"}
    }
}]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Drawing Error Bars (errorBars)

The errorBars properties define error bars that can be drawn on bar risers, line risers, area risers, bubble markers, and scatter markers to visualize where risers are above or below expected values.

The following code shows the properties and default values:

```
"errorBars": {
   "yData": undefined,
   "xData": undefined,
   "hatWidth": '50%',
   "line": {"color": "black","width": 1,"dash": ""},
   "marker": {
      "color": "grey",
      "shape": "diamond",
      "size": "undefined",
      "border": {"width": 1,"color": "black","dash": ""}
   }
},
```

Syntax: How to Draw Error Bars

```
"errorBars": {
"yData": [highValue, markerPosition, lowValue, ...]
   "xData": [highValue, markerPosition, lowValue, ...],
   "hatWidth": value,
   "line": {"color": "string",
          "width": number,
          "dash": "string"},
   "marker": {
      "color": color,
      "shape": "string",
      "size": number,
      "border": {
               "width": number,
               "color": "string",
               "dash": "string"
}
   }
}
```

where:

`yData: [highValue, markerPosition, lowValue, ...] or`

`yData: [highValue, lowValue, ...]`

Defines one or more arrays of high values, marker positions (optional), and low values for each riser. For no error bars on y-axis data, the value should be *undefined*. Each array can have either of the following formats:

- ❏ [highValue,lowValue], in which case there will be error bars on the corresponding riser, but no marker.
- ❏ [highValue, markerPosition, lowValue], in which case both error bars and a marker will be drawn on the corresponding riser.

`xData: [highValue, markerPosition, lowValue, ...] or`

`xData: [highValue, lowValue, ...]`

Defines one or more arrays of high values, marker position (optional), and low values for each riser. For no error bars on x-axis data, the value should be *undefined*. Each array can have either of the following formats:

- ❏ [highValue,lowValue], in which case there will be error bars on the corresponding riser, but no marker.
- ❏ [highValue, markerPosition, lowValue], in which case both error bars and a marker will be drawn on the corresponding riser.

`hatWidth:` *`value`*

Is the width of the error bar as a number in pixels or as a string (enclosed in double quotation marks and including a percent symbol) that specifies a percentage of the width of the riser.

`line:`

Defines the properties of the line.

`color: "`*`string`*`"`

Is a color defined by a color name or numeric specification string that defines the color of the line and hats.

For information about defining colors, see *Colors and Gradients* on page 85.

`width:` *`number`*

Is a number that defines the width of the error bar line in pixels. The default value is 1.

`dash: "`*`string`*`"`

Is a string that defines the dash style of the line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line). Use "" for a solid line.

`marker:`

Defines the properties of the markers for a series.

`color:` *`color`*

Is a color for markers for a series, defined by a color name, numeric specification string, or gradient string (enclosed in double quotation marks), or a gradient defined by a JSON Object. The default value is "grey".

`shape: "`*`string`*`"`

Is a string that defines the shape of markers for a series. For a list of marker shapes, see *Series-Specific Properties* on page 413. The default value is "diamond". Note that bar, circleMinus, circlePlus, cross, and tick markers require a border width and color.

`size:` *`number`*

Is a number that defines the diameter of the marker in pixels, or undefined (to let the engine choose the marker size). The default value is *undefined*.

`border:`

Defines the properties of the marker border.

`width: number`

Is a number that defines the width of the border in pixels. The default value is "black".

`color: "string"`

Is a color for the marker border defined by a color name or numeric specification string. The default value is "black".

`dash: "string"`

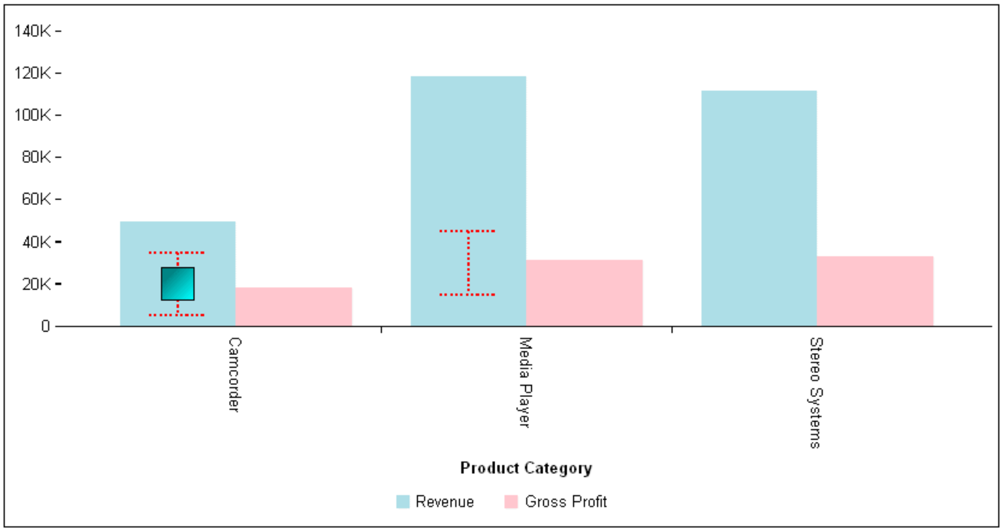
Is a string that defines the marker border dash style. The default value is "" (which generates a solid line). Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes.

Example: Generating Error Bars

The following request generates red dashed error bars on the first two risers. The error bars on the first riser also include a square marker with a gradient color:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"errorBars": {
    "hatWidth": "50%",
    "yData": [[[5000000, 20000000, 35000000], [15000000, 45000000]]],
    "line": {"color": "red", "width": 2, "dash": "2 2"},
    "marker": {
        "color": "linear-gradient(0,0,100%,100%, 20% teal, 95% cyan)",
        "shape": "square", "size": 25}},
"series": [
    {"series": 0, "color": "lightblue"},
    {"series": 1, "color": "pink"}]
*END
ENDSTYLE
END
```

The output is:



Generating HTML Tooltips (htmlToolTip)

When tooltips are defined for one or all series, the htmlToolTip properties enable and define or disable HTML-based (div) style tooltips for any chart tooltips. The series-specific tooltip property defines the text to show in the tooltip.

The following code shows the properties and default values:

```
"htmlToolTip": {
    "enabled": true,
    "mouseMargin": 10,
    "style": "undefined",
    "autoTitleFont": "bold 12pt Sans-Serif",
    "autoContentFont": "10pt Sans-Serif",
    "snap": false,
 "sticky": "auto"
 "fill": 'linear-gradient(to bottom, rgba(250,250,250,0.98)
8%,rgba(230,230,230,0.98) 95%)',
 "border": {
                "width": 1,
             "color": 'rgba(150,150,150,0.95)',
                 "cornerRadius": 0
 "cascadeMenuStyle": {
                "label": {
                         "font": undefined,
                         "color": undefined
                      },
                "nameValue": {
                         "name": {
                                  "font": "9pt sans-serif",
                                  "color": "#505050"
                               },
                         "value": {
                                  "font": "bold 9pt sans-serif",
                                  "color": "black"
                                 }},
      "hover": {
                         "fill": "rgba(220,220,220,0.9)",
                         "labelColor": "black"
                       }
                   }
                  }
```

Syntax: How to Generate HTML Tooltips

```
"htmlToolTip": {
            "enabled": boolean,
   "mouseMargin": number,
   "style": "string",
   "autoTitleFont": "string",
   "autoContentFont": "string",
   "snap": boolean,
   "sticky": boolean ,
   "fill": "string",
   "border": {
                "width": number,
             "color": "string",
                "cornerRadius": {"x": value, "y": value}
                             },
   "cascadeMenuStyle": {
                "label": {
                         "font": "string",
                         "color": "string"
                      },
                "nameValue": {
                         "name": {
                                  "font": "string",
                                  "color": "string"
                               },
                         "value": {
                                  "font": "string",
                                  "color": "string"
                                }},
      "hover": {
                         "fill": "string",
                         "labelColor": "string"
                       }


            }
              }
```

where:

`"enabled": boolean`

Enables or disables HTML-based tooltips. Valid values are:

- true, which uses HTML-based tooltips for all chart tooltips. This is the default value.
- false, which uses standard style tooltips for all chart tooltips.

`"mouseMargin": number`

Is the distance from the top of the cursor to the bottom of the tooltip, in pixels.

`"style": "`*`string`*`"`

Defines the style of the tooltip. Valid values are:

- ❑ "seriesFill", which fills the tooltip background with a lightened version of the series color.
- ❑ an object or string defining CSS properties.
- ❑ undefined. This is the default value.

`"autoTitleFont": "`*`string`*`"`

When series:tooltip is set to "auto", use a CSS font string to define the formatting of automatic tooltip title text. The default value is "bold 12pt Sans-Serif".

`"autoContentFont": "`*`string`*`"`

When series:tooltip is set to "auto", use a CSS font string to define the formatting of automatic tooltip content text. The default value is "10pt Sans-Serif".

`"snap": `*`boolean`*

Enables or disables snapped tooltips. Valid values are:

- ❑ true, to enable snap. The tooltip draws at a hardcoded spot next to each riser on the chart.
- ❑ false, to disable snap. The tooltip draws several pixels above the cursor. This is the default value.

`"sticky": `*`boolean`*

Enables or disables sticky tooltips. Valid values are:

- ❑ true, to enable sticky tooltips. The tooltip is only hidden when the mouse moves out of the chart frame.
- ❑ false, to disable sticky tooltips. The tooltip is hidden when the mouse moves off any riser. This is the default value.

`"fill": "`*`string`*`"`

Is a color or gradient string that defines the fill for the tooltip container, or undefined. If it is undefined, the fill is the default fill coded in the chart engine. The default value is "linear-gradient(to bottom, rgba(250,250,250,0.98) 8%,rgba(230,230,230,0.98) 95%)".

`"border"`

Defines the properties of the border of the tooltip container.

`"width":` *`number`*

Defines the width of the border in pixels, or undefined. If undefined, use the default coded in the chart engine. The default value is 1.

`"color": "`*`string`*`"`

Is a color string that defines the color of the border, or undefined. If undefined, use the default coded in the chart engine. The default value is "rgba(150,150,150,0.95)".

`"cornerRadius": {"x":` *`value`*`, "y":` *`value`*`}`

Defines the properties of the corners of the tooltip container.

The cornerRadius property can consist of one value or two values. A single zero (0) value generates square corners. This is the default corner shape. Rounded corners can be circular or elliptical. The x and y values denote the size of the circle radius or the semi-major and semi-minor axes of the ellipse, where x and y are orientation-aware and depend on chart orientation. x represents the ordinal axis, and y represents the numeric axis. The default value is 0.

Valid values for x and y are:

- ❏ A number that specifies the radius of the corner in pixels.
- ❏ A percent string such as "20%", that represents a percentage of the ordinal or numeric axis of the riser.
- ❏ A CSS length string such as "5em" or "10px" that specifies the radius of the corner in pixels. For information about CSS length strings, see *https://developer.mozilla.org/en-US/docs/Web/CSS/length*.
- ❏ An object with "x" and "y" properties (where "x" and "y" can be any of the above) to specify the corner radius along the ordinal axis (x) and numeric axis (y) of the riser.

`"cascadeMenuStyle"`

Describes the properties of the tooltip menus.

`"label"`

Defines the properties of simple text in the tooltip.

`"font": "`*`string`*`"`

Is a string that defines the font of label text. The default value is undefined.

`"color": "`*`string`*`"`

Is a string that defines the color of label text. The default value is undefined.

`"nameValue":`

Defines the properties of the field names and field values in the tooltip.

`"name"`

Defines the properties of the field names in the tooltip.

`"font": "`*`string`*`"`

Is a string that defines the font of the field names. The default value is "9pt sans-serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the field names. The default value is "#505050".

`"value"`

Defines the properties of the field values in the tooltip.

`"font": "`*`string`*`"`

Is a string that defines the font of the field values. The default value is "bold 9pt sans-serif".

`"color": "`*`string`*`"`

Is a string that defines the color of the field values. The default value is "black".

`"hover"`

Defines the properties when the mouse hovers over the tooltip.

`"fill": "`*`string`*`"`

Defines the hover fill color. The default value is "rgba(220,220,220,0.9)".

`"labelColor": "`*`string`*`"`

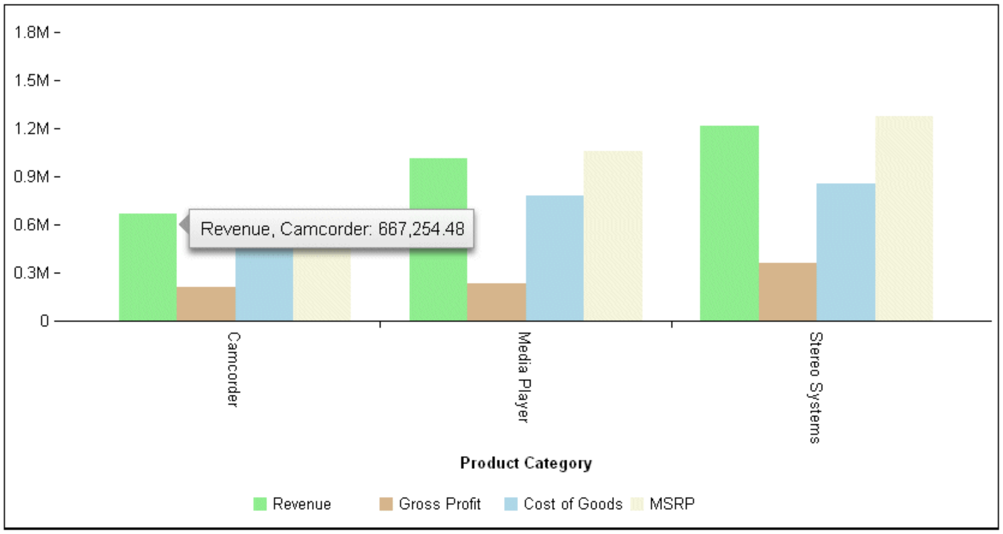
Defines the label hover color for a submenu. The default value is "black".

Example: Generating HTML-based Tooltips

The following request generates HTML-based tooltips:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US COGS_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen"},
    {"series": 1, "color": "tan"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "beige"}],
"htmlToolTip": {"enabled": true, "snap": true}
*END
ENDSTYLE
END
```

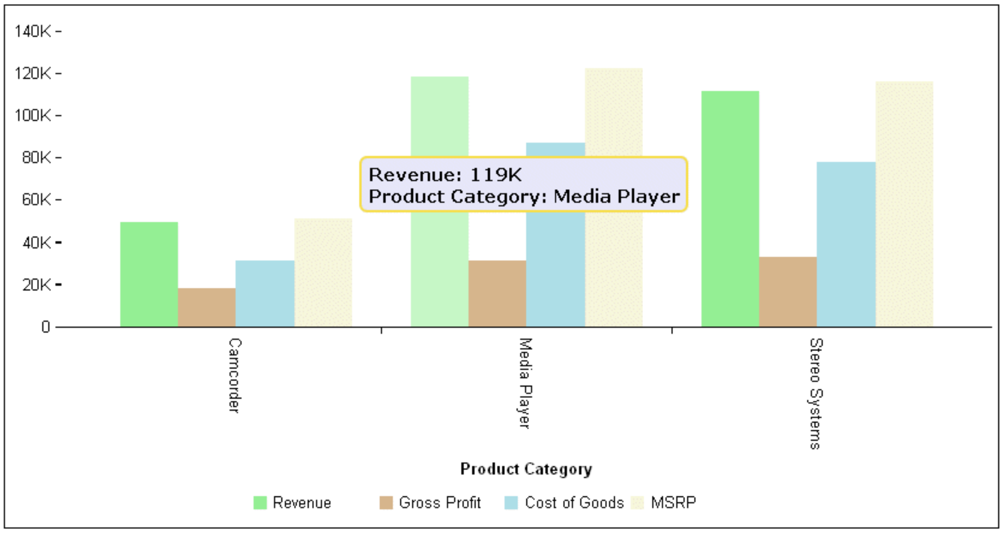
On the output, the tooltips have callout arrows because the snap property is enabled:



The following version of the request formats the background color of the tooltips and the font of the tooltip for series 0, which has been set to tooltip: "auto":

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US COGS_US MSRP_US
BY PRODUCT_CATEGORY
WHERE PRODUCT_CATEGORY EQ 'Camcorder' OR 'Media Player' OR 'Stereo Systems'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"series": [
    {"series": 0, "color": "lightgreen", "tooltip": "auto"},
    {"series": 1, "color": "tan"},
    {"series": 2, "color": "lightblue"},
    {"series": 3, "color": "beige"}],
"htmlToolTip": {
    "enabled": true,
    "style": {"background": "lavender"},
    "autoTitleFont": "italic 12pt Times New Roman",
    "autoContentFont": "bold 10pt Verdana"}
*END
ENDSTYLE
END
```

The output is:

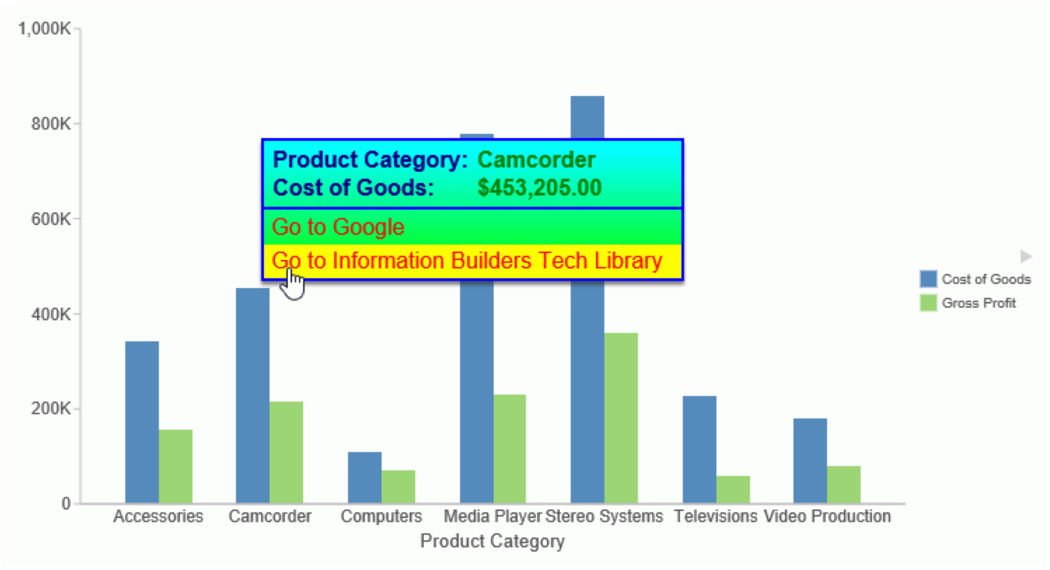


Example: Formatting Cascading Menus in HTML Tooltips

In the following request, series 0 has the default tooltip with drilldowns defined in the StyleSheet, while series 1 has a custom tooltip with a submenu. The field names are navy and their values are green. The labels are red. The hover fill color is yellow, and the hover label color for the submenu is brown.

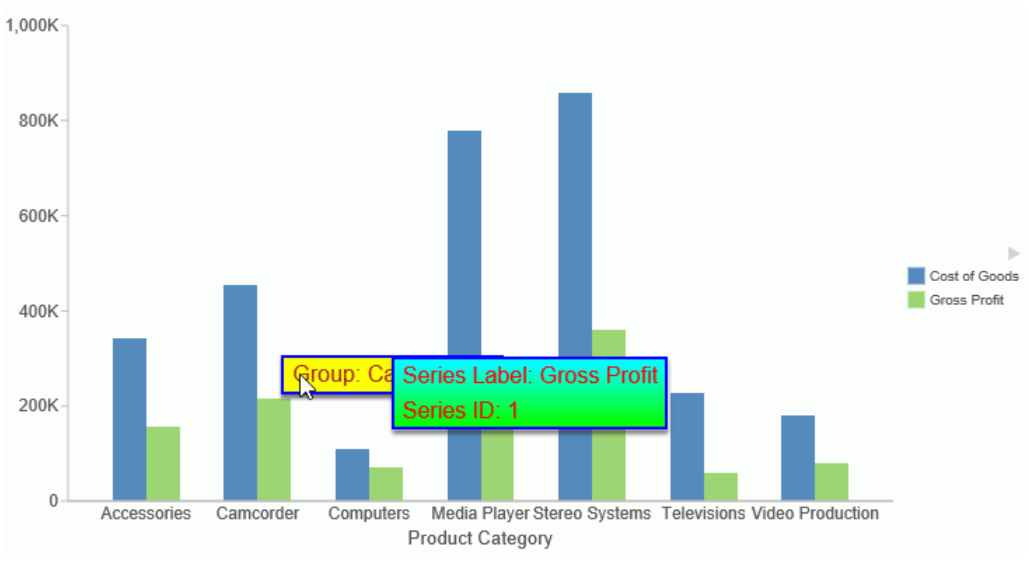
```
GRAPH FILE wfretail82/wf_retail_lite
SUM COGS_US
GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=x-axis, $
TYPE=DATA, COLUMN=N2, BUCKET=y-axis, DRILLMENUITEM='Go to Google',
TARGET='_blank', URL=http://www.google.com,
DRILLMENUITEM='Go to Information Builders Tech Library', TARGET='_blank',
   URL=https://webfocusinfocenter.informationbuilders.com/wfbue/technical-
library.html, $
TYPE=DATA, COLUMN=N3, BUCKET=y-axis, $
*GRAPH_JS_FINAL
"series": [{
    "series": "reset"},
    {"series": 1, "tooltip": [{
        "entry": "Group: {{group_label}}",
        "children": ["Series Label: {{series_label}}",
               "Series ID: {{series_id}} "]
                          }]
            }],
"htmlToolTip": {
    "enabled": true,
    "fill": "linear-gradient(to bottom, cyan 8%, lime 95%)",
    "border": {
                 "width": 2,
                 "color": "blue"
                       },
    "cascadeMenuStyle": {
          "label": {
          "font": "12pt Arial",
          "color": "red"
                   },
          "nameValue": {
                   "name": {
                   "font": "bold 12pt Arial",
                   "color": "navy"
                     },
                  "value": {
                   "font": "bold 12pt sans-serif",
                   "color": "green"
                      }
                        },
            "hover": {
                   "fill": "yellow",
                   "labelColor": "brown"
                     }
                         }
                    }
*END
ENDSTYLE
END
```

The following image shows the mouse hovering over the first drilldown for series 0.



Product Category: Camcorder
Cost of Goods: $453,205.00
Go to Google
Go to Information Builders Tech Library

The following image shows the mouse hovering over the submenu for series 1.



Defining User Interaction With the Chart (interaction)

The interaction property defines user interaction with the chart.

- For bar, line, area, bubble, and scatter charts, mousedrag: 'pan' will reposition the risers, markers, axis labels, and data labels (if shown) when a user clicks and drags the mouse inside the chart frame.
- For line, scatter, and area charts, mousemove:'nearestNeighbor' makes tooltips visible even when the mouse moves away from a data point.
- For 3D charts, mousedrag: 'rotate' will rotate the chart frame when a user clicks and drags the mouse around the chart frame.
- For pie charts, when an *other* slice is defined with pieProperties:otherSlice, the click: 'otherSliceDrillDown' option will drill-down into the other slice components (that is, the drilled-down pie chart will show the slices that make up the other slice).

Syntax: How to Define User Interaction With the Chart

```
"interaction": {
   "click": "string",
   "mousedrag": "string",
   "dblclick": "string",
   "mousemove": "string"
},
```

where:

`"click": "string"`

Is a string that defines interaction on mouse click. Valid values are:

- "otherSliceDrillDown", which shows the slices that make up the other slice.
- undefined. This is the default value.

`"mousedrag": "string"`

Is a string that defines the interaction on mouse drag:

- "select", which selects a chart riser and marker.
- "pan", which on mouse drag, shows other values in the data set.
- "rotate", which for 3D charts only, rotates the 3D cube.
- undefined, which does nothing. This is the default value.

`"dblclick": "string"`

Is a string that defines the interaction on mouse double click. Valid values are:

- "resetView", which works with "pan" and "rotate". It causes a double-click to reset the chart to its original view.
- undefined. This is the default value.

`"mousemove": "string"`

Specifies a tooltip detection method for line, scatter, and area charts. Valid values are:

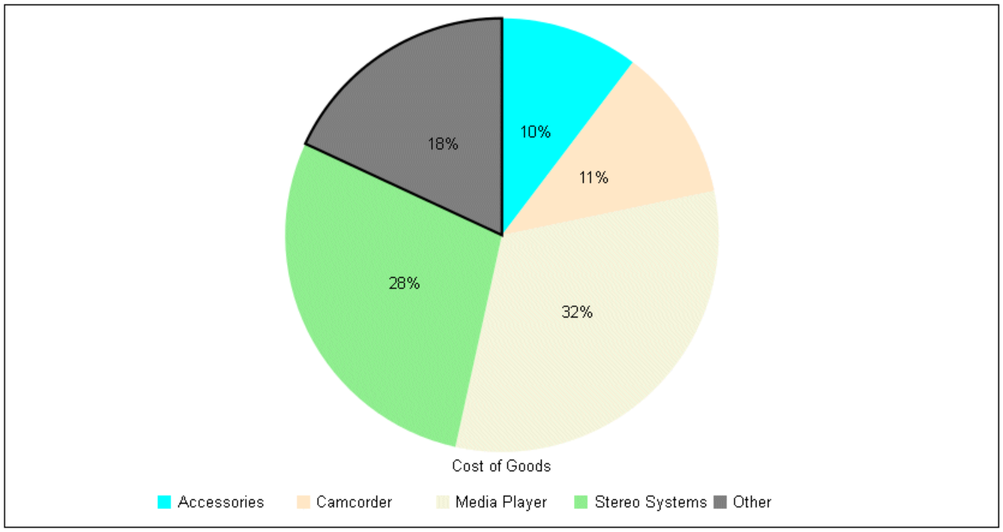
- "over", which only displays a tooltip when the mouse hovers directly over a data point. This is the default value.
- "nearestNeighbor", which displays the tooltip for the nearest data point on line, scatter, and area charts, even when the mouse moves away from a data point.

Example: Enabling Interaction on a Pie Chart

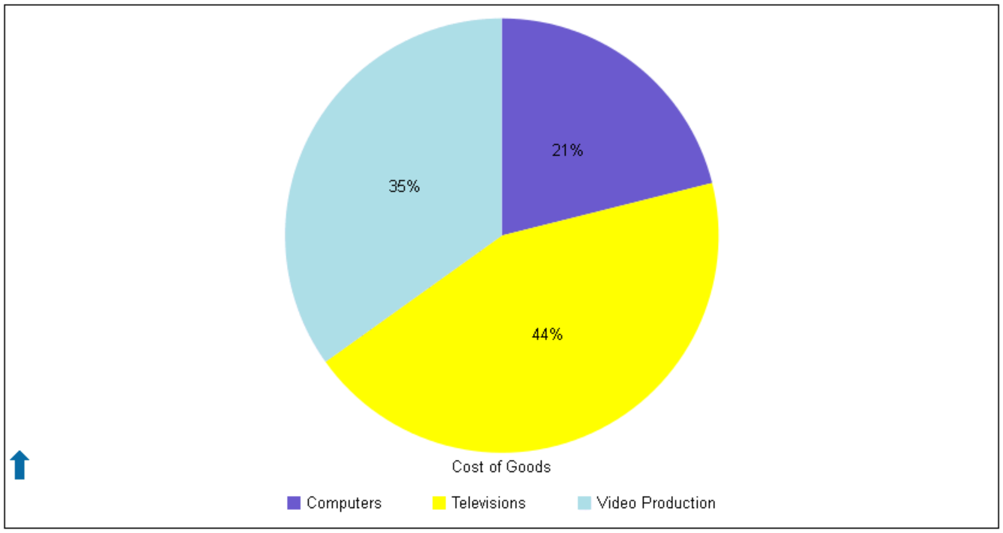
The following request generates a pie chart with the otherSlice set to a threshold of 8%. The interaction setting causes the chart to show the components of the otherSlice when the user clicks the other slice:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
*GRAPH_JS
"dataLabels": {"visible": true},
"pieProperties": {"otherSlice": {"threshold": "8%"}},
"interaction": {"click": "otherSliceDrillDown"},
"series": [
    {"series": 0, "color": "cyan", "showDataValues": true},
    {"series": 1, "color": "bisque", "showDataValues": true},
    {"series": 2, "color": "slateblue", "showDataValues": true},
    {"series": 3, "color": "beige", "showDataValues": true},
    {"series": 4, "color": "lightgreen", "showDataValues": true},
    {"series": 5, "color": "yellow", "showDataValues": true},
    {"series": 6, "color": "lightblue", "showDataValues": true},
    {"series": 7, "color": "lavender", "showDataValues": true},
    {"series": 8, "color": "limegreen", "showDataValues": true},
    {"series": 9, "color": "red", "showDataValues": true}]
*END
ENDSTYLE
END
```

On the output, the slices with values below the threshold percent are grouped together in one grey slice:



Clicking the other slice generates the following drill down of the other slice, showing the components and their percentages within the other slice. The blue arrow on the bottom left resets the original view, when clicked:

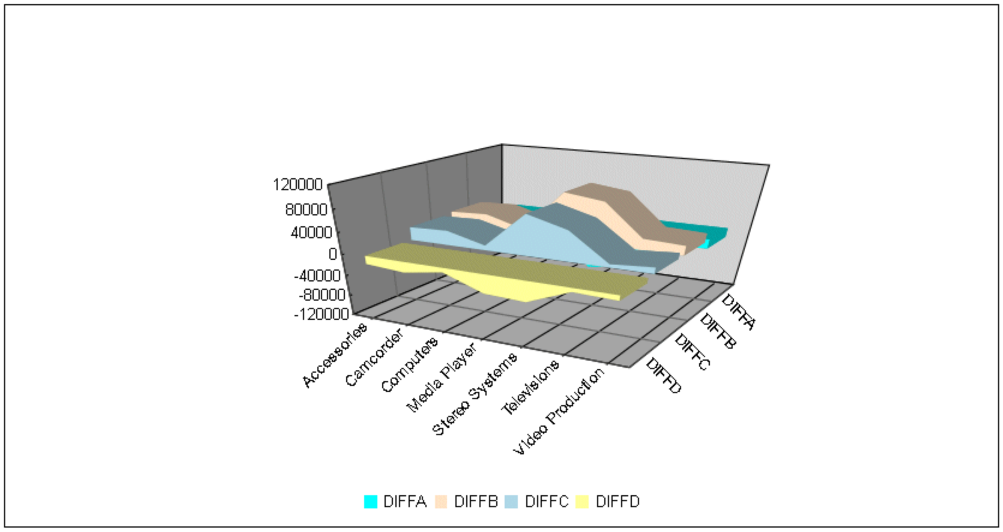


Example: Enabling Rotation on a 3D Chart

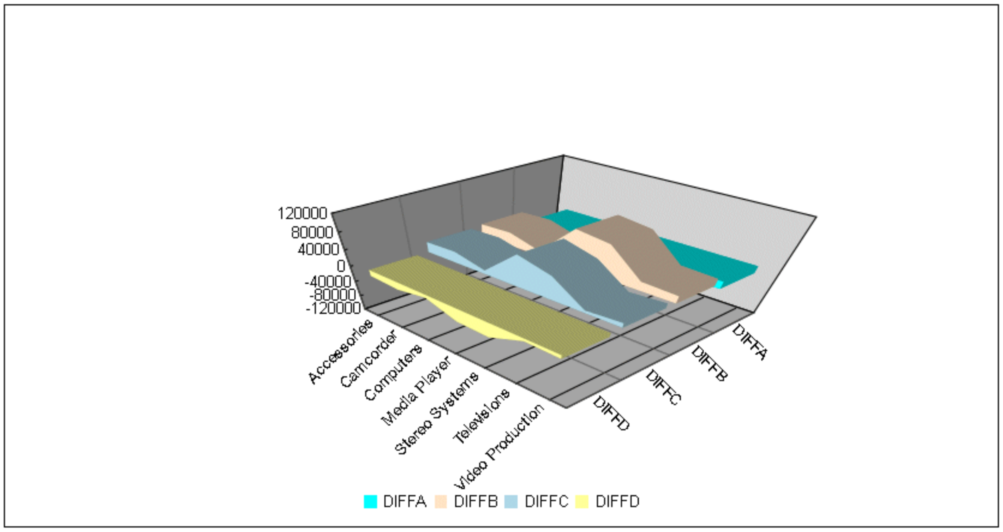
The following request generates a 3D area chart. The interaction properties enable rotating the chart on mousedrag and resetting the original view on double-click:

```
DEFINE FILE WF_RETAIL_LITE
DIFFA = GROSS_PROFIT_US - REVENUE_US;
DIFFB = REVENUE_US - GROSS_PROFIT_US;
DIFFC = COGS_US - (COGS_US * DISCOUNT_US)/100;
DIFFD = COGS_US - MSRP_US;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFFA DIFFB DIFFC DIFFD
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH 3DAREAS
ON GRAPH SET STYLE *
*GRAPH_JS
  interaction:
      {mousedrag: 'rotate',
      dblclick: 'resetView'},
series: [
    {series: 0, color: 'cyan'},
    {series: 1, color: 'bisque'},
    {series: 2, color: 'lightblue'},
]
*END
ENDSTYLE
END
```

The following image shows the original view of the chart:



Clicking outside the chart and dragging rotates the chart:



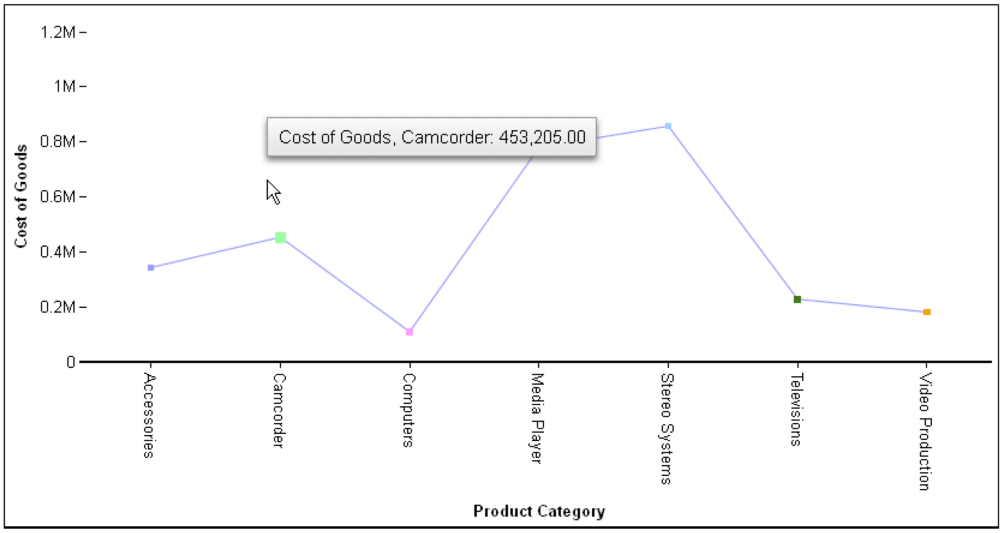
Double-clicking outside the chart resets it to its original view.

Example: Enabling Nearest Neighbor Tooltip Detection for a Line Chart

The following request generates a line chart with nearest neighbor tooltip detection:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY  PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
"interaction": {"mousemove": "nearestNeighbor"}
*END
END
```

The following image shows that the tooltip is visible even though the mouse is not hovering over a data point on the chart:



Formatting the Mouse Over Indicator (mouseOverIndicator)

The mouseOverIndicator property defines a color to highlight the riser when the mouse hovers over a riser. For line, combo, and pareto charts, the marker property defines a marker to draw when the mouse hovers over a line riser.

- For line charts with no markers (series:#, marker:{visible:false}), one marker is added to the line at the point on the line nearest to the cursor. This marker is created with the properties defined by mouseOverIndicator:marker (not the series-dependent marker).
- For line charts with markers (series:#,marker:{visible:true}) and scatter charts, the series-defined marker nearest to the cursor will be drawn as larger (zoomed). The appearance of this border cannot be controlled, and can only be disabled by setting mouseOverIndicator:enabled to false.
- Markers on bubble charts are drawn with a grey border. They do not zoom. The marker size is dependent on the data.
- For other chart types (such as bar, area, pie, tagcloud, treemap, and heatmap), the standard highlight defined by mouseOverIndicator:color is used, and no markers.

The following code shows the properties and default values for the indicator:

```
"mouseOverIndicator": {
    "enabled": true,
    "marker": {
        "color": "red",
        "size": 25,
        "shape": "square",
        "rotation": 0,
        "border": {"width": 1, "color": "darkblue"}
        }}
```

Syntax: How to Format the Mouse Over Indicator

```
"mouseOverIndicator": {
    "enabled": boolean,
   "color": "string",
   "marker": {
      "color": "string",
      "size": number,
      "shape": "string",
      "rotation": number,
      "border": {
         "width": number,
         "color": "string",
         "dash": "string"
      }
   }
},
```

where:

`"enabled": `*`boolean`*

Enables or disables the mouse over indicator. Valid values are:

- true, which enables the mouse over indicator. This is the default value.
- false, which disables the mouse over indicator.

`"color": "`*`string`*`"`

Defines a color to apply when the mouse hovers over a riser. A color can be defined by a color name or numeric specification string. It can also be defined as a percentage string in the range "100%" to "100%", to lighten or darken the series color by the specified percentage. The default value is undefined or the string "".

`"marker":`

Defines the properties of markers for a line riser.

`"color": "`*`string`*`"`

Defines the marker color for a line riser (in line, combo or pareto charts). It can be a color defined by a color name or numeric specification string. The default value is "lightblue".

`"size":` *`number`*

Is a number that defines the size of the marker. The default value is 5.

`"shape": "`*`string`*`"`

Is a string that defines the shape of a marker. The default value is "square".

For information about marker shapes, see *Series-Specific Properties* on page 413.

`"rotation":` *`number`*

Is a number between 0 and 360 that defines the angle (in degrees) of the marker. The default value is zero (0).

`"border":`

Defines the properties of the marker border.

`"width":` *`number`*

Is a number that defines the width of the marker border in pixels. The default value is 1.

`"color": "`*`string`*`"`

Is a color for the marker border defined by a color name or numeric specification string. The default value is "darkblue".

`"dash": "`*`string`*`"`

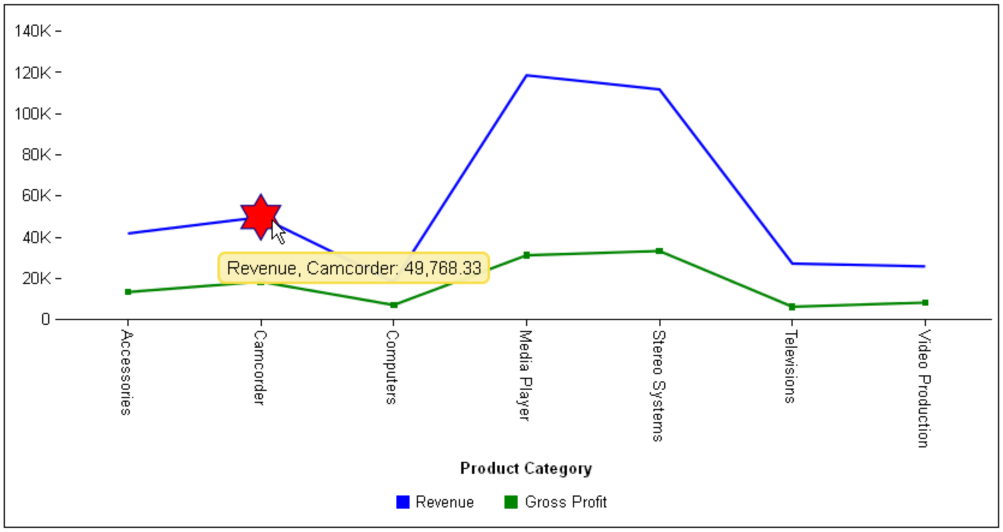
Is a string that defines the border dash style. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes. The default value is "" (which draws a solid line).

Example: Formatting the Mouse Over Indicator

The following request generates a vertical line chart with two series. Series 0 has no markers and series 1 has default markers. The mouse over indicator is defined as red, and the marker shape for lines with no markers is a six-pointed star with a dark blue border:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VLINE
ON GRAPH SET STYLE *
*GRAPH_JS
series:[
"series": [
    {"series": 0, "color": "blue", "marker": {"visible": false}},
    {"series": 1, "color": "green"}],
"mouseOverIndicator": {
    "enabled": true,
    "marker": {
        "color": "red", "size": 35, "shape": "sixStar", "rotation": 0,
        "border": {"width": 1, "color": "darkblue"}}}
*END
ENDSTYLE
END
```

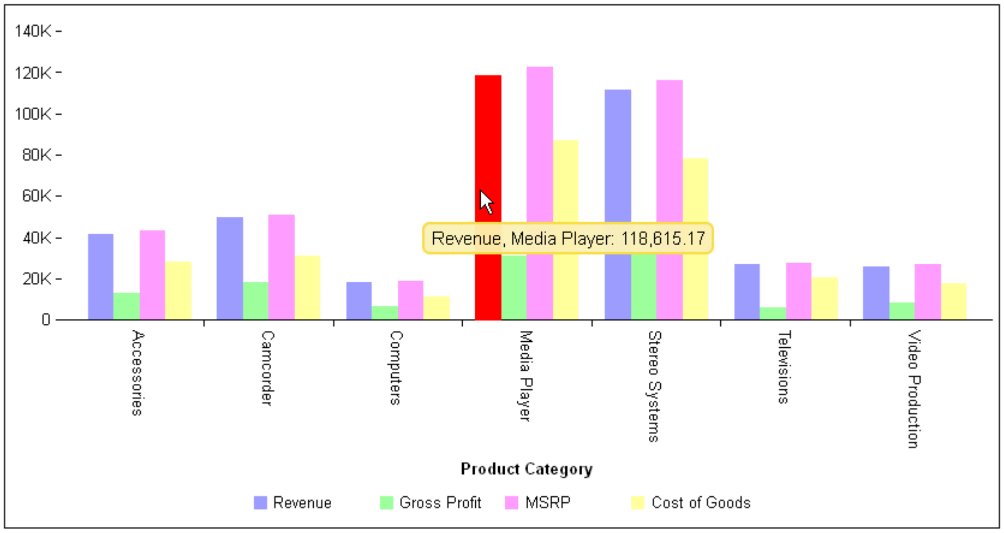
The following image shows the mouse over indicator on the line that had no markers:



The following request generates a vertical bar chart and makes the mouse over indicator red:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US MSRP_US COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"mouseOverIndicator": {"enabled": true, "color": "red"}
*END
ENDSTYLE
END
```

The following image shows the how a riser becomes red when the mouse hovers over it:



Drawing Reference Lines (referenceLines)

The referenceLines properties define one or more reference lines to draw on an axis.

Note: If an axis has an automatic scale, and you define a reference line at a value that is below the minimum or above the maximum value displayed on the scale, the scale will adjust its minimum or maximum value to make sure that the reference line is visible.

Syntax: How to Draw Reference Lines

```
"referenceLines": [
   {
   "value": number,
   "axis": "string",
   "line": {
      "color": "string",
      "width": number,
      "dash": "string"
   },
   "label": {
      "text": "string",
      "font": "string",
      "color": "string"
   },
   "anchor": "string",
   "showValue": boolean   }
]
```

where:

`"value": number`

Is a number or string that defines where on the axis to draw the line.

- ❑ For a numeric axis, the number must be a value that is visible on the axis, and a string must represent a percentage between "0%" and "100%".
- ❑ For the ordinal axis, a string must be a group label that is visible on the axis, and a number must be between 0 and 1 (for example, .5 will draw the reference line in the center of the chart).

`"axis": "string"`

Is a string that defines the axis on which to draw the line: "x", "y", "y2", or undefined (do not draw line).

`"line":`

Defines the properties of the reference line.

`"color": "string"`

Is a color defined by a color name or numeric specification string that defines the color of the line. The default is "black".

`"width": number`

Is a number that defines the width of the line in pixels. The default is 1.

`"dash": "`*`string`*`"`

Is a string that defines the dash style of the line. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line). Use "" for a solid line.

`"label":`

Defines the properties of the reference line label.

`"text": "`*`string`*`"`

Is an optional string that defines the label to draw beside a reference line. Use "", or undefined Empty string, or undefined, to draw no label.

`"font": "`*`string`*`"`

Is a string that defines the size and type face of the label. The default is "7.5pt Sans-Serif".

`"color": "`*`string`*`"`

Is a color for the label defined by a color name or numeric specification string that defines the color of the label.

`"anchor": "`*`string`*`"`

Is a string that defines where to draw the reference label relative to the line. Valid values are: "start" (same side as axis labels) or "end" (opposite side).

- "start", which draws the reference label on the same side as axis labels.
- "<u>end</u>", which draws the reference label on the side opposite the axis labels. This is the default value.

`"showValue": `*`boolean`*

Enables or disables showing the reference line value. Valid values are:

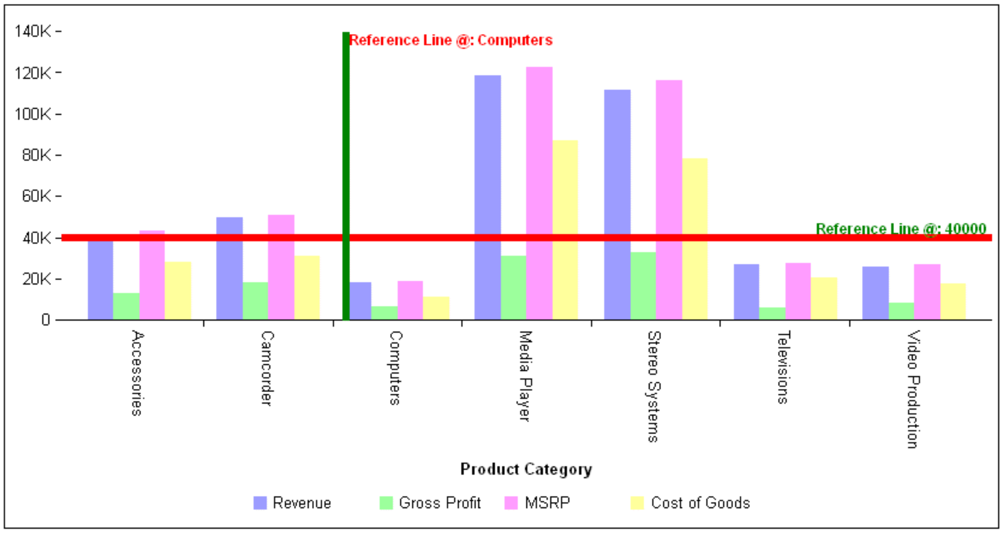
- true, to draw the value at the specified anchor location relative to the line.
- <u>false</u>, to not draw the value. This is the default value.

Example: Drawing Reference Lines

The following request generates a vertical bar chart with two reference lines. The y-axis reference line is a red line drawn at the value 40,000, with a green label that shows the value as part of the label. The x-axis reference line is a green line drawn at the value Computers, with a red label that shows the value as part of the label:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US GROSS_PROFIT_US MSRP_US COGS_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_JS
"referenceLines": [
    {"value": 40000, "axis": "y",
        "line": {"color": "red", "width": 6, "dash": ""},
        "label": {"text": "Reference Line @:",
            "font": "bold 8pt Sans-Serif", "color": "green"},
        "anchor": "end", "showValue": true},
    {"value": "Computers", "axis": "x",
        "line": {"color": "green", "width": 6, "dash": ""},
        "label": {"text": "Reference Line @:",
            "font": "bold 8pt Sans-Serif", "color": "red"},
        "anchor": "end", "showValue": true}]
*END
ENDSTYLE
END
```

The output is:



Drawing Trendlines in Bubble and Scatter Charts (trendline)

The trendline properties draw a trendline and equation label in bubble and scatter charts.

Note: You can also draw a trendline for an individual series, For information, see *Series-Specific Properties* on page 413.

The following code shows the properties and default values for the trendline:

```
"trendline": {
    "enabled": false,
    "mode": "undefined",
    "lineStyle": {
        "width": 1,
        "color": "black",
        "dash": ""
        },
    "equationLabel": {
        "visible": false,
        "font": "8pt Sans-Serif",
        "color": "black",
        "mode": "equation"
        }
    }
```

Syntax: How to Draw a Trendline in Bubble and Scatter Charts

```
 "trendline": {
     "enabled": boolean,
     "mode": "string"
     "lineStyle": {
          "width": number,
          "color": "string",
          "dash": "string"
      },
     "equationLabel": {
         "visible": boolean,
         "font": "string",
         "color": "string",
         "mode": "string"
      }
    }
```

where:

`"enabled": boolean`

Enables or disables the trendline. Valid values are:

- true, which enables the trendline.
- false, which disables the trendline. This is the default value.

`"mode": "string"`

Is string that defines the trendline mode (the type of equation used to generate the trendline). Valid values are:

- "exponential".
- "geometric".
- "hyperbolic",
- "linear".
- "logarithmic".
- "logQuadratic".
- "modExponential".
- "modHyperbolic".
- "polynomial".
- "quadratic".
- "rational".
- undefined. This is the default value.

`"lineStyle":`

Defines the properties of the trendline.

`"width": number`

Is a number that defines the width of the trendline in pixels. The default value is 1.

`"color": "string"`

Is a color defined by a color name or numeric specification string that defines the color of the trendline. The default value is "black".

`"dash": "string"`

Is a string that defines the dash style of the trendline. Use a string of numbers that defines the width of a dash followed by the width of the gap between dashes (for example, dash: "1 1" draws a dotted line). The default value is "", which draws a solid line.

`"equationLabel":`

Defines the properties of the equation label.

`"visible":` *`boolean`*

Controls the visibility of the equation label. Valid values are:

- ❑ true, which shows the equation label.
- ❑ false, which does not show the equation label. This is the default value.

`"font": "`*`string`*`"`

Is a string that defines the size, style, and, typeface of the equation label. The default value is "8pt Sans-Serif".

`"color": "`*`string`*`"`

Is a color for the equation label defined by a color name or numeric specification string. The default value is "black".

`"mode": "`*`string`*`"`

Is a string that defines the equation label mode. Valid values are;

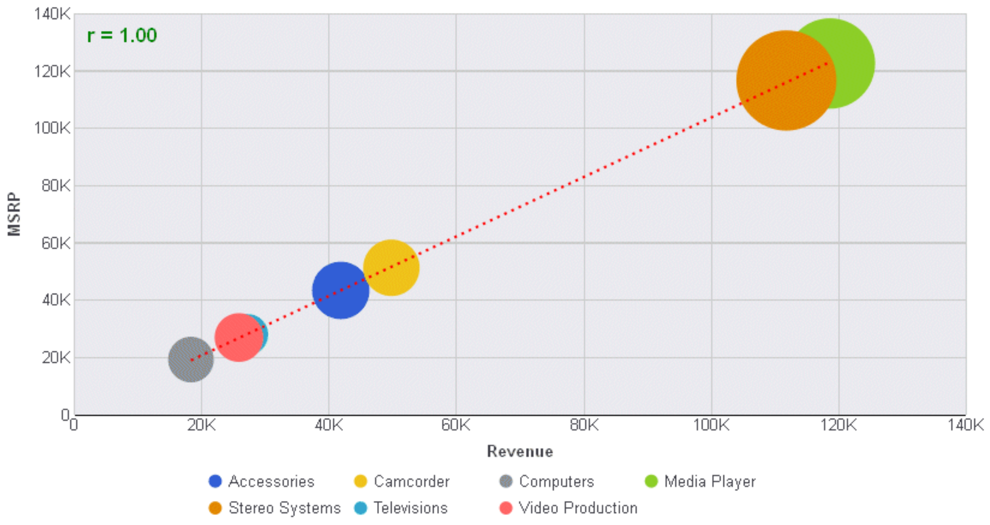
- ❑ "equation" (displays the equation in the format most appropriate for the type of trendline generated. For a linear trendline, the equation displays in slope intercept form). This is the default value.
- ❑ "r" (displays the correlation coefficient).

Example: Drawing Trendlines in Bubble and Scatter Charts

The following request generates a bubble chart and defines a linear trendline that is a red dashed line and has a green equation label that shows the correlation coefficient:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US MSRP_US DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
*GRAPH_JS
"trendline": {
    "enabled": true, "mode": "linear",
    "lineStyle": {"width": 2, "color": "red", "dash": "2 4"},
    "equationLabel": {
        "visible": true, "font": "bold 12pt Sans-Serif",
        "color": "green", "mode": "r"}}
*END
ENDSTYLE
END
```

The output is:



Chapter 13

Map Support

Basic GIS (Geographic Information System) capabilities are built into WebFOCUS. Data that is bound to a geolocation, such as state, country, or ZIP code, or to longitude and latitude values, can be viewed as symbol layers integrated into a powerful map viewer. Popular supported formats are proportional symbol maps (also called bubblemaps), in which markers are drawn on top of the map, and choropleths (polygons that color locations on top of the map).

Note: Map requests must use strict JSON syntax. This means that all property and object names and all string values must be enclosed in double quotation marks.

In this chapter:

Overview of Map Support

The WebFOCUS mapping architecture features two HTML5 map viewer engines, one from Environmental Systems Research Institute, Inc. (ESRI®) and one from Leaflet. Both of these engines use Esri map tiles as their base maps. They both also provide zoom, pan and scale controls with 19 levels of zoom detail.

- The Esri map viewer provides a more accurate geometry with richer symbolization than the Leaflet map viewer. Using the Esri map viewer, your charts can include demographic layers with data available on an Esri server (such as population density data) and feature layers that outline geographic areas on the map. It also provides a host of customization options through its API. However, the Esri viewer cannot be easily redirected in house, as the Leaflet map viewer can, if you encounter firewall or security policy issues. Also, not all of the Esri options are freely available. If you require services or data not available for free, you can license additional services or a local server from Esri.

Esri HTML5 charts are only supported with WebFOCUS chart attribute syntax, described in *WebFOCUS Chart Attribute Syntax* on page 143.

- The map viewer from Leaflet includes a default mapping server supplied by Esri. You can easily import map tiles (bitmaps) from other mapping services such as Google™, and UrbanMaps©. For more information about supported map services and other supported features of the Leaflet engine, see *http://leafletjs.com/*.

  Leaflet HTML5 charts support both WebFOCUS chart attribute syntax and traditional WebFOCUS syntax. If you use the chart attribute syntax, the attribute categories are the same as for the Esri charts.

Like all HTML5 visualizations, the highlighted markers and regions on the maps support additional chart properties and WebFOCUS request features such as drill, multi-drill, and informational tooltips. Advanced users can configure the map server to add their own locations and map definitions, using GeoJSON and JavaScript files.

WebFOCUS Parameters and Attribute Categories for Map Charts

When using chart attribute syntax, each field in the WebFOCUS request must be assigned to the appropriate attribute category. The categories used depend on the chart type.

Syntax: How to Specify the Chart Type for Map Charts

To generate a choropleth chart, specify the following command in your WebFOCUS request:

```
ON GRAPH SET LOOKGRAPH CHOROPLETH
```

To generate a proportional symbol (bubblemap) chart, specify the following command in your WebFOCUS request:

```
ON GRAPH SET LOOKGRAPH BUBBLEMAP
```

where:

`CHOROPLETH`

Fills specified regions of a map with heat-coded or discrete color values.

`BUBBLEMAP`

Draws proportional symbols on top of a map.

Reference: Attribute Categories for Map Charts

In order to use the Esri map engine, you must use chart attribute syntax in your WebFOCUS chart request. Leaflet maps also support chart attribute syntax, in addition to traditional chart syntax.

For complete information about WebFOCUS chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

The following table lists the attribute categories required for each type of map chart.

Choropleth	Bubblemap	Description
location **Note:** ❑ In a choropleth, if the location is represented by a single point instead of a polygon, a bubble marker (circle) is drawn to represent the location. The markers will all be the same size, the default marker size defined in the WebFOCUS StyleSheet. The field assigned to the color category will be used to color the markers. ❑ If you are using Reporting Server syntax to generate an Esri map chart, the sort field is assigned in the Esri properties, not in the location attribute category.	location, point, or coordinates	Defines the areas to be colored on a choropleth or for placement of symbols on a bubblemap. These are generally sort fields in the request.
color	color (optional)	The area color on a choropleth or the symbol color on a bubblemap will depend on the field assigned to the *color* attribute category. If the field is a measure, the color will be visualized as a gradient. If the field is a dimension, the colors will be discrete.

Choropleth	Bubblemap	Description
	size	The size of the symbols on a bubblemap will depend on the value of the measure assigned to the *size* attribute category.

For example, the following request specifies the latitude and longitude sort fields and the size measure for a bubblemap, and assigns the fields to the appropriate attribute categories:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=STATE_PROV_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=STATE_PROV_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
...
```

In order to display the symbols and the base map, you need to add map properties to the WebFOCUS StyleSheet, as described in the following sections.

Introduction to Map Properties

WebFOCUS supplies default property values for many map attributes, such as the extent of the map in the frame and the default maximum and minimum zoom. It also finds the center point for the geometry being used. However, you can customize all Esri map properties using their API, described at:

https://developers.arcgis.com/javascript/jsapi/map-amd.html#map1

You can also use several general WebFOCUS chart properties to enhance the charts.

Understanding Map Layers

On a map chart, a base layer displays the map tile. Overlay layers add data on top of the base map.

The WebFOCUS request output provides the measure values and the location names or latitude and longitude values of the areas to be overlaid with colors or symbols. The colored areas or symbols form one overlay layer.

The Esri map engine provides additional overlay layers that you can choose to add to the map chart. These additional overlay layers can be either:

- **Tile (demographic) layers.** These layers consist of pre-aggregated demographic data available on an Esri server, which may be a local server.
- **Dynamic feature layers.** These layers provide demographic data that is aggregated dynamically.
- **Reference Feature layers.** These layers place symbols on the maps. WebFOCUS uses them to outline areas on the map.

WebFOCUS supports one base layer and one WebFOCUS overlay layer. If you use the Esri map engine, you can add multiple demographic and feature layers.

WebFOCUS is licensed to access Esri ArcGIS maps, which exist on an Esri server or a local server from Esri. If you are licensed to use a different map provider, you can access their maps instead.

In addition, free Mapbox© OpenStreet© maps are installed with WebFOCUS. You can use them for Leaflet map charts, if you are not able to import the Esri maps into your system.

The connection between geolocations and the fields in your data source are implemented by specifying a URL to a JSON, ESRI, or geoJSON location file (also called an endpoint) for Leaflet maps, and a geoJSON or ESRI endpoint for Esri maps.

WebFOCUS provides JSON location files, but you can add additional locations or different types of location files by adjusting the map properties to point to the files you want.

Location endpoints contain lists of major political subdivisions (such as state and city) along with the latitudes and longitudes that define the borders of the subdivisions.

Generating Esri Map Charts

An Esri chart is defined using a set of extensions (pointers to external chart components) and properties in the GRAPH_JS block of the WebFOCUS StyleSheet. Using these StyleSheet attributes, you specify the base map to use, any demographic and feature layers to add, and the properties of the overlay layer generated by the WebFOCUS request output.

There are two types of syntax you can use for generating Esri map charts:

- **Chart engine syntax.** With this syntax, you add all of the properties Esri needs in order to return the correct geography objects and images.

  When you use this syntax, you must add parameters and pointers to the map service needed to retrieve the geographies, demographic layers, feature layers, and basemaps from Esri to your WebFOCUS StyleSheet.

For geolocation-based charts, you have to supply the URL for the location endpoint to be used with the request, and the geometry locator field. For bubblemap requests based on latitude and longitude values, no endpoint is necessary, as WebFOCUS can plot the coordinates for the symbols.

For more information about the chart engine syntax, see *Generating Esri Map Charts With Chart Engine Syntax* on page 846.

- **Reporting Server syntax.** The Reporting Server comes with a geographic configuration file that contains properties for all standard geographic roles, basemaps, demographic layers, and reference layers. Using this configuration file, the number of properties you have to specify in the WebFOCUS StyleSheet is greatly reduced. The Reporting Server uses this configuration file to generate the underlying chart engine syntax. You can edit the geographic configuration file to customize the list of geographic roles, basemaps, and layers.

  For more information about Reporting Server syntax, see *Generating Esri Map Charts With Reporting Server Syntax* on page 823.

This section describes the required properties and some of the most useful additional properties for Esri map charts. In addition to the required properties, you can include any properties available through the Esri API. These will be passed directly to Esri for processing. The Esri API is described at:

https://developers.arcgis.com/javascript/jsapi/map-amd.html#map1

Every Esri endpoint represents a specific geometry. For example, the World Countries endpoint specifies country boundaries:

http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/World_Countries_(Generalized)/FeatureServer/0

A request that uses this endpoint can only map locations that represent countries, so the sort field assigned to the *location* attribute category should be a field containing country values. If you follow the URL for an endpoint, you will see the names of the geometry locator fields for the endpoint.

Syntax: How to Specify WebFOCUS StyleSheet Attributes for Esri Charts

The TYPE=REPORT declaration in the WebFOCUS StyleSheet must point to the Esri map provider:

```
TYPE=REPORT, CHART-LOOK=com.esri.map, $
```

In addition, you must assign the fields in the request to attribute categories, as described in *Attribute Categories for Map Charts* on page 818, and add Esri map extensions to the StyleSheet, as described in the following sections.

Generating Esri Map Charts With Reporting Server Syntax

The Reporting Server can process Esri map chart requests using a geographic configuration file that contains all of the parameters and pointers for the standard geographies, layers, and basemaps provided by Esri. You can customize objects available by editing this configuration file.

Defining Esri Map Layers Using Reporting Server Syntax

A WebFOCUS Esri map chart has one base layer that displays a background map image and an array of overlay layers. One of the overlay layers contains the output from the WebFOCUS request, which provides the locations or coordinates for placing the colored polygons or symbols. Other types of overlay layers include demographic layers and feature layers.

All supported geographic roles, demographic layers, feature layers, and reference layers are configured in the geo_services.xml file on the Reporting Server, as described in *The Reporting Server Geographic Configuration File* on page 836. The Reporting Server uses the sort field in the chart request to identify the geographic role, which is then used to determine the map service needed based on the URI parameters configured in the geographic configuration file. You can customize the list of geographic roles and layers by editing the geographic configuration file.

Syntax: How to Specify Esri Map Extensions

The TYPE=REPORT declaration in the WebFOCUS StyleSheet must point to the Esri map provider:

```
TYPE=REPORT, CHART-LOOK=com.esri.map, $
```

You must also add an extensions object that defines the base layer and overlay layers, in any order, to the GRAPH_JS_FINAL block of the WebFOCUS StyleSheet.

```
"extensions": {
     "com.esri.map": {
        "overlayLayers": [
          array_of_overlay_layers
                        ],
       "baseMapInfo": {
           basemapproperties
                    }
                    }
             }
```

The properties needed to define each type of layer are described in the following sections.

Syntax: How to Define a Base Layer Using Reporting Server Syntax

To select a basemap from the list of basemaps in the geo_services.xml file, you must specify the value from the name property in the basemap object.

```
"baseMapInfo": {
"customBaseMaps": [
       {
        "ibiBaseLayer": "basemapname"
       }
                  ]
             }
```

where:

`"ibiBaseLayer": "basemapname"`

Specifies the basemap name as configured in the geo_services.xml file. For example:

```
"ibiBaseLayer": "gray"
```

Only one ibiBaseLayer is supported in a request. However, you can display a basemap control with multiple basemaps and switch between them, as described in *How to Add Basemap Information to an Esri Map Chart* on page 895.

Syntax: How to Define a Choropleth Data Layer Using Reporting Server Syntax

To define a data layer for a choropleth, the field to be used as the location dimension must be the BY field in the chart request and must be assigned a geographic role in the Master File, or in a DEFINE command. The geographic role assigned must match a role id value in the geo_services.xml file.

```
{
    "ibiDataLayer": {
              "map-metadata": {
                   "map_by_field": "fieldname"
                               }
                    }
}
```

where:

`"map-metadata"`

Specifies that the BY field in the request represents a geography object and has been assigned a geographic role in the metadata.

`"map_by_field": "`*`fieldname`*`"`

Specifies the name of the BY field in the request. Must be the same field name specified in the BY phrase in the chart request.

Note: This field is not assigned to an attribute category in the WebFOCUS StyleSheet. Instead, it is assigned to the "map_by_field" property in the "ibiDataLayer" object.

Example: Generating a Choropleth Esri Map Using Reporting Server Syntax

The following request will use the field STATE_PROV_NAME as its BY field. In the WF_RETAIL_GEOGRAPHY Master File, this field is assigned the geographic role STATE:

```
FIELDNAME=STATE_PROV_NAME, ALIAS=STATE_PROV_NAME, USAGE=A30V, ACTUAL=A30V,
      MISSING=ON,
      TITLE='State,Province', DESCRIPTION='Name of State or Province',
      GEOGRAPHIC_ROLE=STATE,  $
```

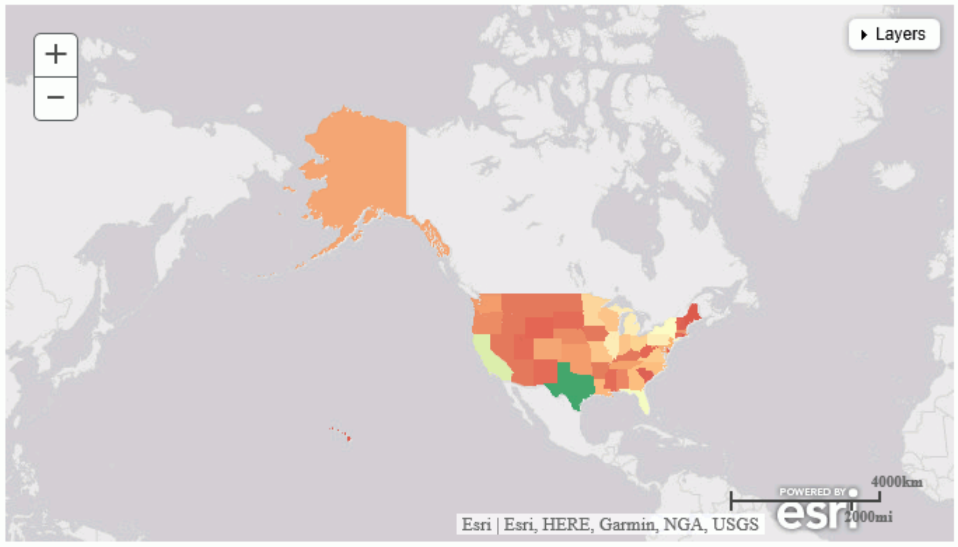
The geographic role STATE and the basemap "gray" are also defined in the geo_services.xml file:

```
<geo_role id="STATE" type="alpha" role_name="STATE"
  role_name_title="State" role_format="NAME" role_format_title="Name"
geo_type="geography" unified="yes" > ...

<basemap custom="false" default="true" name="gray" title="Light Gray Canvas
Map" icon="qb/gray_map_108x72.png" />
```

In the following request, the BY field is STATE_PROV_NAME and the polygons are colored by the COGS_US field.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "STATE_PROV_NAME"
                                              }
                                   }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
             ]
          }
                          }
                  }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Syntax: **How to Define a Proportional Symbol Data Layer Using Reporting Server Syntax**

To define a data layer for a bubblemap, the field to be used as the location dimension must be the BY field in the chart request and must be a JSON geometry point. You can create a geometry point in a DEFINE command using the GIS_POINT function described in the *Using Functions* manual.

```
{
    "ibiDataLayer": {
              "map-geometry": {
                   "map_by_field": "fieldname"
                               }
                    }
}
```

where:

`"map-geometry"`

Specifies that the BY field in the request represents a geometry point.

`"map_by_field": "fieldname"`

Specifies the name of the BY field in the request. Must be the same field name specified in the BY phrase in the chart request. This field is not assigned to an attribute category in the WebFOCUS StyleSheet.

Example: Generating a Proportional Symbol Esri Map Using Reporting Server Syntax

The following request creates a geometry point using the fields STATE_PROV_CAPITAL_LONGITUDE and STATE_PROV_CAPITAL_LATITUDE and uses that point as its BY field.

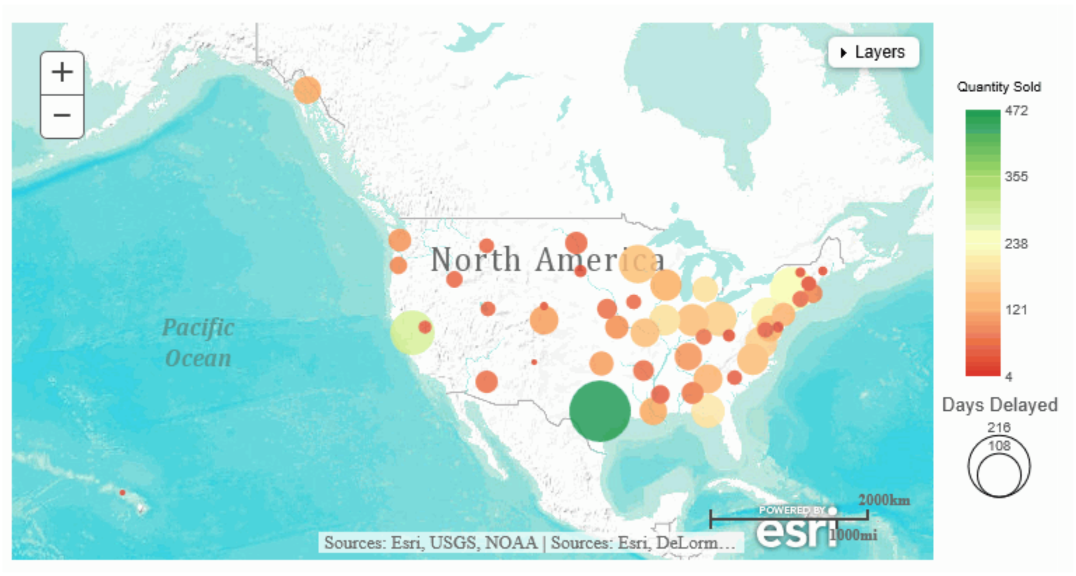
The basemap, "terrain", is defined in the geo_services.xml file.

```
 <basemap custom="false" default="false" name="terrain" title="Terrain with
Labels" icon="qb/terrain_map_108x72.png" />
```

In the following request, the BY field is GEOPOINT, created in the DEFINE FILE command, the symbols are sized by the DAYSDELAYED field and colored by the QUANTITY_SOLD field.

```
DEFINE FILE WF_RETAIL_LITE
GEOPOINT/A200 = GIS_POINT('4326', STATE_PROV_CAPITAL_LONGITUDE,
STATE_PROV_CAPITAL_LATITUDE);
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY GEOPOINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *

INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=DAYSDELAYED, BUCKET=size, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "GEOPOINT"
                                                }
                                       }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "terrain"
                }
            ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Syntax: **How to Add a Demographic or Feature Layer To an Esri Map Chart Using Reporting Server Syntax**

To add a demographic or reference feature layer to an Esri chart, you must add a layer to the overlay layers that supplies the name of the demographic or reference layer, as defined in the geo_services.xml file.

```
{
    "ibiAddLayer": "layer_name"
}
```

where:

`"ibiAddLayer": "`*`layer_name`*`"`

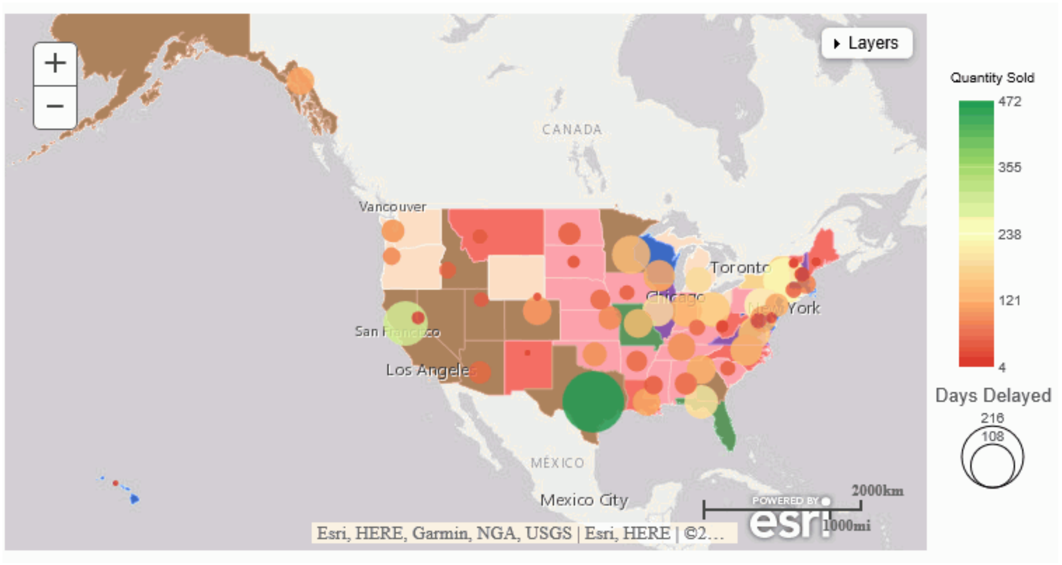
Specifies the name of the demographic or reference layer using the value of the name property from the geo_services.xml file.

Example: Adding a Demographic Layer to an Esri Map Using Reporting Server Syntax

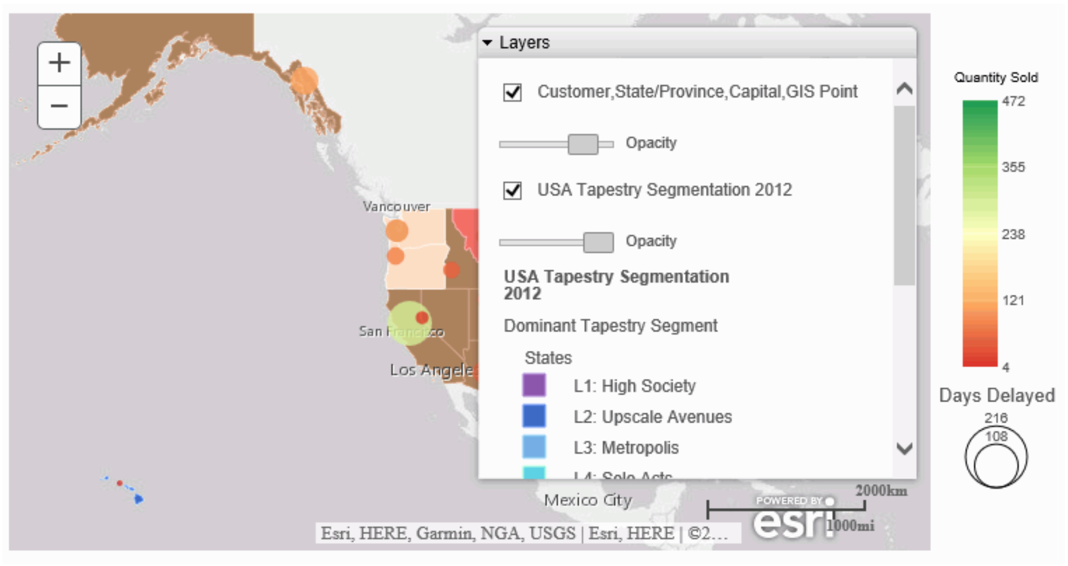
In the following request, the BY field is STATE_PROV_CAPITAL_POINT, created in a DEFINE in the WF_RETAIL_GEOGRAPHY Master File and assigned the geographic role GEOMETRY_POINT. The demographic layer named USA_Tapestry_Segmentation_2012 is added to the chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY STATE_PROV_CAPITAL_POINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
 ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "STATE_PROV_CAPITAL_POINT"
                                            }
                                   }
         },
         {"ibiAddLayer": "USA_Tapestry_Segmentation_2012"}
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
             ]
          }
                         }
                  }
*END
ENDSTYLE
END
```

The output is shown in the following image. The proportional symbols represent the data layer and the polygons represent the demographic layer.



You can open the table of contents to see the legend for the demographic layer, to hide a layer, or to change the opacity of a layer, as shown in the following image.



☑ Customer,State/Province,Capital,GIS Point

Opacity

☑ USA Tapestry Segmentation 2012

Opacity

USA Tapestry Segmentation 2012

Dominant Tapestry Segment

States

🟪 L1: High Society

🟦 L2: Upscale Avenues

🟦 L3: Metropolis

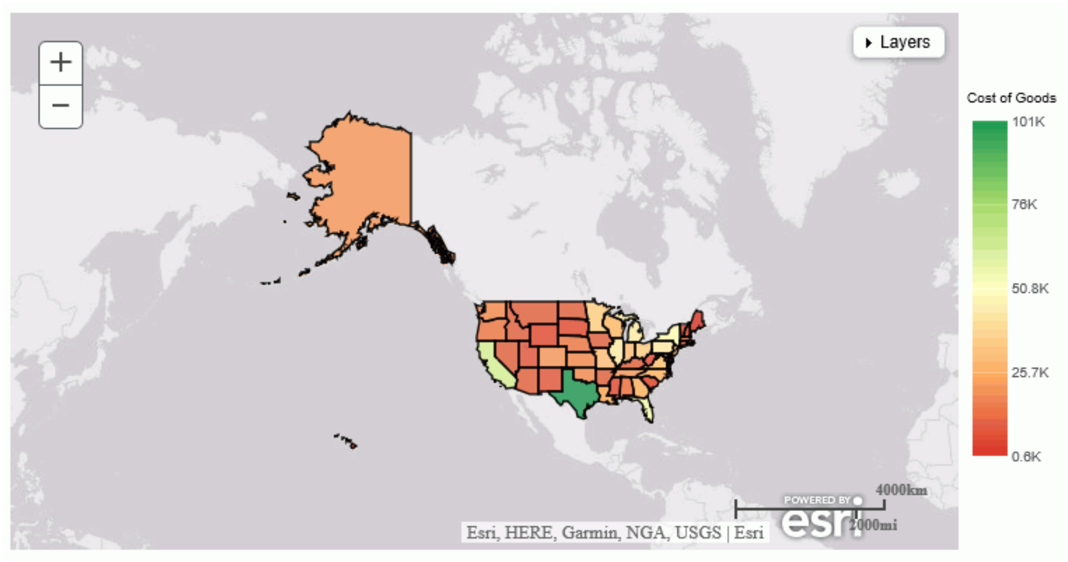
🟦 L4: Solo Acts

Example: Adding a Reference Feature Layer to an Esri Map Using Reporting Server Syntax

The following request generates a choropleth Esri map chart using the BY field STATE_PROV_NAME. The reference feature layer named USA_States_Gen is added to the chart.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "STATE_PROV_NAME"
                                     }
                            }
         },
       {"ibiAddLayer": "USA_States_Gen"}
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
            ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Syntax: How to Change an Esri Map Layer Title Using Reporting Server Syntax

By default, the title for each layer in the Table of Contents (Layers) control is taken from the geographic configuration file. You can change the title by adding a "title" property to the layer properties.

```
 "overlayLayers":
     [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "fieldname"
                                         }
                                },
                    "title": "data_layer_title"
         },
         {"ibiAddLayer": "layer name",
                    "title": "context_ref_layer title"
         }
       ],
```

where:

`"title": "data_layer_title"`

Specifies a new title for the data layer.

`"title": "`*`context_ref_layer title`*`"`

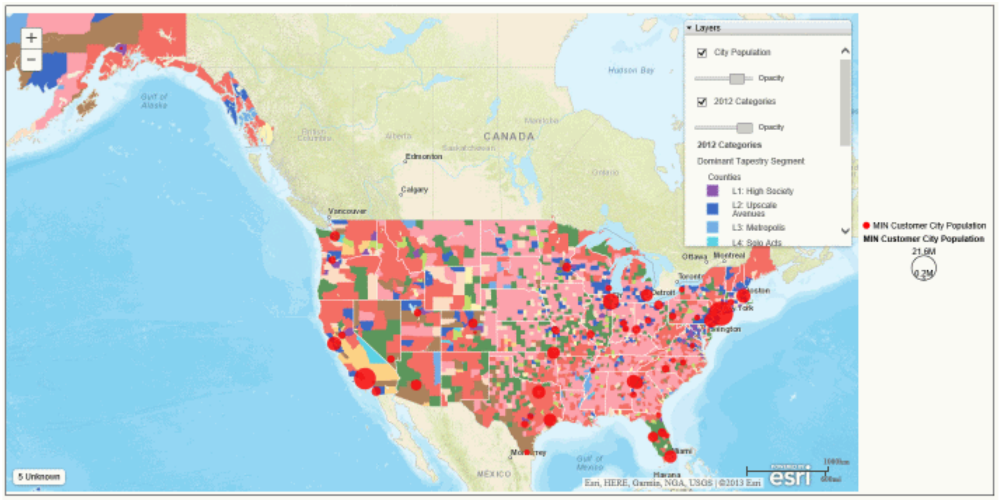
Specifies a new title for a demographic or reference layer.

Example: Changing Esri Map Layer Titles Using Reporting Server Syntax

The following request changes the data layer title to "City Population" and the demographic layer title to "2012 Categories".

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY CITY_NAME
WHERE CITY_POPULATION NE  ' ' OR MISSING
WHERE COUNTRY_NAME EQ 'United States'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS_FINAL
"bubbleMarker": {"maxSize":"5%", "color": "red"},

"extensions": {
  "com.esri.map": {
      "overlayLayers":
     [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "CITY_NAME"
                                            }
                                   },
                    "title": "City Population"
         },
         {"ibiAddLayer": "USA_Tapestry_Segmentation_2012",
           "title": "2012 Categories"
         }
       ],

      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "streets"
                }
             ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.



The Reporting Server Geographic Configuration File

The Reporting Server comes with a configuration file (geo_services.xml) that contains elements that describe all of the geographic roles, geographic hierarchies, URLs to the map services, and base maps available to the Esri map viewer. This file is located in the catalog directory under the server home directory.

```
_edahome/catalog/
```

Note: You can show this directory on the application tree by clicking *Show edahome, edaconf, edaprfu, scaroot, edatemp, edalog and foccache for all users* on the Filter menu of the Reporting Server Console Applications page. Then, you can edit the geo_services.xml file directly in the Reporting Server Console.

The geographic role selections that you use in your requests are built dynamically using this configuration file. Each role definition in the configuration file, when referenced in an Esri chart request, generates metadata and a request that is sent to Esri in order to download the appropriate map and place the markers or geography objects on the map.

A geographic role can be part of a hierarchy. For example, the World geographic role is at the top of a hierarchy that contains continents, countries, states, and cities. These hierarchies are also described in the geo_services.xml file.

In the same application directory that contains the geo_services.xml file is a series of report requests that display information about the items configured in the file.

To add a custom geographic role, you must add the necessary parameters for the geographic role to this file.

Following standard XML syntax rules, each element is enclosed in element start and end tags (<elementname>, </elementname>), and attribute values are enclosed in double quotation marks (").

Reference: Geographic Role Definitions

A geographic role is stored as a *geo_role* element in the geo_roles object of the geo_services.xml file. A geographic role must be defined with:

- ❑ An ID that will identify the role in the configuration file.
- ❑ A format and length for the data to be returned.
- ❑ A role name.
- ❑ A display title for the role name (to appear as a selection in the App Studio Settings panel).
- ❑ An optional role format (if the role can have multiple formats, such as a name and an abbreviation).
- ❑ A display title for the format.
- ❑ A role type (*geography* for geographic objects such as country or state, *geometry* for geometry points, geometry areas, or geometry lines, or *coordinate* for longitude or latitude).
- ❑ An optional vocabulary rule element containing vocabulary elements for associating the role with a field in the metadata.

The syntax is

```
<geo_role id="id" value_size="size" unified="false" role_name="rname"
        role_name_title="rname_title" role_format="rformat"
        role_format_title="rformat_title" geo_type="gtype"
        type="datatype" >
        <vocabulary_rules>
            <vocabulary>vrule</vocabulary>
         </vocabulary_rules>
      </geo_role>
```

where:

`id="id"`

Is an alphanumeric uppercase value, up to 50 characters, used to identify the geographic role.

`type="`*`datatype`*`"`

Is the data type for the ID. Can be one of the following.

- **"alpha".** For alphanumeric data, formats A*n* or I*n*.
- **"integer".** For integer numeric data, format I*n*.
- **"numeric".** For fractional numeric data, formats P*n*.*m*, D*n*.*m*, or F*n*.*m*.
- **"text".** For text data, format TX*n*.

`value_size="`*`size`*`"`

Is the optional number of characters in USAGE format length (any, if not set).

`role_name="`*`rname`*`"`

Is the name of the geographic role.

`role_name_title="`*`rname_title`*`"`

Is the title of the geographic role, to be displayed in the Settings panel for selection.

`role_format="`*`rformat`*`"`

Is an optional format for the geographic role, useful when the role can be referenced using multiple formats, such as a name, an ISO code, and an abbreviation. Standard role titles include the following.

- **NAME.** Specifies that the role defines the name of a geographic entity, such as Florida.
- **ABBR.** Specifies that the role defines an abbreviation, such as FL.
- **ZIP3.** Specifies that the role defines a 3-digit zip code.
- **ZIP5.** Specifies that the role defines a 5-digit zip code.
- **FIPS.** Specifies that the role defines a FIPS (Federal Information Processing Standard) code.
- **ISO2.** Specifies that the role defines an ISO 3166-2 code published by the International Organization for Standardization (ISO).
- **ISO3.** Specifies that the role defines an ISO 3166-3 code published by the International Organization for Standardization (ISO).
- **LINE.** Specifies that the role defines one line of a role, such as an address line.
- **FULL.** Specifies that the role defines a full geographic role, such as a full address.

`role_format_title="`*`rformat_title`*`"`

Is an optional title for the format of the geographic role. It will be shown in parentheses along with the role title in the Settings panel, for example, State (Abbreviation).

`geo_type="`*`gtype`*`"`

Is one of the following predefined role types.

- **"geography".** For geographic objects such as country or state.
- **"geometry".** For geometry objects such as geometry point, geometry line, and geometry area.
- **"coordinate".** For coordinates such as latitude and longitude.

`<vocabulary>`*`vrule`*`</vocabulary>`

Is an element that consists of a group of vocabulary elements that explicitly describe column names for the geographic role. These rules will be used to select the best geographic data for the role.

Elements in a rule are connected by the Boolean logic operation OR (only one needs to be satisfied). Each vocabulary element contains words enclosed with special characters. Words in the rule element are connected by the Boolean logic operation AND (all need to be satisfied).

A word may be prefixed and/or suffixed with the percent character (%), which is a place holder for any sequence of characters. If an element contains more than one word, each word has to be prefixed by the character plus (+) or minus (-). Plus indicates that the word must be found in the column name. Minus indicates that word *must not* be found in the column name.

You can add a custom geographic role by supplying a geo_role element with the necessary parameters to the geo_roles object in the geo_services.xml file. Once you have added a geographic role definition, you can run the geo_srv_roles procedure to see that the parameters were added correctly.

Example: Sample Geographic Role Definitions

The following defines the State Abbreviation geographic role. The role ID is USSTATE_ABBR. The role name is USSTATE with a role format of ABBR. The titles that show in the Settings panel are US state (Abbreviation). The format is A2, and the vocabulary rules specify that the characters *state* must be present, but the characters *iso*, *capital*, and *population* must not be present. The geo type is geography, indicating that the returned data will be a geographic area.

```
<geo_role
id="USSTATE_ABBR"
 value_size="2"
 type="alpha"
role_name="USSTATE"
role_name_title="US state"
role_format="ABBR"
role_format_title="Abbreviation"
 geo_type="geography">
<vocabulary_rules>
<vocabulary>+%state%-%iso%-%capital%-%population%</vocabulary>
</vocabulary_rules>
</geo_role>
```

The following is a role definition for latitude values. The role ID is LATITUDE. The role name is also LATITUDE. Its format is numeric. The title that displays in the Settings panel is Latitude. The geo type is coordinate, indicating that the returned data will be points. The vocabulary rules specify that the characters *latitude* must be present.

```
<geo_role
 id="LATITUDE"
 type="numeric"
 role_name="LATITUDE"
role_name_title="Latitude"
 geo_type="coordinate">
 <vocabulary_rules>
  <vocabulary>%latitude%</vocabulary>
 </vocabulary_rules>
</geo_role>
```

The following is the definition for the city role. The ID is CITY. The role name is also CITY. Its format is NAME. The title that displays in the Settings panel is City (Name). The definition has a set of vocabulary elements. Only one of the elements in the list must be true. Therefore, the characters *city*, or *town*, or *country* plus *capital*, or *state* plus *capital* must be present.

```
<geo_role
 id="CITY"
type="alpha"
role_name="CITY"
role_name_title="City"
role_format="NAME"
role_format_title="Name"
 geo_type="geography">
 <vocabulary_rules>
  <vocabulary>+%city%-%population%</vocabulary>
  <vocabulary>+%town%-%population%</vocabulary>
  <vocabulary>+%country%+%capital%-%population%</vocabulary>
  <vocabulary>+%state%+%capital%-%population%</vocabulary>
 </vocabulary_rules>
</geo_role>
```

Reference: Geographic Role URI Definitions

If you add a custom geographic role to the geo_services.xml file, you must also add its URI to the <URIS> object. The URI objects point to the map services for specific geographic roles. The syntax is.

```
<uri description="description">
   <returned_geometry>type </returned_geometry>
   <returned_georole>role</returned_georole>
   <url type="esri" authorization="auth" synonym="">
         "url_to_georole"
   <parameters>
    <parm order="number" parm_name="pname" parm_georole="parmrole"/>
   </parameters>
  </uri>
```

where:

`"`*`description`*`"`

Is a description for the geographic role the URI is pointing to.

`type`

Is any supported geometry type, such as GEOMETRY AREA.

`role`

Is the name of the returned geographic role.

"*auth*"

Is the type of authorization needed to access this geographic role. Valid values are:

- ❑ **silent.** Credentials for your ArcGIS application are provided in the connection string of the Adapter for Esri.
- ❑ **none.** No authorization is needed.
- ❑ **named.** User credentials are provided in the connection string of the Adapter for Esri.
- ❑ **on premise.** User credentials for a locally hosted ArcGIS server are provided in the connection string of the Adapter for Esri.

"*url_to_georole*"

Is the URL for the geographic role.

parm order="*number*"

Is the number of a parameter needed to retrieve the correct geographic role.

parm_name="*pname*"

Is the name of the parameter associated with *parm order*.

Note that the BY field in the request has to correspond to the first parameter. The parameters are used to match the geographic roles with the appropriate URI definitions.

parm_georole="*parmrole*"

Is the name of the geographic role associated with *parm order*.

Once you have added the URI, you can run the geo_srv_map_uris procedure to see that the parameters were added correctly.

Reference: Basemap Definitions

Following are some sample standard basemap definitions from the geo_services.xml file.

```
<BASEMAPS>
...
 <basemap custom="false" default="false" name="streets"
    title="World Street Map" icon="qb/streets_map_108x72.png" />
 <basemap custom="false" default="false" name="satellite"
    title="World Imagery" icon="qb/imagery_map_108x72.png" />
 <basemap custom="false" default="false" name="terrain"
    title="Terrain with Labels" icon="qb/terrain_map_108x72.png" />
 <basemap custom="false" default="true" name="gray"
    title="Light Gray Canvas Map" icon="qb/gray_map_108x72.png" />
...
</BASEMAPS>
```

The properties defined for the standard basemaps are *custom* (which is *false* for the standard basemaps), *default*, *name*, *title*, and *icon*.

To define a custom basemap you must define these properties for your custom basemap, and add a URL that points to the map service that has the basemap image.

Note: Any basemap you add must be a tiled map layer.

You can also customize the list of basemaps by deleting basemap definitions from the <basemaps> object. To configure Esri on Premise, you should remove any basemap definition that references a URL that points to a location outside of your locally hosted Esri server.

The syntax for a basemap definition is

```
<basemap custom="boolean" default="boolean" name="mapname"
  title="maptitle" icon="url_to_icon" url="url_to_basemap"
/>
```

where:

`custom="boolean"`
: Specifies whether the basemap is standard or custom. Valid values are true and false. To add a custom basemap, specify custom="true".

`default="boolean"`
: Specifies whether this basemap should be the default basemap if the user does not select a basemap. Valid values are true and false.

`name="mapname"`
: Is a name for the basemap.

`title="maptitle"`
: Is a title that will display in the Basemap drop-down list in the Properties panel of the Esri Viewer.

`icon="url_to_icon"`
: Is the location of a thumbnail image for the basemap.

`url="url_to_basemap"`
: Is the URL to the map service that provides the basemap.

You can add a custom basemap by adding a basemap element with the necessary parameters to the BASEMAPS object of the geo_services.xml file. After you add the basemap definition, you can run the geo_srv_basemaps procedure to see if the parameters were added correctly.

Reference: Demographic Layer Definitions

A demographic layer displays data about populations. Two types of demographic layers are configured in geo_services.xml. One type of layer provides cached data. The layerType for this type of demographic layer is "tile". The other type of demographic layer dynamically renders the population data. The layerType for this type of demographic layer is featurelayer. Each demographic layer is defined as a context layer within a context layer group in the CONTEXTLAYERGROUPS object in the geo_services.xml file. The following syntax shows a sample demographic context layer.

```
<CONTEXTLAYERGROUPS>
    <contextlayergroup name="USA_Population" title="USA Population" >
      <contextlayer name="USA_Population_Density_2012" authorization="none"
         layerType="tile" title="USA Population Density 2012" >
        <uri>http://server.arcgisonline.com/ArcGIS/rest/services/
Demographics/USA_Population_Density/MapServer</uri>
      </contextlayer>
```

The syntax is

```
 <contextlayer name="string" authorization="string"
           layerType="string" title="string" >
           <uri>uri_to_map_server</uri>
      </contextlayer>
```

where:

`name="`*`string`*`"`

Is the name of the layer to be referenced in requests.

`authorization="`*`string`*`"`

Specifies the type of authorization needed to access the layer. Valid values are:

- **silent.** Credentials for your ArcGIS application are provided in the connection string of the Adapter for Esri.
- **none.** No authorization is needed.
- **named.** User credentials are provided in the connection string of the Adapter for Esri.
- **on premise.** User credentials for a locally hosted ArcGIS server are provided in the connection string of the Adapter for Esri.

`layerType="`*`string`*`"`

For a cached demographic layer, the layerType value is "tile". For a dynamic demographic layer, the layerType is "featurelayer".

uri_to_map_server

Is the URI to the map server for the demographic layer.

Reference: Reference Feature Layer Definitions

A reference feature layer adds outlines to geographical features on the map. Each reference layer is defined as a context layer within the context layer group named Ref_layers in the CONTEXTLAYERGROUPS object of the geo_services.xml file. The following syntax shows a sample reference layer. For a reference layer, the layerType is "featurelayer".

```
<CONTEXTLAYERGROUPS>
    <contextlayergroup name="Ref_layers" title="Reference layers" >
      <contextlayer name="USA_ZIP" authorization="none"
layerType="featurelayer" title="USA ZIP" >
      <uri>http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/
USA_ZIP_Code_Points_analysis/FeatureServer/0</uri>
    </contextlayer>
```

The syntax is

```
 <contextlayer name="string" authorization="string"
           layerType="featurelayer" title="string" >
           <uri>uri_to_map_server</uri>
      </contextlayer>
```

where:

name="*string*"

Is the name of the layer to be referenced in requests.

authorization="*string*"

Specifies the type of authorization needed to access the layer. Valid values are:

- **silent.** Credentials for your ArcGIS application are provided in the connection string of the Adapter for Esri.
- **none.** No authorization is needed.
- **named.** User credentials are provided in the connection string of the Adapter for Esri.
- **on premise.** User credentials for a locally hosted ArcGIS server are provided in the connection string of the Adapter for Esri.

layerType="featurelayer"

For a reference layer, the layerType value is "featurelayer".

uri_to_map_server

Is the URI to the map server for the feature layer.

Generating Esri Map Charts With Chart Engine Syntax

The chart engine syntax for generating Esri map charts includes parameters and pointers to the map service needed to retrieve the geographies, demographic layers, feature layers, and basemaps from Esri.

The properties for each type of layer are described in the following sections.

Syntax: How to Specify Esri Map Extensions Using Chart Engine Syntax

All layers in the chart and the objects within them are defined within the Esri map extensions properties of the WebFOCUS StyleSheet. An extension is a block that incorporates external data into the request.

WebFOCUS supports one base layer and an array of overlay layers. One of the overlay layers contains the output from the WebFOCUS request, with the locations or coordinates for placing the colored polygons or symbols.

Properties for overlay layers that are based on a location field are different from those that are based on longitude and latitude fields. Requests based on a location field require a location file (endpoint) that lists the longitude and latitude values for the borders of each location value. You also need to specify the geometry locator field within the endpoint. No endpoint is needed for requests that supply longitude and latitude values, as WebFOCUS can plot the coordinates.

Demographic layers use their own endpoints and do not require you to specify a geometry locator field.

You can define the base and overlay layers in either order. In the following syntax, the overlay layers are defined first:

```
"extensions": {
    "com.esri.map": {
        "overlayLayers": [
           array_of_overlay_layers
            ],
        "baseLayer": {
            basemapproperties
                 }
   }
}
```

Syntax: How to Define a Base Layer Using Chart Engine Syntax

Only the base map name is required for generating a base layer:

```
"baseLayer": {
     "title": "string",
     "basemap": "string"
              }
```

where:

`"title": "string"`

Is an optional title for the layer.

`"basemap": "string"`

Is the name of one of the supported base maps. Valid values are:

- **dark-gray.** Provides a dark background that allows the focus to be on other layers.
- **gray.** Provides a light gray neutral background.
- **hybrid.** Provides a detailed World Imagery map layer with labels.
- **national-geographic.** Provides a general reference map from National Geographic.
- **oceans.** Provides a background designed to be used as a reference map for ocean data.
- **osm.** Provides the OpenStreetMap community map layer.
- **satellite.** Provides a detailed World Imagery map layer.
- **streets.** Provides a multiscale street map for the world.
- **terrain.** Provides a terrain base map with labels.
- **topo.** Provides a topographic map.

Note: Additional properties for the base layer, such as the initial center point and zoom level, are described in *Customizing Esri Map Output* on page 879.

Syntax: How to Define an Overlay Layer Based on a Location Field Using Chart Engine Syntax

WebFOCUS request output and feature layers are defined based on a location field using the following properties:

```
{
   "title": "string",
   "layerType": "string",
   "quantizationThreshold": number,
   "geometrySourceType": "string",
   "geometryLocateField": ["string"],
   "geometryDataField": "name",
   "url": "url_to_endpoint"
   }
```

where:

`"title": "string"`

Is a title to display on the table of contents control on the map.

`"layerType": "string"`

Is the type of layer. Valid values are:

- **choropleth.** Displays a WebFOCUS choropleth layer.
- **bubble.** Displays a WebFOCUS bubblemap layer.
- **featurelayer.** Displays outlined locations on a map. For information about editing the properties of the symbol used to outline locations on a reference feature layer, see *How to Format a Reference Feature Layer Symbol* on page 879. For information about supplying smartMapping properties that create renderers for styling a reference feature layer or dynamic map service layer, see *How to Style a Dynamic Map Service Layer or Reference Feature Layer Using Chart Engine Syntax* on page 863.
- **tile.** Displays pre-aggregated and cached demographic data. For information about demographic tile layers, see *How to Define a Demographic Overlay Layer Using Chart Engine Syntax* on page 859.

`"quantizationThreshold": number`

Is a threshold value in feet for drawing small locations. Without this property, small ZIP codes whose length or width is smaller than the threshold value may not be drawn. If the geometry returned is smaller than the quantizationThreshold value, the query is submitted again with no quantizationParameters. In this way smaller ZIP codes are painted properly. The default value is 10560 feet (2 miles / 3.2 km).

`"geometrySourceType": "`*`string`*`"`

Specifies the type of location file being used. Valid values are:

- esri
- geojson

`"geometryLocateField": ["`*`string`*`"]`

Is the geography locator field name in the location endpoint. If you follow the URL to the endpoint, the field names will be documented.

`"url": "`*`url_to_endpoint`*`"`

Is the URL for the location endpoint.

Sample Endpoints

The following endpoints are free and accessible from the WebFOCUS tools immediately after WebFOCUS installation. You may have access to additional endpoints, in which case you can point to those URLs.

World Countries:

http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/World_Countries_(Generalized)/FeatureServer/0

World Cities:

http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/World_Cities/FeatureServer/0

US Zipcodes - 3 digit:

http://services1.arcgis.com/szcgiQwor8m0ZTum/arcgis/rest/services/USA_3zip_codes/FeatureServer/0

US Zipcodes - 5 digit:

http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/USA_ZIP_Codes_2014/FeatureServer/0

US States:

http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/USA_States_Generalized/FeatureServer/0

US Cities:

http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/USA_Major_Cities/FeatureServer/0

US Counties:

http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/USA_Counties_Generalized/FeatureServer/0

Provinces and Territories of Canada

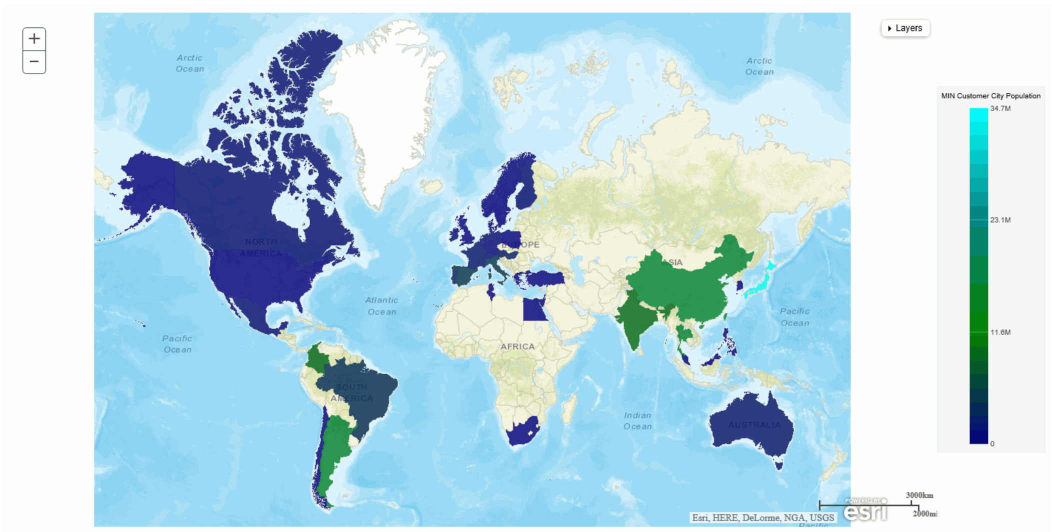
http://services.arcgis.com/zmLUiqh7X11gGV2d/ArcGIS/rest/services/Canada_Provinical_boundaries_generalized/FeatureServer/0

Example: Defining a WebFOCUS Overlay Layer Based on a Location Field Using Chart Engine Syntax

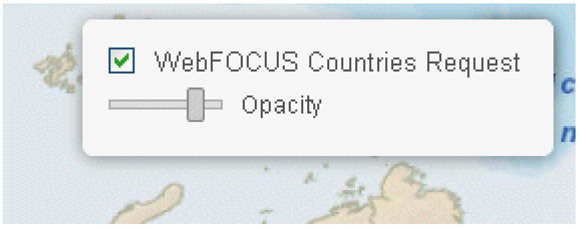
The following request generates a choropleth using the *streets* base layer and the World Countries endpoint:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=color, $
*GRAPH_JS
"legend": {
    "visible": true
},
"colorScale": {"colors":["navy", "green", "teal", "cyan"]},
"extensions": {
"com.esri.map": {
"overlayLayers":
[
 {
 "title": "WebFOCUS Countries Request",
 "layerType": "choropleth",
 "geometrySourceType": "esri",
 "geometryLocateField": ["Country"],
 "geometryDataField": "name",
 "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
 }
 ],
 "baseLayer": {
 "basemap": "streets"
          }
}
}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Clicking the table of contents control at the top right of the chart shows the title of the overlay layer, a check box for displaying it, and a slider for changing its opacity:

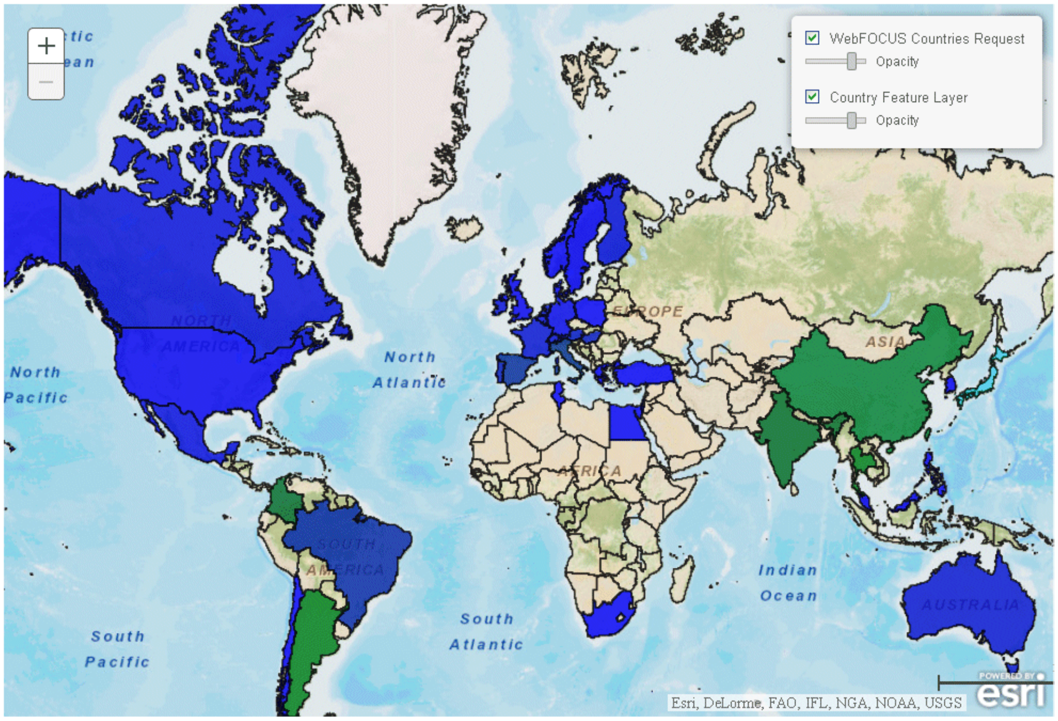


Example: Adding a Reference Feature Layer to a Chart Using Chart Engine Syntax

The following request adds a reference feature layer to a choropleth chart. The feature layer uses the World Countries endpoint to outline the country borders:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY  COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=color, $
*GRAPH_JS
"legend": {"visible": false},
"colorScale": {"colors":["blue", "green", "teal", "cyan"]},
"extensions": {
 "com.esri.map":
 {
  "baseLayer":{ "title": "Base Layer",
  "basemap": "streets"
 },
 "overlayLayers":
 [
  {
   "title": "WebFOCUS Countries Request",
 "layerType": "choropleth",
 "geometrySourceType": "esri",
 "geometryLocateField": ["Country"],
 "geometryDataField": "name",
 "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
 },

{
  "title": "Country Feature Layer",
  "layerType": "featurelayer",
  "geometrySourceType": "esri",
  "geometryLocateField": ["Country"],
  "geometryDataField": "name",
  "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
  }
 ],
 }
}
*END
ENDSTYLE
END
```

The output is shown in the following image. The table of contents control has check boxes and opacity sliders for displaying each overlay layer:



Example: Drawing Small Areas Using the Quantization Threshold Using Chart Engine Syntax

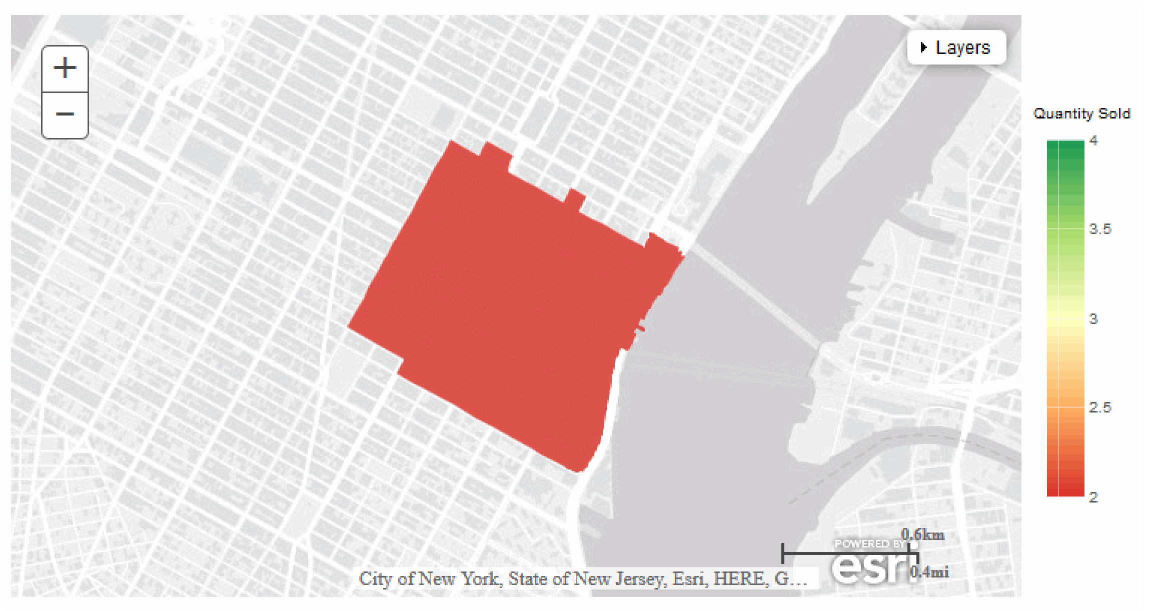
The following request sets the quantizationThreshold property to 1000 and attempts to draw ZIP Code 10116.

```
GRAPH FILE WF_RETAIL_LITE
SUM QUANTITY_SOLD
 BY POSTAL_CODE
 WHERE COUNTRY_NAME EQ 'United States';
 WHERE POSTAL_CODE EQ '10016' ;
 ON GRAPH PCHOLD FORMAT JSCHART
 ON GRAPH SET LOOKGRAPH CHOROPLETH
 ON GRAPH SET STYLE *
 INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
 TYPE=REPORT, ORIENTATION=LANDSCAPE, CHART-LOOK=com.esri.map, $
 TYPE=DATA, COLUMN=N1, BUCKET=location, $
 TYPE=DATA, COLUMN=N2, BUCKET=color, $
 *GRAPH_JS_FINAL
 "extensions": {
 "com.esri.map": {
 "overlayLayers": [
 {
 "layerType": "choropleth",
 "quantizationThreshold": 1000,
 "geometryDataField": "name",
 "geometryLocateField": ["ZIP_CODE"],
 "title": "World Postal Codes (temporary = USA ZIP5)",
 "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/
USA_ZIP_Codes_2015/FeatureServer/0",
 "geometrySourceType": "esri"
 }
 ]
}
},
 "legend":
{ "visible": true }
*END
 ENDSTYLE
 END
```

The output is shown in the following image. Nothing is drawn on the map, and the postal code is listed as unknown.

Layers
Quantity Sold
4
3.5
3
2.5
2
1 Unknown
City of New York, State of New Jersey, Esri, HERE, G...
POWERED BY esri
0.6km
0.4mi

Changing quantizationThreshold to 6000 produces the output shown in the following image. The postal code is now drawn on the map.



Syntax: How to Define an Overlay Layer Based on Longitude and Latitude Fields Using Chart Engine Syntax

A bubblemap can be based on longitude and latitude fields. They should be numeric. If they are not, they will be converted, if possible. No endpoint is used for this type of request. The values needed are in the data (geometrySourceType: "seriesdata"):

```
{
  "title": "string",
  "layerType": "bubble",
  "geometryXY": {
      "y": "lat",
      "x": "lng"
              },
  "geometrySourceType": "seriesdata"
  }
```

where:

`"title": "string"`

Is a title for the layer.

```
"y": "lat", "x": "lng"
```

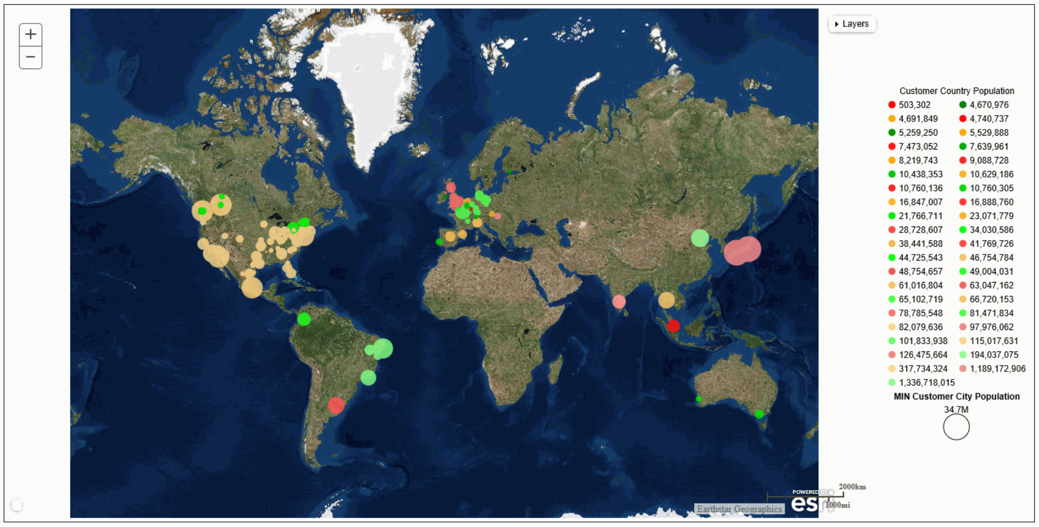
Is required to make the connection to the chart attribute categories.

Example: Defining a WebFOCUS Overlay Layer Based on Longitude and Latitude Fields Using Chart Engine Syntax

The following request generates a bubblemap using the *satellite* base map and the CITY_LONGITUDE and CITY_LATITUDE sort fields. The symbols are colored by the COUNTRY_POPULATION sort field and are sized by the MIN.CITY_POPULATION measure:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY COUNTRY_POPULATION
BY CITY_LATITUDE
BY CITY_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_POPULATION, BUCKET=color, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"bubbleMarker":{"maxSize":"5%"},
"extensions": {
    "com.esri.map": {
        "overlayLayers":             [
                {
                    "title": "Population",
                    "layerType": "bubble",
                    "geometryXY": {
                        "y": "lat",
                        "x": "lng"
                    },
                    "geometrySourceType": "seriesdata"
                }
            ],
        "baseLayer": {
            "basemap": "satellite"
        }    }
}

*END
ENDSTYLE
END
```

The output is shown in the following image:



Syntax: How to Define a Demographic Overlay Layer Using Chart Engine Syntax

You can add demographic overlay layers to an Esri map chart. The layer type is *tile*. You supply a layer title and the URL to the appropriate endpoint. The demographic layer displays as a choropleth without a legend:

```
{
 "title": "string",
 "layerType": "tile",
 "url": "url_to_endpoint"
 }
```

where:

`"title": "string"`

Is a title for the layer.

`"url": "url_to_endpoint"`

Is the URL to the demographic layer endpoint.

Sample Demographic Endpoints

The WebFOCUS tools give you access to the following free demographic layers immediately after WebFOCUS installation. You may have access to other demographic endpoints, in which case you can point to those URLs in the request.

USA Population Density -2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Population_Density/MapServer

USA Projected Population Growth 2012-2017

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Projected_Population_Change/MapServer

USA Population Change 2010-2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Recent_Population_Change/MapServer

USA Population Change 2000-2010

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Recent_Population_Change/MapServer

USA Population Change 1990-2000

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_1990-2000_Population_Change/MapServer

USA Median Household Income 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Median_Household_Income/MapServer

USA Population Older than Age 64

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Percent_Over_64/MapServer

USA Unemployment Rate 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Unemployment_Rate/MapServer

USA Median Home Value 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Median_Home_Value/MapServer

USA Population Younger than Age 18

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Percent_Under_18/MapServer

USA Average Household Size 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Average_Household_Size/MapServer

USA Median Net Worth 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Median_Net_Worth/MapServer

USA Owner Occupied Housing 2012

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Owner_Occupied_Housing/MapServer

USA Labor Force Participation Rate 2010

http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Labor_Force_Participation_Rate/MapServer

USA Recent Population Change 2010-2012

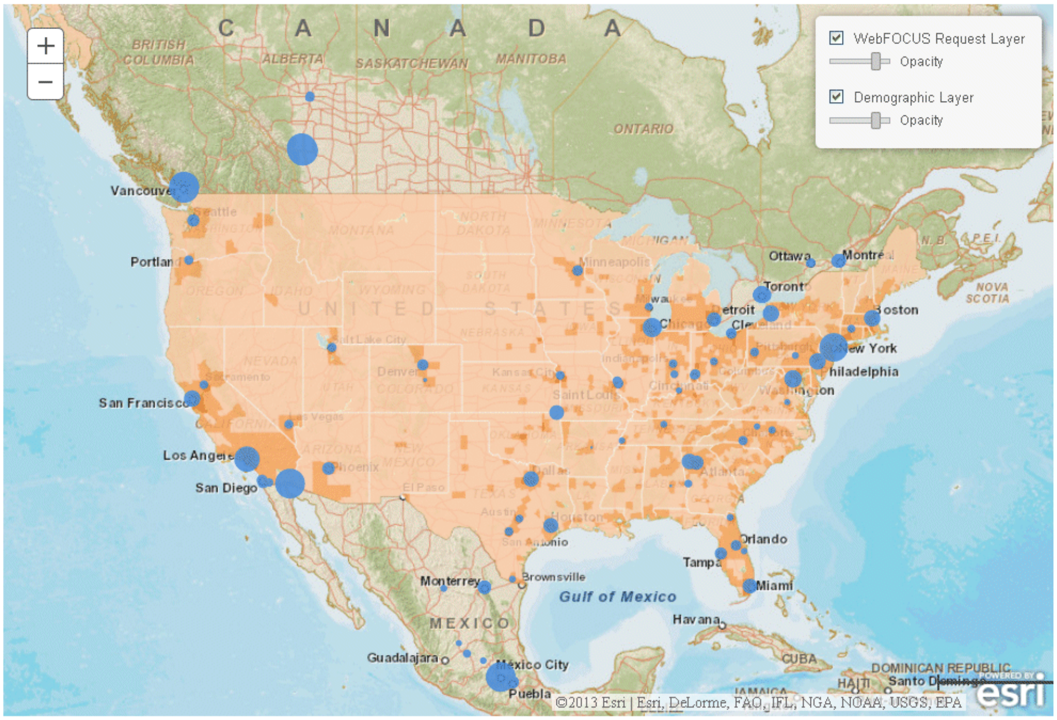
http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Recent_Population_Change/MapServer

Example: Adding a Demographic Overlay Layer to a Chart Using Chart Engine Syntax

The following request adds the USA Population Density demographic layer to a bubblemap. Since the demographic layer applies to the US, the base map is centered in the US and is zoomed in, as described in *Customizing Esri Map Output* on page 879.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker":{"maxSize":"5%"},
"extensions": {
 "com.esri.map": {
   "overlayLayers":             [
     {
     "title": "WebFOCUS Request Layer",
     "layerType": "bubble",
     "geometryXY": {
             "y": "lat",
             "x": "lng"
                 },
    "geometrySourceType": "seriesdata"
     },
  {
       "title": "Demographic Layer",
       "layerType": "tile",
       "url": "http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/
USA_Population_Density/MapServer"
      },
   ],


 "baseLayer": {
   "basemap": "streets",
  "center": [-97, 40],
  "zoom": 4
         }
  }
}
*END
ENDSTYLE
END
```

The output is shown in the following image. The table of contents control has check boxes and opacity sliders for displaying each overlay layer:



Syntax:

How to Style a Dynamic Map Service Layer or Reference Feature Layer Using Chart Engine Syntax

The styles and functions used to create a renderer for a dynamic map service layer or reference feature layer can be customized by supplying smartMapping properties for fields accessible in the layer.

This is different from a tiled map layer, such as a demographic layer, that has been saved (cached) and can only be superimposed on a base map, as is. Both dynamic map service layers and reference feature layers are classified in the WebFOCUS request as layerType:"featurelayer".

A dynamic map service layer displays data that has not been cached, but is generated dynamically by the map service as the user navigates the map.

The syntax for supplying the smart mapping parameters is:

```
{
   "title": "string",
   "layerType": "featurelayer",
   "layerObjectType": "string",
   "smartMapping" :
   [
   array of fields or smartmapping objects,
    ],
  "geometrySourceType": "esri",
  "url": "url_to_endpoint"
   }
```

where:

`"title": "string"`

Is a title for the layer.

`"layerObjectType": "string"`

Specifies the type of layer to be styled.

- For a dynamic map service layer:

  `"layerObjectType": "esri/layers/ArcGISDynamicMapServiceLayer"`

- For a reference feature layer, which is the default layerObjectType:

  `"layerObjectType": "esri/layers/FeatureLayer"`

  A reference feature layer URL ends with a number that points to a specific layer of a feature map service, as opposed to the entire map service.

`"smartMapping":`

Can be an array of:

- Strings that are field names for layers on a map server.
- Objects that will be passed to Esri to create a *Promise* as defined by smartmapping at:

  https://developers.arcgis.com/javascript/jsapi/esri.renderers.smartmapping-amd.html#createclassedcolorrenderer

The layers of the map server represent different levels of granularity for the same geography. For example, there may be state, county, and ZIP Code layers. You can specify a field or object for each layer. The layer fields or objects are comma-separated. To skip a layer, represent that layer with only a comma (,).

One of the parameters in the smartMapping object can be a renderer creation function name. If one is supplied, it is called in the smartMapping object. If not, the default renderer creation function (createClassedColorRenderer) is called. Each renderer creation function has properties that can be customized in the smart mapping array. These properties are defined at the smart mapping URL listed previously. If you specify smartMapping properties, they will be passed to Esri.

The layers and field names for each layer are defined at the map server endpoint.

Note that when you specify smartMapping properties, you cannot customize the layer symbol.

`"url": "`*`url_to_endpoint`*`"`

Is the URL to the map server endpoint.

WebFOCUS adds the layer type and basemap to the properties.

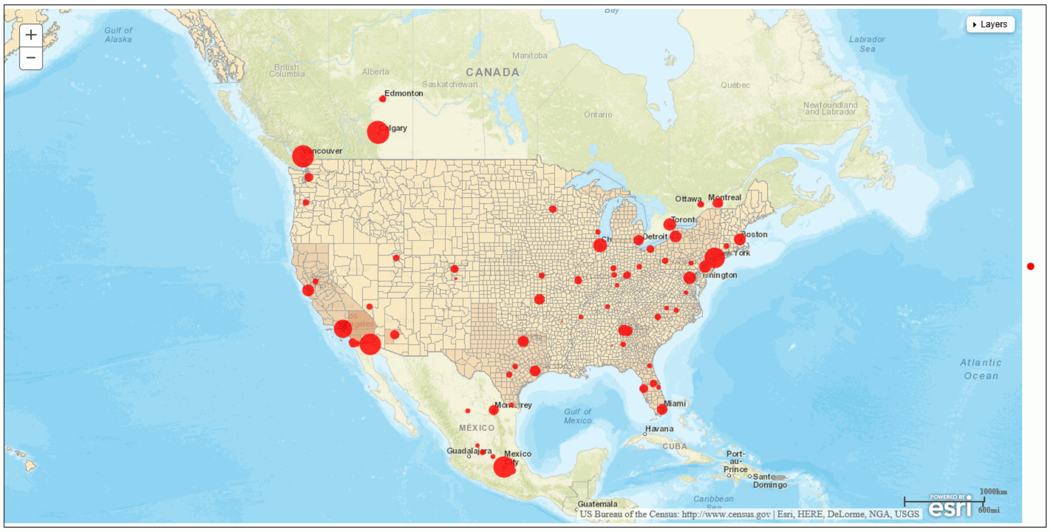
Example: Styling a Dynamic Map Service Layer Using Chart Engine Syntax

This example calls the smartMapping *createClassedColorRenderer* function (the default renderer creation function) for each layer that has an entry in the smartMapping array. There is no entry in the 0th array location (which represents census block points and is skipped using an initial comma), so nothing is done for that layer. For positions 1 (census block groups) and 2 (detailed counties), the *field* entry is set to "POP2007”. For position 1, the smartMapping classificationMethod property is set to "quantile" (this is the WebFOCUS default). For position 2, the classificationMethod property is set to "equal-interval". No object is specified for position 3, which represents states:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
```

```
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker": {"maxSize":"5%"},
"extensions": {
    "com.esri.map":
           {
             "baseLayer":{ "title" : "Base Layer",
             "basemap": "streets",
                       "center": [-98, 40],
                       "zoom": 4,
    },
             "overlayLayers":
[
                {
                    "title": "WebFOCUS Request Layer",
                    "layerType": "bubble",
                    "geometryXY": {
                        "y": "lat",
                        "x": "lng"},
                    "geometrySourceType": "seriesdata"
                },

              {
                    "title": "Feature Layer",
                    "layerType": "featurelayer",

                   "layerObjectType" : "esri/layers/ArcGISDynamicMapServiceLayer",
                    "smartMapping" :
                    [ ,
                     {"field":"POP2007","classificationMethod" : "quantile", },
                     {"field":"POP2007","classificationMethod" : "equal-
interval","numClasses":10}
                    ],

                     "geometrySourceType": "esri",
                     "url": "http://sampleserver6.arcgisonline.com/arcgis/rest/
services/Census/MapServer"
                }
 ]
              }
                  }

*END
ENDSTYLE
END
```

The output is shown in the following image:

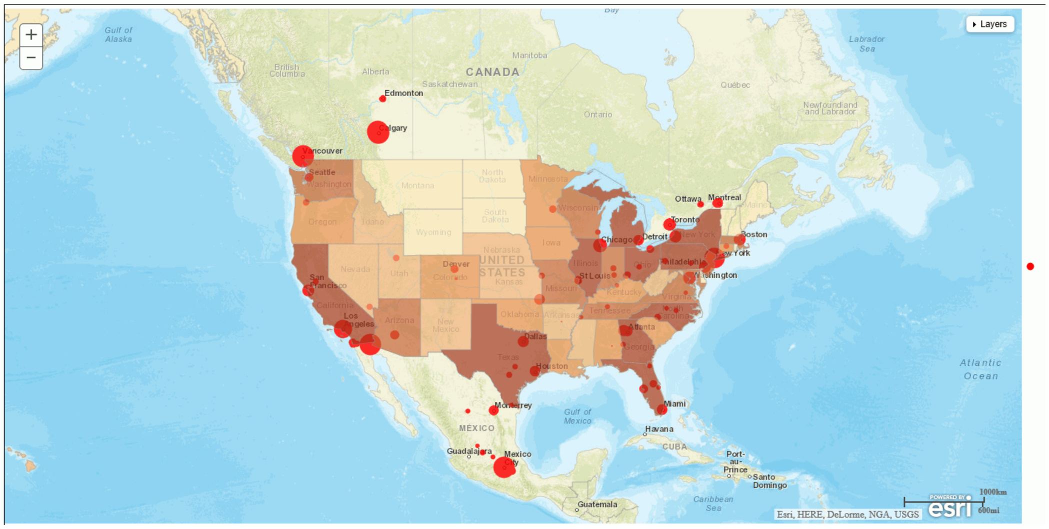


Example: Styling a Reference Feature Layer Using Chart Engine Syntax

In the following request, the endpoint is a reference feature layer that represents states. Therefore, the layerObjectType property can be omitted, as this is the default layer type. The smartMapping array specifies only a field name (POP2010), so default styles are used:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker": {"maxSize":"5%"},
"extensions": {
    "com.esri.map":
           {
       "baseLayer":{ "title" : "Base Layer",
               "basemap": "streets",
            "center": [-98, 40],
            "zoom": 4,
       },
   "overlayLayers":             [
    {
     "title": "WebFOCUS Request Layer",
     "layerType": "bubble",
     "geometryXY": {
      "y": "lat",
      "x": "lng"},
     "geometrySourceType": "seriesdata"
     },


     {
      "title": "Feature Layer",
      "layerType": "featurelayer",
"url":"http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/
USA_States_Generalized/FeatureServer/0",
     smartMapping : ["POP2010"],
     geometrySourceType : "esri"
    }
    ],
 }
}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Support for Delimited Files Using Chart Engine Syntax

When you generate an Esri map chart in HTML5 format, your request needs to point to a location that describes the geography for the request location values. You can point to a comma-separated value (.csv) file or other delimited file that contains the names of the regions you want to show on the map and their longitude and latitude values.

Syntax: How to Define Properties for a Delimited File

If you are using a delimited file for your geometry data, you can add a csvOptions object to the layer that will access the delimited file for its geometry data. However, there are defaults for all of the options. Therefore, if your delimited file conforms to the default options, you do not need to include the csvOptions object.

The default delimited file will not have a first row containing field names (titles). It will be comma-delimited with a \n line-end character at the end of each line. It will contain the following three values per line:

1. The name of the geometric location enclosed in single or double quotation marks. This field will be referred to as *region* in the map extensions properties.
2. The longitude that corresponds to the region, which is referred to as *lng* in the map extensions properties.
3. The latitude that corresponds to the region, which is referred to as *lat* in the map extensions properties.

If your file does not conform to these criteria, you can describe the structure of your file using the csvOptions object in your layer properties.

```
"extensions": {
  "com.esri.map": {
    "overlayLayers":
    [
     {
        "csvOptions":
         {"fieldNames": ["fld1", "fld2", "fld3"],
          "fieldSep": "string",
          "lineSep": "string",
          "firstRowIsTitles": boolean,
          "removeSurroundingQuotes": boolean}
       }
    ],
                   }
                     }
```

where:

`"fieldNames": ["fld1", "fld2","fld3"]`

Are the field names in your file. The default values are ["region", "lng", "lat"].

`"fieldSep": "string"`

Is the field separator character in your file. The default value is a comma (","). It can be any printable character.

`"lineSep": "string"`

Is the line separator string in your file. The default value is a line end character ("\n").

`"firstRowIsTitles": boolean`

Specifies whether the first row in your file contains titles. Valid values are:

- true, if your file has a first row containing titles. If it does, these names will take precedence over the field names listed in the fieldNames object. When you specify the value of geometryLocateField, use the title from the file.
- false, if your file does not have a first row containing titles. This is the default value. If you specified a fieldNames object, use the value of the appropriate field name as the geometryLocateField value.

`"removeSurroundingQuotes": boolean`

Specifies whether to remove quotation marks surrounding values in your file. Valid values are:

- true, to remove quotation marks surrounding values. This is the default value.

- ❑ false, if your file does not have quotation marks surrounding values.

The following are the default values for the csvOptions object:

```
"csvOptions":
     {
      "fieldNames": ["region", "lng", "lat"],
      "fieldSep": ",",
      "lineSep": "\n",
      "firstRowIsTitles": false,
      "removeSurroundingQuotes": true
     }
```

The following layer properties must then be set to values that correspond to your csvOptions:

```
"title": "string",
"layerType": "string",
"geometrySourceType": "csv",
"geometryLocateField": ["string"],
"url": "url_to_csv_file"
```

where:

`"title": "string"`

Is a title for the layer.

`"layerType": "string"`

Is the type of layer. Valid values are:

- ❑ **choropleth.** Displays a WebFOCUS choropleth layer.
- ❑ **bubble.** Displays a WebFOCUS bubblemap layer.

`"geometryLocateField": ["string"]`

Is the field name for the geometry locator field. If your file has titles, the value should come from the file. If not, it should be the appropriate field name from the fieldNames property of your csvOptions object. If neither of these is specified, use the default, which is "region".

`"url": "url_to_csv_file"`

Is the URL that points to your file.

Example: Using a .csv File to Generate a Bubblemap

The country.csv file contains longitude and latitude values for each country.

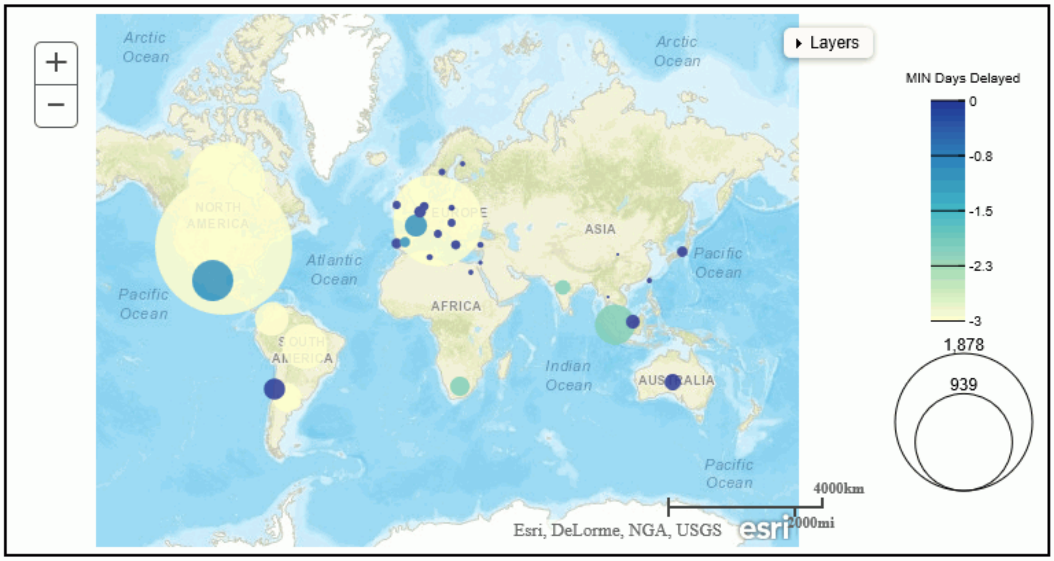
Following are the first few lines of the country.csv file:

```
"Argentina",-64.00000000,-34.00000000
"Australia",133.00000000,-27.00000000
"Austria",13.33330000,47.33330000
"Belgium",4.00000000,50.83330000
"Brazil",-55.00000000,-10.00000000
"Canada",-95.00000000,60.00000000
"Chile",-71.00000000,-30.00000000
"China",105.00000000,35.00000000
"Colombia",-72.00000000,4.00000000
```

The following request generates a bubblemap chart that shows days delayed by country using the country.csv file as the geometry data file. This file conforms to the default structure, so no csvOptions object is included in the layer properties:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
MIN.DAYSDELAYED
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=country_name, BUCKET=location, $
TYPE=DATA, COLUMN=daysdelayed, BUCKET=size, $
TYPE=DATA, COLUMN=min.daysdelayed, BUCKET=color, $
*GRAPH_JS
"extensions": {
  "com.esri.map": {
    "baseLayer": {
      "title": "base title",
      "basemap": "streets"
                 },
    "overlayLayers": [
      {
         "title": "csv default",
         "layerType": "bubble",
        "geometrySourceType": "csv",
        "geometryLocateField": ["region"],
         "url": "http://ecl.informationbuilders.com/jschart/country.csv"
      }
                        ],
                     }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image:



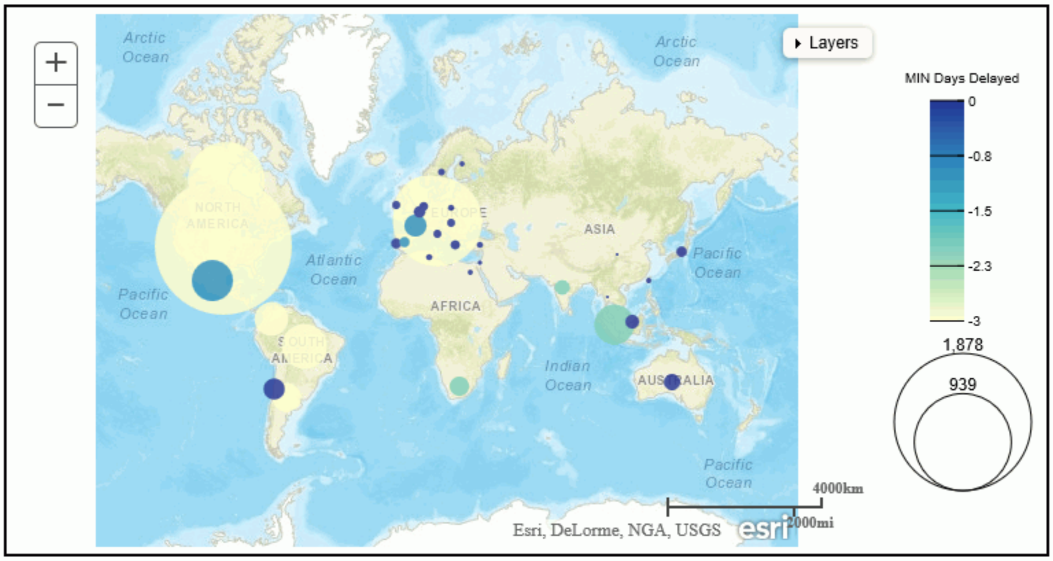
The following file (named countryp.txt) has a row with titles as the first row, the separator is a pipe character (|), and the values are enclosed in double quotation marks. The first several lines in the file are shown below:

```
"Country"|"lng"|"lat"
"Argentina"|"-64.0000000"|"-34.0000000"
"Australia"|"133.0000000"|"-27.0000000"
"Austria"|"13.3333000"|"47.3333000"
"Belgium"|"4.0000000"|"50.8333000"
"Brazil"|"-55.0000000"|"-10.0000000"
"Canada"|"-95.0000000"|"60.0000000"
```

The following version of the request uses this file. It has a csvOptions object to describe the non-default options. Since the file has titles, no fieldNames object is included. The title row in the file specifies “Country” as the geometry locator field:

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
MIN.DAYSDELAYED
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=country_name, BUCKET=location, $
TYPE=DATA, COLUMN=daysdelayed, BUCKET=size, $
TYPE=DATA, COLUMN=min.daysdelayed, BUCKET=color, $
*GRAPH_JS
"extensions": {
  "com.esri.map": {
    "baseLayer": {
      "title": "base title",
      "basemap": "streets"
                      },
    "overlayLayers": [
      {
        "title": "csv options",

        "csvOptions":
         {
            "fieldSep": "|",
            "firstRowIsTitles": true,
            "removeSurroundingQuotes": true
         } ,
        "layerType": "bubble",
        "geometrySourceType": "csv",
        "geometryLocateField": ["Country"],
         "url": "http://ecl.informationbuilders.com/jschart/countryp.txt"
       }
                               ],
                                  }
                           }
*END
ENDSTYLE
END
```

The output is shown in the following image:



Support for Geojson Files Using Chart Engine Syntax

When you generate an Esri map chart in HTML5 format, your request needs to point to a location that describes the geography for the request location dimension. You can point to a geojson file that contains the names of the locations you want to show on the map and their longitude and latitude values.

Syntax: How to Define Properties for a Geojson File

Add the following properties to the overlay layer that will use the geojson file.

```
"overlayLayers": [
 {
 "title": "string",
 "layerType": "choropleth",
 "geometrySourceType": "geojson",
 "geometryLocateField": ["string"],
 "geometryDataField": "name",
 "url": "http://url_to_geojson_file"
 }
 ],
```

where:

`"title": "string"`

Is a title to display in the layer table of contents.

`"geometryLocateField": ["string"]`

Is the name of the location field in the geojson file.

`"url": "http://url_to_geojson_file"`

Is the URL that points to the location of the geojson file.

Example: Using a Geojson File With an HTML5 Map Chart

The sample_wfretail_regions.geojson file contains latitude and longitude values for the borders of United States regions. The following shows a partial listing of the beginning of the file.

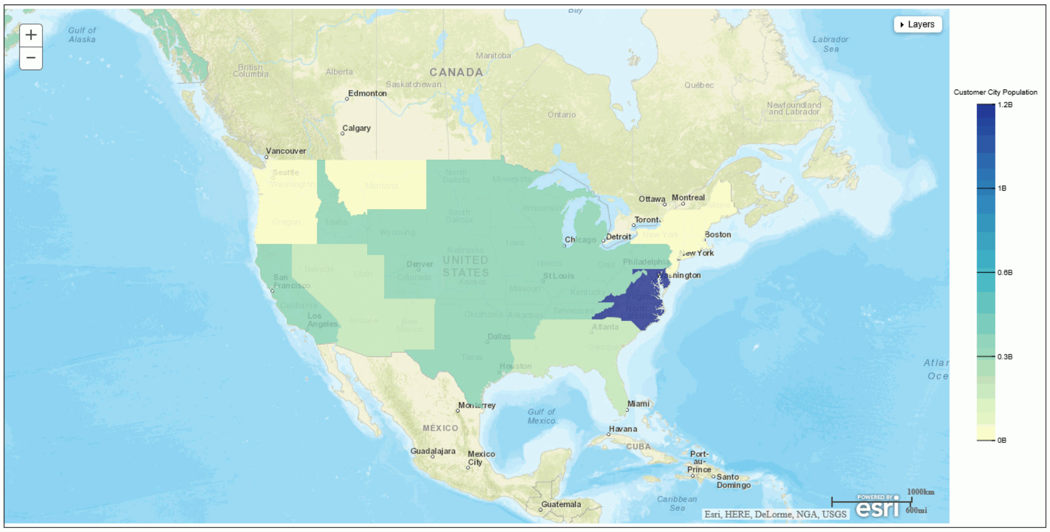
```
{
"type": "FeatureCollection",
"features":[
{ "type": "Feature","properties": { "region": "Central" },
"geometry": { "type": "Polygon","coordinates":
[[[-84.7570,45.7847],[-84.7288,45.7880],[-84.7212,45.7666],
[-84.5570,45.7084],[-84.4780,45.6566],[-84.4295,45.6448],
[-84.4150,45.6530],[-84.4285,45.6697],[-84.3768,45.6558],
[-84.3293,45.6643],[-84.2165,45.6349],[-84.1326,45.5666],
[-84.1123,45.5077],[-84.0749,45.4888],[-83.9362,45.4925],
[-83.8103,45.4204],[-83.7414,45.4037],[-83.7157,45.4144],
[-83.5988,45.3523],[-83.5374,45.3578],[-83.4953,45.3396],
[-83.4912,45.3590],[-83.4780,45.3453],[-83.4860,45.3308],
[-83.4636,45.3307],[-83.3830,45.2718],[-83.3913,45.2527],
[-83.4164,45.2566],[-83.4125,45.2402],[-83.3624,45.1652],
[-83.3165,45.1418],[-83.3199,45.1178],[-83.3008,45.0922],
[-83.3263,45.0850],[-83.3130,45.0849],[-83.3140,45.0546],
[-83.2616,45.0253],[-83.3390,45.0412],[-83.3779,45.0776],
[-83.4364,45.0558],[-83.4564,45.0230],[-83.4332,45.0045],
[-83.4647,45.0028],[-83.4280,44.9282],[-83.3153,44.8802],
[-83.3218,44.8553],[-83.2938,44.8083],[-83.2959,44.7413],
[-83.2717,44.7132],[-83.3145,44.6087],[-83.3072,44.5435],
[-83.3334,44.3372],[-83.3749,44.3267],[-83.4498,44.2507],
[-83.4431,44.2715],[-83.4788,44.2800],[-83.5357,44.2507],
[-83.5659,44.1634],
 ...
}
```

The name of the field that contains the region names is *region*. The following request assigns states to the region names defined in the geojson file and uses this file as the geometry file for the map layer.

```
DEFINE FILE WF_RETAIL_LITE
STATE/A20=WF_RETAIL_LITE.WF_RETAIL_GEOGRAPHY_CUSTOMER.STATE_PROV_NAME;
REGION/A20 WITH WF_RETAIL_LITE.WF_RETAIL_GEOGRAPHY_CUSTOMER.STATE_PROV_NAME=
IF STATE EQ 'Illinois' OR 'Indiana' OR 'Iowa'
   OR 'Kentucky' OR 'Michigan' OR 'Minnesota' OR 'Missouri'
   OR 'Ohio' OR 'Wisconsin' OR 'Kansas' OR 'Nebraska' THEN 'Central'
ELSE IF STATE EQ 'Alabama' OR 'Florida' OR  'Georgia' OR 'Louisiana'
   OR 'Mississippi' OR 'North Carolina' OR 'South Carolina'
   OR 'Virginia' THEN 'SouthEast'
ELSE IF STATE EQ 'Arkansas' OR 'Tennessee' OR 'West Virginia'
   OR 'Texas' OR 'Oklahoma'
   OR 'New Mexico' OR 'Hawaii' THEN 'SouthWest'
ELSE IF STATE EQ 'Connecticut' OR 'Maine' OR 'Massachusetts'
   OR 'New Hampshire' OR 'Rhode Island' OR 'Vermont' THEN 'NorthEast'
ELSE IF STATE EQ 'New Jersey' OR 'New York' OR 'Pennsylvania'
   OR 'Delaware' OR 'Maryland' THEN 'MidEast'
ELSE IF STATE EQ 'Colorado' OR 'Idaho' OR 'Montana' OR 'North Dakota'
   OR 'South Dakota' OR 'Wyoming' OR 'Alaska' THEN 'NorthWest'
ELSE IF STATE EQ 'Arizona' OR 'California' OR 'Nevada' OR 'Oregon'
   OR 'Utah' OR 'Washington' THEN 'West'
ELSE 'Unknown';
END
```

```
GRAPH FILE WF_RETAIL_LITE
SUM CITY_POPULATION
BY REGION
WHERE COUNTRY_NAME EQ 'United States'
WHERE REGION NE 'Unknown'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=region, BUCKET=location, $
TYPE=DATA, COLUMN=city_population, BUCKET=color, $
*GRAPH_JS
"extensions": {
 "com.esri.map": {
 "overlayLayers": [
 {
 "title": "Geojson",
 "layerType": "choropleth",
 "geometrySourceType": "geojson",
 "geometryLocateField": ["region"],
 "geometryDataField": "name",
 "url": "http://ecl.informationbuilders.com/jschart/sample_wfretail_regions.geojson"
 },
 ],
 "baseLayer": {
 "basemap": "streets",
"center": [-98, 40],
"zoom": 4                  }
 }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Customizing Esri Map Output

You can specify Esri properties in your chart request to customize the initial display of the map and the controls available on the map output.

Syntax: How to Format a Reference Feature Layer Symbol

A reference feature layer places symbols on the map chart. WebFOCUS uses these layers to outline locations. The symbol used by WebFOCUS is comprised of a simple fill symbol and a simple line symbol. If you have access to point layers, you may want to use a simple marker symbol. You can change the properties of the symbol by adding a symbol object to the feature layer.

For example, the default symbol used by WebFOCUS to outline locations consists of a simple line symbol for the outline that is black and opaque, and a simple fill symbol defined to have no fill color or style. You can edit the symbol definition to add a fill style and color and to change the color, style, and width of the outline.

Simple fill symbols are defined at:

https://developers.arcgis.com/javascript/jsapi/simplefillsymbol-amd.html

Simple line symbols are defined at:

https://developers.arcgis.com/javascript/jsapi/simplelinesymbol-amd.html

Simple marker symbols are defined at:

https://developers.arcgis.com/javascript/jsapi/simplemarkersymbol-amd.html

WebFOCUS symbols, by default, consist of the following symbol properties:

```
"symbol": {
          "type": "esriSFS",
            "style": "esriSFSstring",
            "color": [r, g, b, t],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSstring",
            "color": [r, g, b, t],
            "width": number                      }
        },
```

where:

`"type": "esriSFS"`

Specifies that the fill uses a simple fill symbol. You can find the definition and options for simple fill symbols at:

https://developers.arcgis.com/javascript/jsapi/simplefillsymbol-amd.html

`" style": "esriSFSstring"`

Defines the fill style. The WebFOCUS default is no fill, "esriSFSNull".

`"color": [r, g, b, t]`

Defines the fill color in RGBA format. The color is defined by specifying the intensity of the red, green, and blue components, and a transparency value. Each component can have a value from 0 to 255. For the transparency value, zero (0) means totally transparent, and 255 means totally opaque. The WebFOCUS default is [0,0,0,0].

`"type": "esriSLS"`

Specifies that the outline uses a simple line symbol. You can find the definition and options for simple line symbols at:

https://developers.arcgis.com/javascript/jsapi/simplelinesymbol-amd.html

`"style": "esriSLSstring"`

Defines the line style. The WebFOCUS default is a solid line, "esriSLSSolid".

`"color": [r, g, b, t]`

Defines the line color in RGBA format. The color is defined by specifying the intensity of the red, green, and blue components, and a transparency value. Each component can have a value from 0 to 255. For the transparency value, zero (0) means totally transparent, and 255 means totally opaque. The WebFOCUS default is [0,0,0,255].

"width": *number*

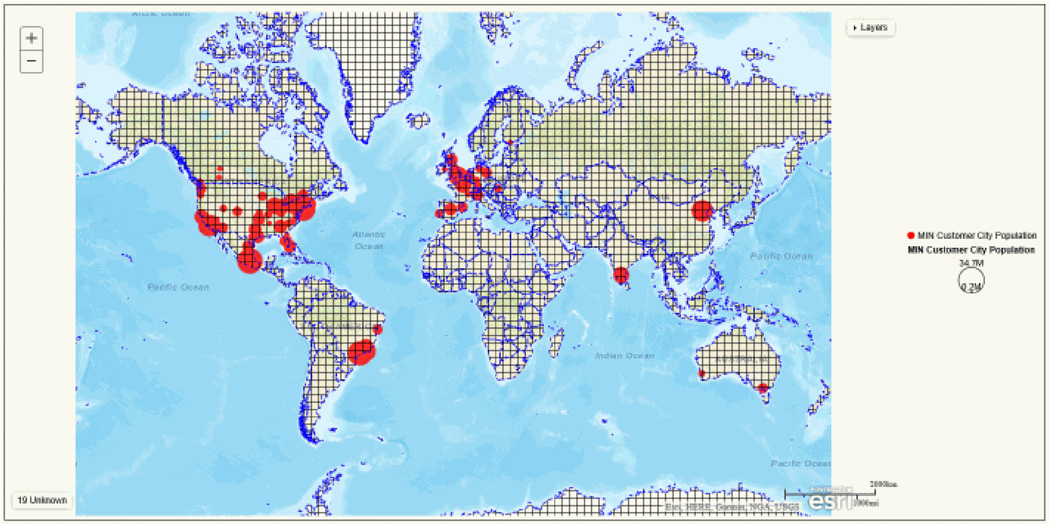
Is the line width in pixels. The WebFOCUS default is 1.

Example: Formatting a Reference Feature Layer Symbol Using Reporting Server Syntax

The following request changes the symbol fill style to cross-hatch and the outline to a blue dashed line:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY CITY_NAME
WHERE CITY_POPULATION NE  ' ' OR MISSING
WHERE COUNTRY_NAME NE  'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS_FINAL
"bubbleMarker": {"maxSize":"5%", "color": "red"},

"extensions": {
  "com.esri.map": {
      "overlayLayers":
     [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "CITY_NAME"
                                             }
                                    }
         },
       {"ibiAddLayer": "World_Countries",
           "symbol": {
           "type": "esriSFS",
             "style": "esriSFSCross",
              "color": [0, 0, 0, 255],
                  "outline": {
                  "type": "esriSLS",
                 "style": "esriSLSDash",
                        "color": [0,0,255, 255],
                 "width": 1
                           }
                      }
       }
      ],
    "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "streets"
                }
            ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image:

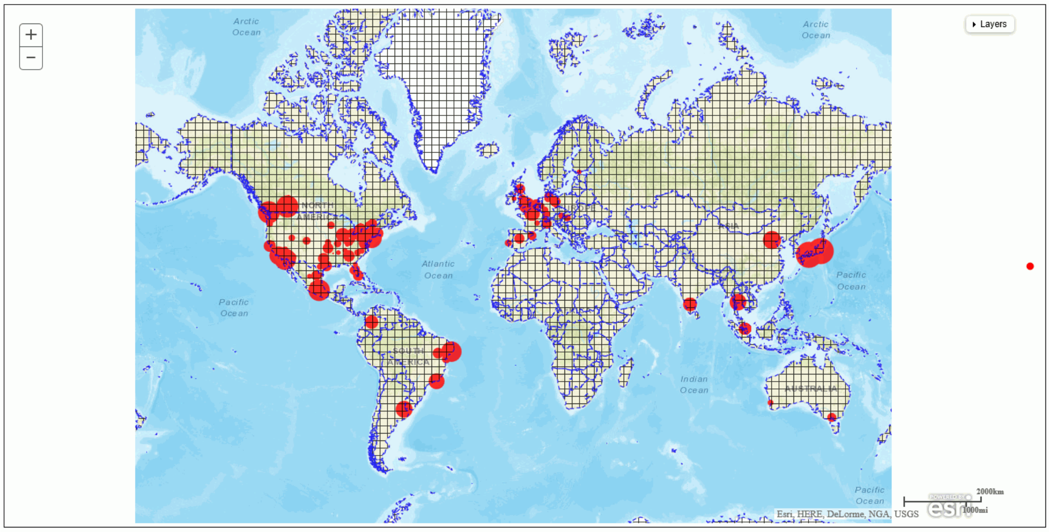

Layers
Atlantic Ocean
Pacific Ocean
Pacific Ocean
Indian Ocean
MIN Customer City Population
MIN Customer City Population
34.7M
0.2M
19 Unknown
Esri, HERE, Garmin, NGA, USGS

Example: Formatting a Reference Feature Layer Symbol Using Chart Engine Syntax

The following request changes the symbol fill style to cross-hatch and the outline to a blue dashed line:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker": {"maxSize":"5%"},
"extensions": {
    "com.esri.map":
           {
             "baseLayer":{ "title": "Base Layer",
         "basemap": "streets"
    },
             "overlayLayers":             [
                {
                    "title": "WebFOCUS Request Layer",
                    "layerType": "bubble",
                    "geometryXY": {
                        "y": "lat",
                        "x": "lng"},
                    "geometrySourceType": "seriesdata"
                },
              {
                    "title": "Country Feature Layer",
                    "layerType": "featurelayer",
                        "symbol": {
                             "type": "esriSFS",
                             "style": "esriSFSCross",
                             "color": [0, 0, 0, 255],
                             "outline": {
                                "type": "esriSLS",
                                "style": "esriSLSDash",
                                "color": [0,0,255, 255],
                                "width": 1
                                         }
                                     },
                    "geometrySourceType": "esri",
                    "geometryLocateField": ["Country"],
                    "geometryDataField": "name",
                    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/
services/World_Countries_(Generalized)/FeatureServer/0"
                }

            ],

             }
                  }

*END
ENDSTYLE
END
```

The output is shown in the following image:



Syntax:

How to Add Layer Options

To add Esri layer options, add the *options* object to the layer definition. WebFOCUS will not validate any options you add, but will pass them directly to Esri:

```
"options": {"layer_option":value, ...}
```

where:

layer_option

Is a layer option available in the Esri API. These are passed to Esri for processing. The Esri API is described at:

https://developers.arcgis.com/javascript/jsapi/map-amd.html#map1

value

Is a valid value for the layer option.

Example:

Defining the Initial Layer Opacity Using Reporting Server Syntax

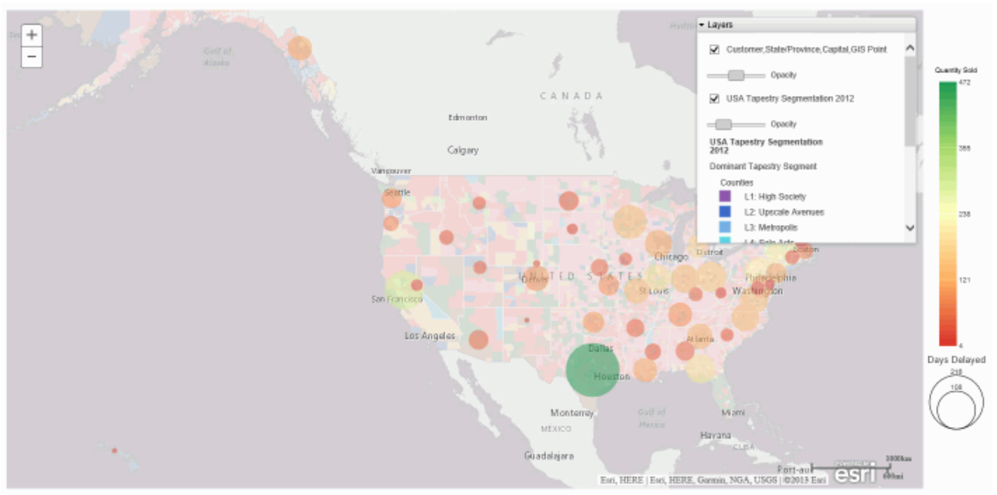
The following request adds initial opacity values for each overlay layer. Opacity is a number from 0.0 to 1 that represents the percentage of layer opacity, where 0 means 100% transparent and 1 means 100% opaque.

Note: For values less than 1, the leading zero and decimal point (0.) are required.

In the following request, the proportional symbols are 50% opaque and the demographic layer is 20% opaque:

```
GRAPH FILE wfretail82/WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY STATE_PROV_CAPITAL_POINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                     "map_by_field": "STATE_PROV_CAPITAL_POINT"
                                            }
                                 },
                  "options": {"opacity": 0.5}
         },
         {"ibiAddLayer": "USA_Tapestry_Segmentation_2012",
           "options": {"opacity": 0.2}
        }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
             ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image. Note that the opacity sliders in the table of contents control reflect the opacity values in the request.



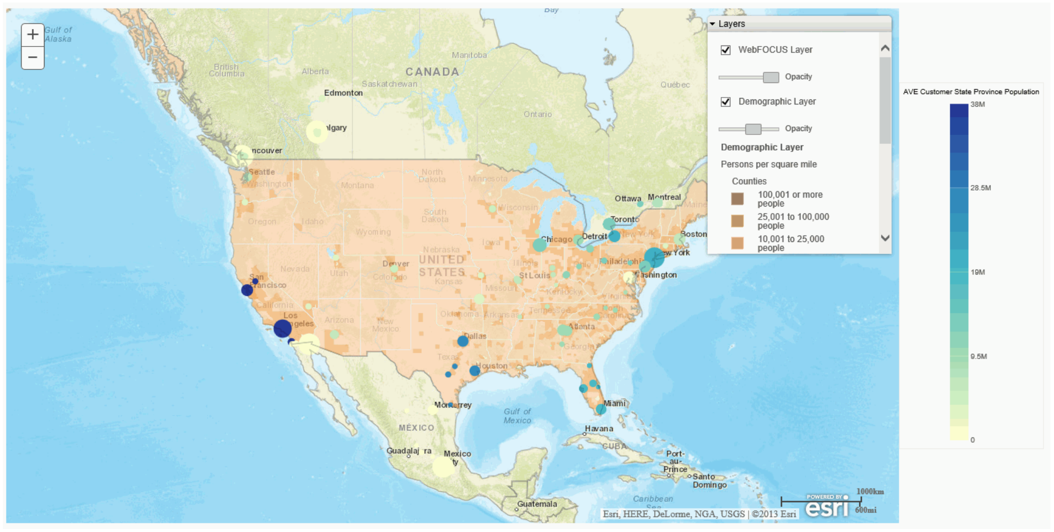
Example: Defining the Initial Layer Opacity Using Chart Engine Syntax

The following request adds initial opacity values for each overlay layer. Opacity is a number from 0 to 1 that represents the percentage of layer opacity, where 0 means 100% transparent and 1 means 100% opaque.

In the following request, the proportional symbols are 100% opaque, the demographic layer is 60% opaque, and the feature layer is 20% opaque:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
AVE.STATE_PROV_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=AVE.STATE_PROV_POPULATION, BUCKET=color, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker": {"maxSize":"5%"},
extensions:
{
 "com.esri.map":
  {
   "baseLayer":
  {"title" : "Base Layer",
   "basemap": "streets",
   "center": [-98, 40],
   "zoom": 4
   },
   "overlayLayers":
   [
   {
    "title": "WebFOCUS Layer",
    "layerType": "bubble",
    "geometryXY": {
         "y": "lat",
         "x": "lng"},
    "geometrySourceType": "seriesdata",
    "options": {"opacity":1}
    },
    {
    "title": "Demographic Layer",
    "layerType": "tile",
    "url": "http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/
USA_Population_Density/MapServer",
    "options": {"opacity":.6}
    },
    {
    "title": "Country Feature Layer",
    "layerType": "featurelayer",
    "geometrySourceType": "esri",
    "geometryLocateField": ["Country"],
    "geometryDataField": "name",
    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0",
    "options": {"opacity":.2}
    }
    ]
  }
 }
*END
ENDSTYLE
END
```

The output is shown in the following image. Note that the opacity sliders in the table of contents control reflect the opacity values in the request.



Syntax: How to Set the Initial Center Point and Zoom Properties for an Esri Map Chart

Useful properties that you can add to the base layer definition are an initial center point, maximum zoom level, minimum zoom level, and initial zoom level. If you do not provide these, WebFOCUS calculates their values based on the geometry in the request output:

```
"baseLayer": {title: "layertitle",
             "basemap": "string",
             "center": [longitude,latitude],
             "minZoom": number,
             "maxZoom": number,
             "zoom": number       }
```

where:

`"title": "layertitle"`

Is an optional title for the layer.

`"basemap": "string"`

Is the name of one of the supported base maps.

Note: With Reporting Server syntax, the basemap name is taken from the baseMapInfo object. If you specify a basemap name in the baseLayer object, it will be ignored.

`"center": [`*`longitude`*`,`*`latitude`*`]`

Are the longitude and latitude for the center of the base map when initially displayed.

`"minZoom":` *`number`*

Is the minimum zoom level for the chart.

`"maxZoom":` *`number`*

Is the maximum zoom level for the chart.

`"zoom":` *`number`*

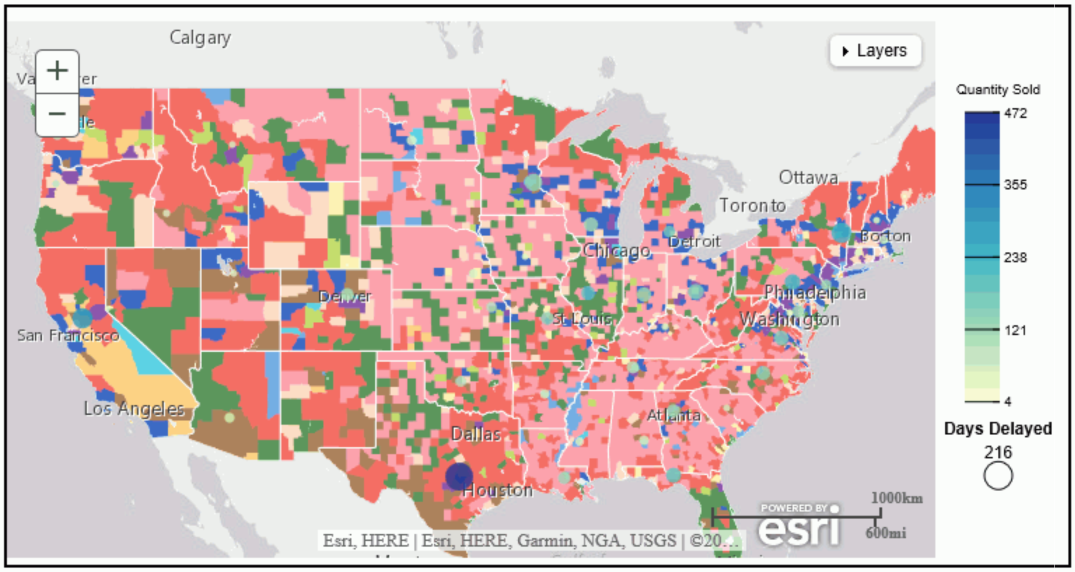
Is an integer from 0 (zero) to 19 specifying the zoom level for the chart when initially displayed.

Example: Adding a Center Point and Initial Zoom Level Using Reporting Server Syntax

The following request centers the base map in the US and sets the initial zoom to 4.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY STATE_PROV_CAPITAL_POINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"bubbleMarker": {"maxSize": "5%"},
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                     "map_by_field": "STATE_PROV_CAPITAL_POINT"
                                            }
                                   }
          },
         {"ibiAddLayer": "USA_Tapestry_Segmentation_2012"
            }
      ],
   "baseLayer": {
       "center": [-97, 40],
       "zoom": 4
                }, 
       "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
            ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.

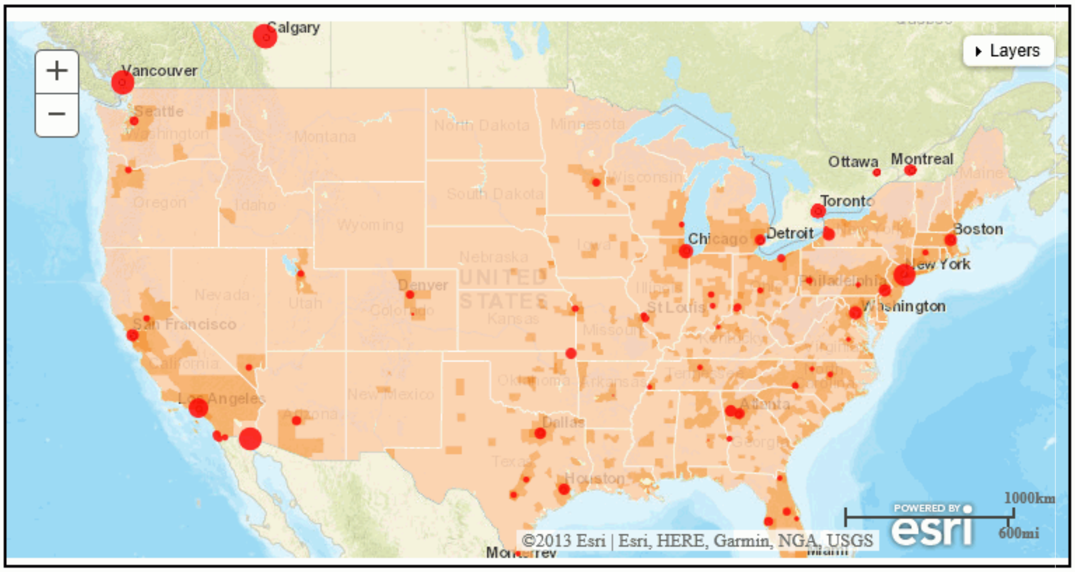


Example: Adding a Center Point and Initial Zoom Level Using Chart Engine Syntax

The following request centers the base map in the US and sets the initial zoom to 4.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker":{"maxSize":"5%"},
"extensions": {
"com.esri.map": {
"overlayLayers": [
{
"title": "WebFOCUS Request Layer",
"layerType": "bubble",
"geometryXY": {
"y": "lat",
"x": "lng"
              },
"geometrySourceType": "seriesdata"
},
{
"title": "Demographic Layer",
"layerType": "tile",
"url": "http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/
USA_Population_Density/MapServer"
}
],
"baseLayer": {
  "basemap": "streets",
  "center": [-97, 40],
  "zoom": 4
             }
}
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Syntax: **How to Add Basemap Information to an Esri Map Chart**

By adding a baseMapInfo object to an Esri chart request using chart engine syntax, you can specify an array of custom map images to use as the background for your chart. With Reporting Server syntax, you can add custom basemaps to the geography configuration file, as described in *The Reporting Server Geographic Configuration File* on page 836.

With both chart engine syntax and Reporting Server syntax, you can place a basemap control on the chart output that displays thumbnails of all available basemaps and lets you switch between basemaps on the resulting chart output. You can also change the title of the basemap thumbnail on the basemap control.

```
"baseMapInfo": {
 "drawBasemapControl": boolean,
 "showArcGISBasemaps": boolean,
 "customBaseMaps": [
 {
 "name": "string",
 "title": "string",
 "url": "url_to_basemap",
 "thumbnailUrl": "url_to_thumbnail"
 }
 ]
}
```

where:

`"drawBasemapControl": `*`boolean`*

Specifies whether to draw a basemap control on the chart output. Valid values are:

- true, which draws a basemap control.
- false, which does not draw a basemap control. This is the default value.

`"showArcGISBasemaps": `*`boolean`*

Specifies whether to show the ArcGIS Online basemaps on the basemap control. Valid values are:

- true, which shows the ArcGIS Online basemaps on the basemap control. This is the default value.
- false, which does not show the ArcGIS Online basemaps on the basemap control.

`"name": "`*`string`*`"`

With Reporting Server syntax, this property is ignored. With chart engine syntax, this is a name for the basemap that you can use as the baselayer:basemap property to select this basemap for the chart.

`"title": "`*`string`*`"`

Is a title to display under the thumbnail for this basemap on the basemap control.

`"url": "`*`url_to_basemap`*`"`

Applies to chart engine syntax only. Is the URL to the endpoint for the basemap.

`"thumbnailUrl": "`*`url_to_thumbnail`*`"`

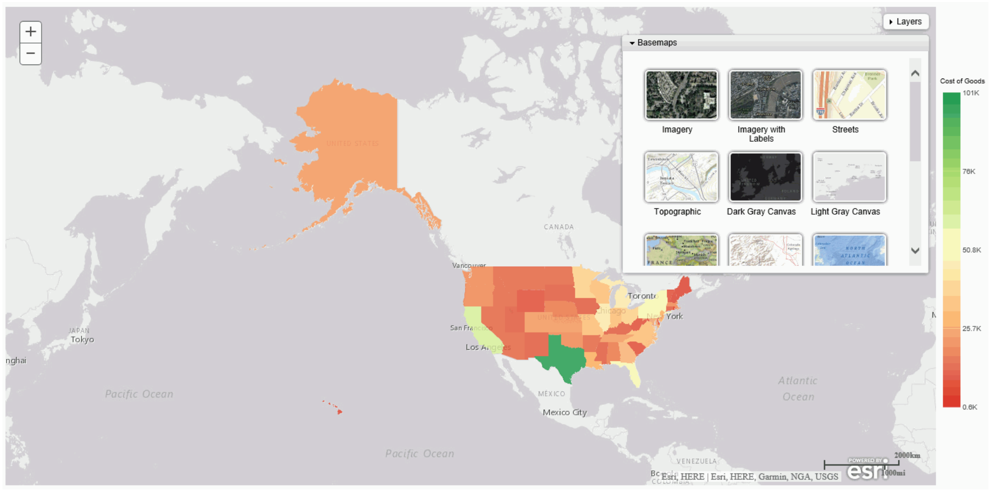
Applies to chart engine syntax only. Is the URL to the thumbnail for the basemap.

Example: Adding a Basemap Control to an Esri Map Chart Using Reporting Server Syntax

The following request adds a basemap control to the chart.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-metadata": {"map_by_field": "STATE_PROV_NAME"}
                               }
           }
         ],
      "baseMapInfo":  {
          "drawBasemapControl" : true,
           "showArcGISBasemaps" : true,
           "customBaseMaps":
           [
               {"ibiBaseLayer": "gray" }
            ]
                              }
                         }
                  }
*END
ENDSTYLE
END
```

The output, with the basemap control open, is shown in the following image.



To change the chart background, click a thumbnail on the basemap control.

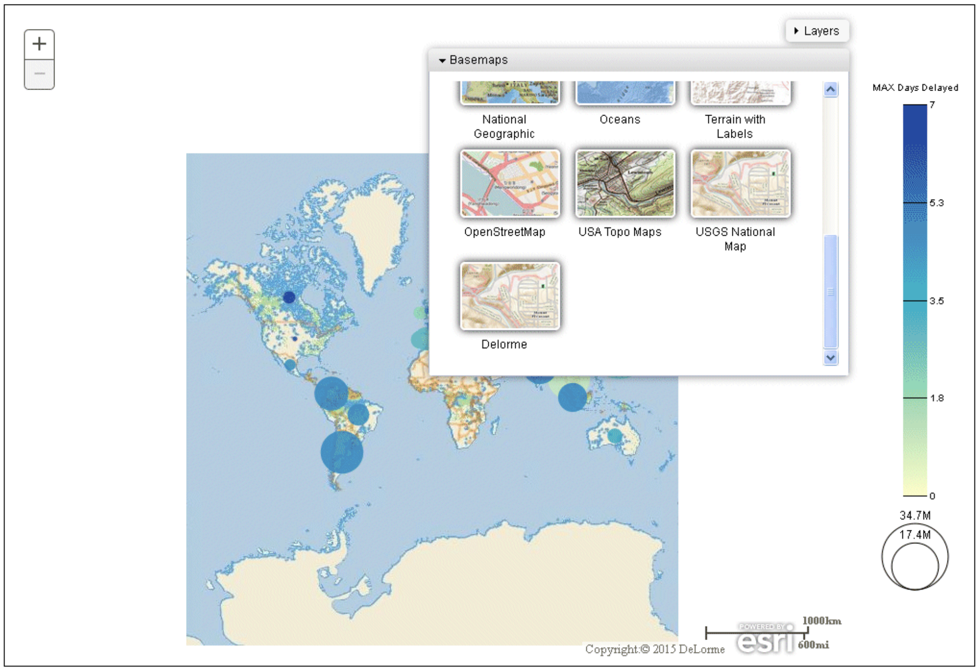
To close the basemap control, click anywhere in the header.

Example: Adding Basemap Properties to an Esri Map Chart Using Chart Engine Syntax

The following request adds a basemap from DeLorme and a basemap control to the chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION MAX.DAYSDELAYED
BY COUNTRY_NAME
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
TYPE=DATA, COLUMN=MAX.DAYSDELAYED, BUCKET=color, $
*GRAPH_JS
"bubbleMarker": {"maxSize": "10%" },
"extensions": {
"com.esri.map": {
"overlayLayers": [
{
"title": "Population",
"layerType": "bubble",
"geometrySourceType": "esri",
"geometryLocateField": ["Country"],
"geometryDataField": "name",
"url":
"http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
}
],
"baseMapInfo" : {
"drawBasemapControl" : true,
"showArcGISBasemaps" : true,
"customBaseMaps" : [
{
"name":"Delorme",
"title":"Delorme",
"url":"http://services.arcgisonline.com/ArcGIS/rest/services/Specialty/
DeLorme_World_Base_Map/MapServer",
"thumbnailUrl":"http://www.arcgis.com/sharing/rest/content/items/
6d9fa6d159ae4a1f80b9e296ed300767/info/thumbnail/national_map.jpg"
}
                       ]
                        },

"baseLayer": {
"basemap": "Delorme"
}
}
}
*END
ENDSTYLE
END
```

The output, with the basemap control open, is shown in the following image.



To change the chart background, click a thumbnail on the basemap control.

To close the basemap control, click anywhere in the header.

Syntax: How to Customize the Scale Bar on an Esri Map Chart

By default, a scale bar with dual units is placed on the lower right corner of the map. You can add properties to place the scale bar in another location and display either metric or English units.

The following code shows how to place a scale bar on the chart and define the type of units shown.

```
"extensions": {
    "com.esri.map":{
      "scalebar": {"scalebarUnit": "string",
                 "attachTo": "string" },
                   }
          }
```

where:

`"scalebarUnit": "string"`

Specifies the units to display. Valid values are:

- ❑ **english.** Shows English units such as miles.
- ❑ **metric.** Shows metric units such as kilometers.
- ❑ **dual.** Shows both types of units.

`"attachTo": "string"`

Specifies where to place the scale bar on the chart. Valid values are:

- ❑ **top-right**
- ❑ **bottom-right**
- ❑ **top-center**
- ❑ **bottom-center**
- ❑ **bottom-left**
- ❑ **top-left**

For other scale bar properties, see the scale bar API definition at:

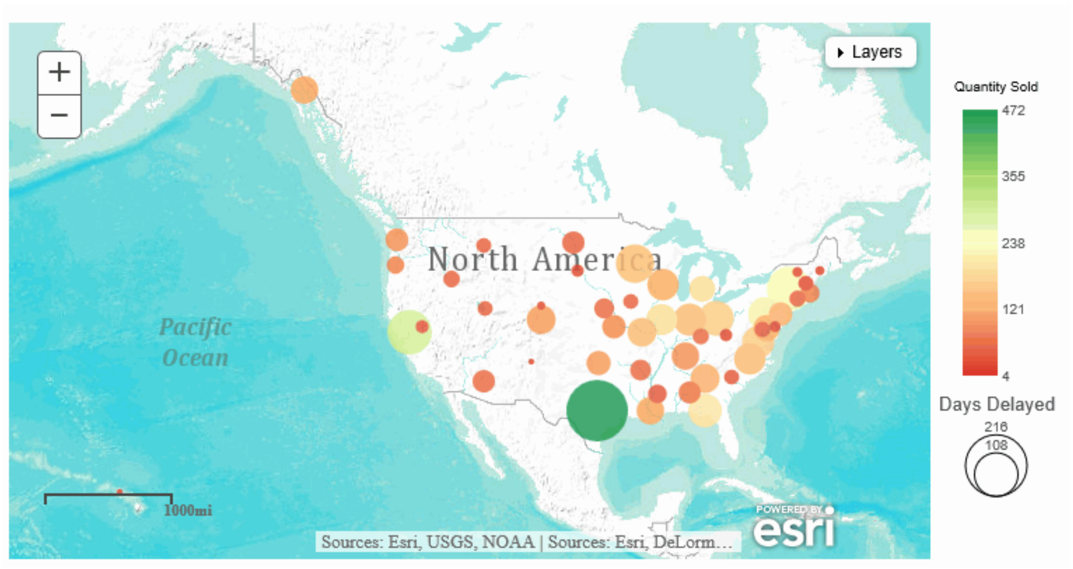
https://developers.arcgis.com/javascript/jsapi/scalebar-amd.html

Example: Customizing the Scale Bar on an Esri Map Chart Using Reporting Server Syntax

The following request moves the scale bar to the lower left corner of the map and displays only English units.

```
DEFINE FILE WF_RETAIL_LITE
GEOPOINT/A200 = GIS_POINT('4326', STATE_PROV_CAPITAL_LONGITUDE,
STATE_PROV_CAPITAL_LATITUDE);
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY GEOPOINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
   "scalebar" : {"scalebarUnit": "english",
    "attachTo" : "bottom-left"
                 },

       "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "GEOPOINT"
                                }
                             }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "terrain"
                }
            ]
          }
                         }
                  }
*END
ENDSTYLE
END
```

The output is shown in the following image.

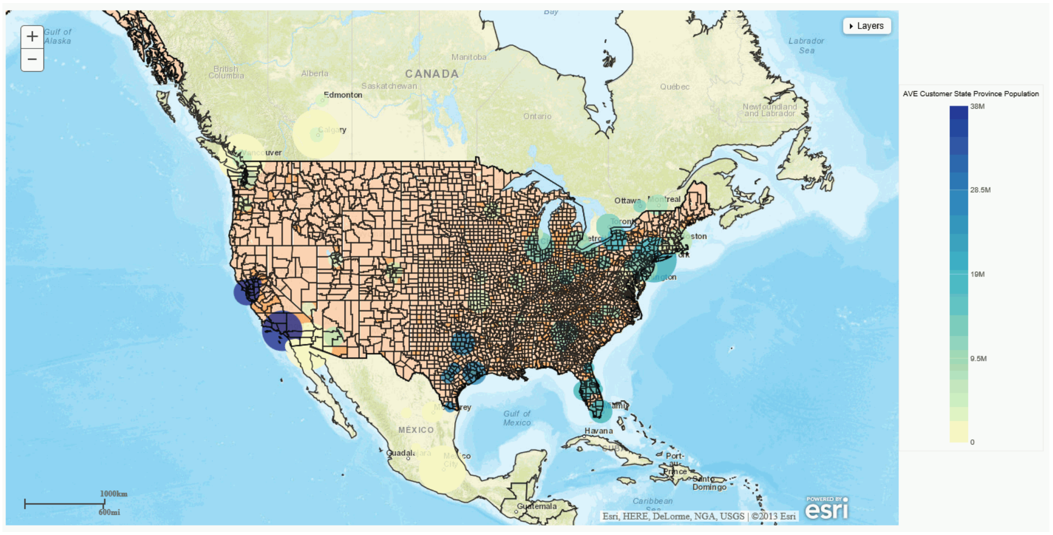


Example: Customizing the Scale Bar on an Esri Map Chart Using Chart Engine Syntax

The following request places a scale bar showing dual units on the bottom-left of the chart. It also defines one demographic and two feature layers:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
AVE.STATE_PROV_POPULATION
BY  CITY_LATITUDE
BY CITY_LONGITUDE
WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=AVE.STATE_PROV_POPULATION, BUCKET=color, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"extensions": {
  "com.esri.map":
  {
   "scalebar" : {"scalebarUnit": "dual",
                "attachTo" : "bottom-left"
                        },
  "baseLayer":{ "title" : "Base Layer",
   "basemap": "streets",
   "center": [-98, 40],
   "zoom": 4
         },
  "overlayLayers":
  [
    {
     "title": "WebFOCUS Layer",
     "layerType": "bubble",
     "geometryXY": {
           "y": "lat",
           "x": "lng"},
    "geometrySourceType": "seriesdata"
    },
    {
     "title": "Demographic Layer",
     "layerType": "tile",
     "url": "http://server.arcgisonline.com/ArcGIS/rest/services/Demographics/
USA_Population_Density/MapServer"
   },
   {
    "title": "Country Feature Layer",
    "layerType": "featurelayer",
    "geometrySourceType": "esri",
    "geometryLocateField": ["Country"],
    "geometryDataField": "name",
    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
    },
    {
     "title": "City Feature Layer",
     "layerType": "featurelayer",
     "geometrySourceType": "esri",
     "geometryLocateField": ["state_name"],
     "geometryDataField": "name",
     "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
USA_Counties_Generalized/FeatureServer/0"
    },
   ],
  }
 }
 *END
ENDSTYLE
END
```

The output is shown in the following image:


Layers
AVE Customer State Province Population
38M
28.5M
19M
9.5M
0
CANADA
MEXICO
Gulf of Mexico
1000km
600mi
Esri, HERE, DeLorme, NGA, USGS | ©2013 Esri

Reference: Controlling Display of the Table of Contents

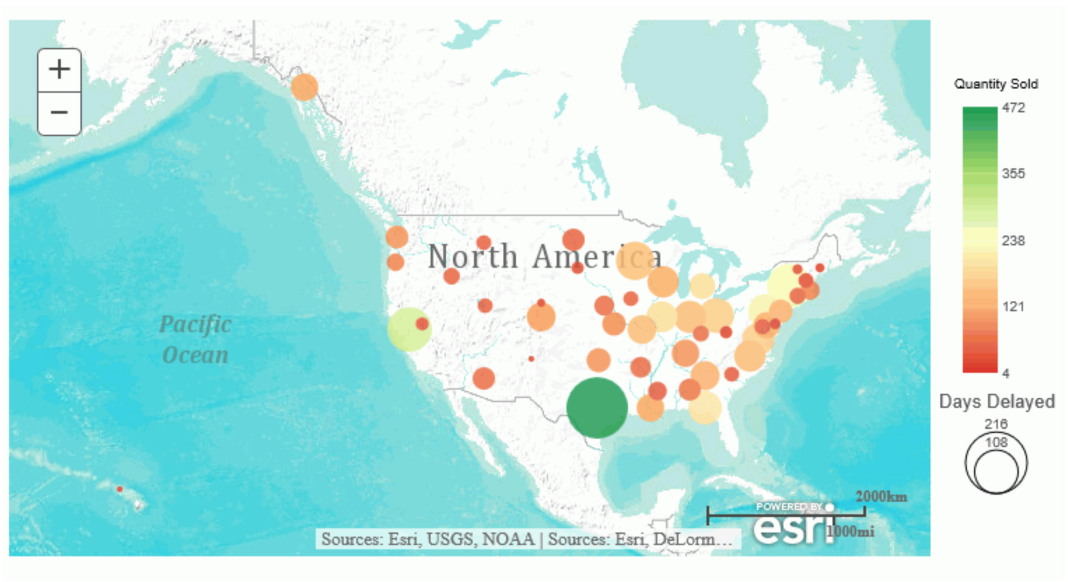
The table of contents (Layers) control is drawn because of the default *drawToc:true* property of the com.esri.map object. You can eliminate the table of contents control by setting this property to false.

Example: Removing the Table of Contents From an Esri Map Chart Using Reporting Server Syntax

The following request removes the Table of Contents from the map chart.

```
DEFINE FILE WF_RETAIL_LITE
GEOPOINT/A200 = GIS_POINT('4326', STATE_PROV_CAPITAL_LONGITUDE,
STATE_PROV_CAPITAL_LATITUDE);
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED QUANTITY_SOLD
BY GEOPOINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=color, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"extensions": {
  "com.esri.map": {
      "drawToc": false,
       "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "GEOPOINT"
                                            }
                                   }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "terrain"
                }
             ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.

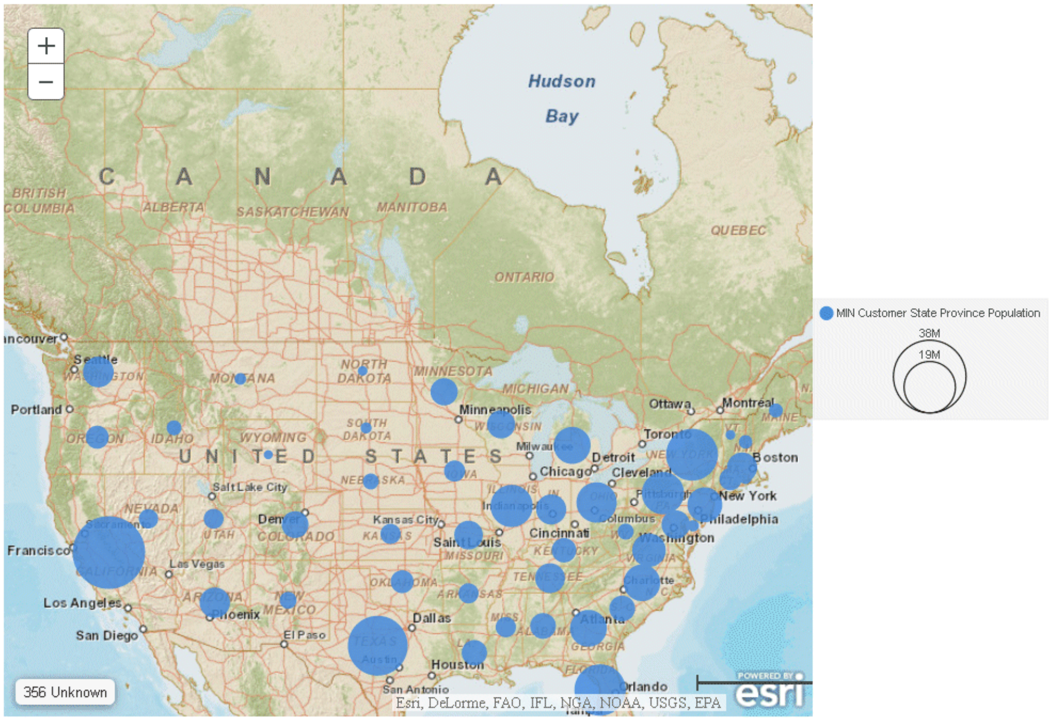


Example: Removing the Table of Contents From an Esri Map Chart Using Chart Engine Syntax

The following request removes the Table of Contents from the map chart.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=STATE_PROV_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.STATE_PROV_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": true},
"bubbleMarker": {"maxSize": "10%" },
"extensions": {
 "com.esri.map": {
  "drawToc":false,
  "overlayLayers":
  [
   {
   "title": "WebFOCUS Request Output",
   "layerType": "bubble",
   "geometrySourceType": "esri",
   "geometryLocateField": ["STATE_NAME"],
   "geometryDataField": "name",
   "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services/
USA_States_Generalized/FeatureServer/0"
   }
  ],
   "baseLayer": {
   "basemap": "streets",
   "center":[-97,48],
   "zoom":4
   }
  }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.

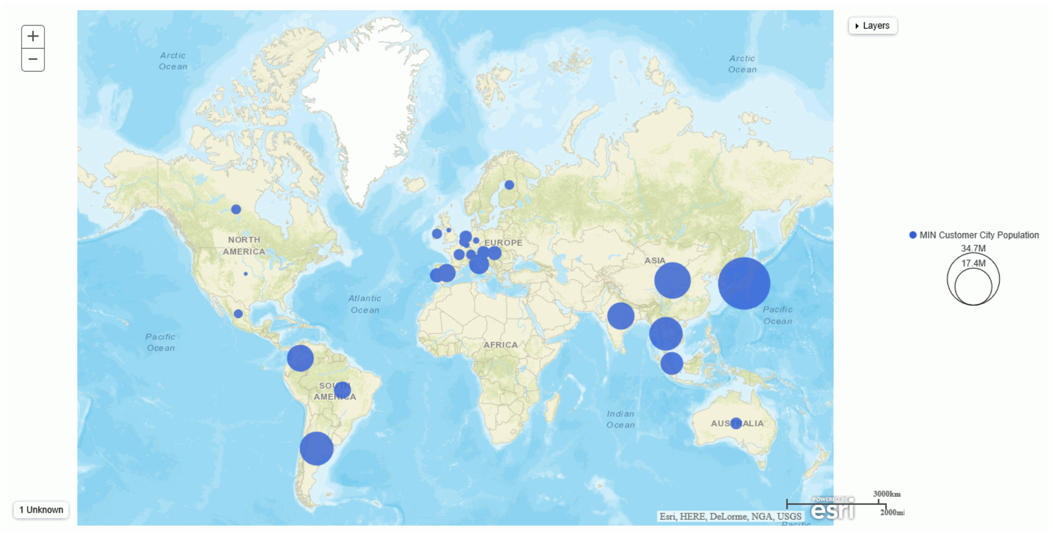


Reference: Handling Missing Geography

If the geography is not known for some locations in the request, an indication is placed on the chart. For example, consider the following request:

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COUNTRY_NAME, BUCKET=location, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {
    "visible": true
},
"bubbleMarker": {"maxSize": "10%" },
"extensions": {
 "com.esri.map": {
  "overlayLayers":             [
  {
  "title": "Population",
  "layerType": "bubble",
  "geometrySourceType": "esri",
  "geometryLocateField": ["Country"],
  "geometryDataField": "name",
  "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
   }
  ],
 "baseLayer": {
   "basemap": "streets"
  }
 }
}
*END
ENDSTYLE
END
```

The output is shown in the following image. When the mouse hovers over the box labeled *1 Unknown*, the missing geography value is listed:



You can use the information to screen out any missing location values or find an endpoint with values for the missing geographies.

Reference: Adding the Page Attribute Category to a Map Chart Request

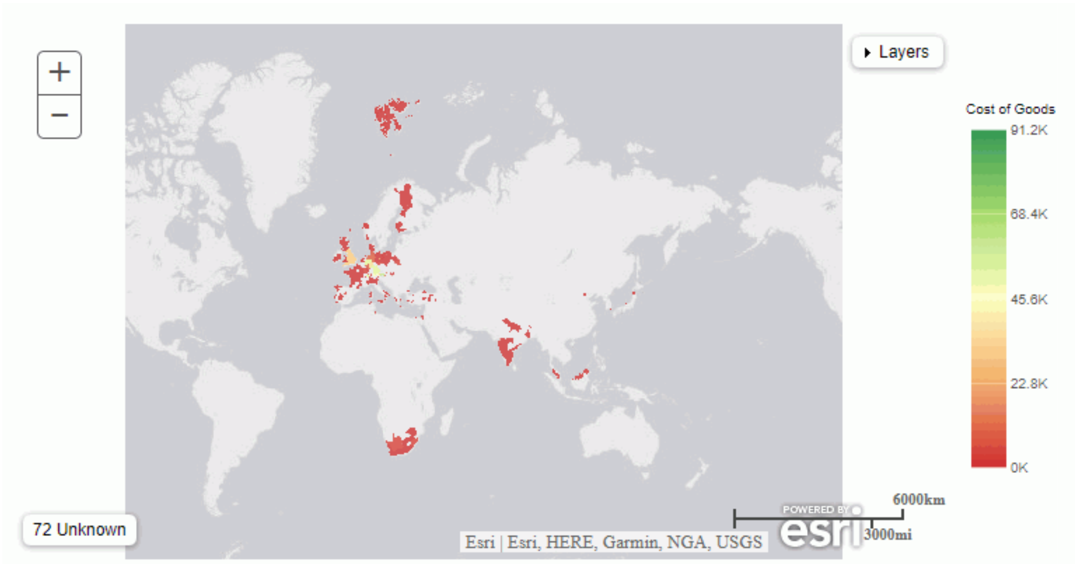
The *page* attribute generates separate charts based on the value of a high-level sort field.

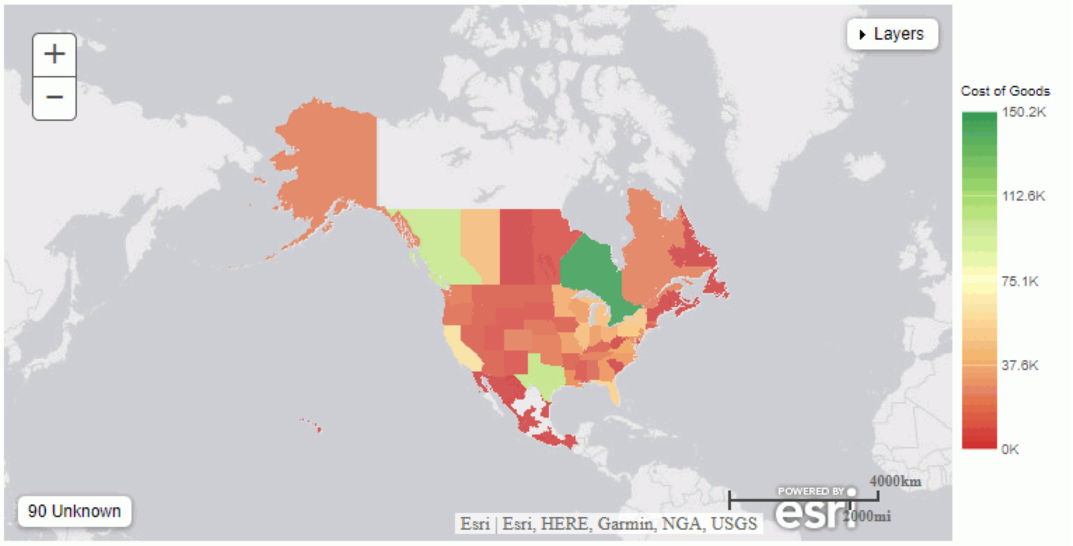
Example: Adding the Page Attribute to an Esri Chart Using Reporting Server Syntax

In the following request, BUSINESS_REGION is added as a high-level sort field to the request and is assigned to the *page* category.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY BUSINESS_REGION
BY STATE_PROV_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, TITLETEXT='WebFOCUS Chart', CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=N1, BUCKET=page, $
TYPE=DATA, COLUMN=N3, BUCKET=color, $
*GRAPH_JS_FINAL
"extensions": {
    "com.esri.map": {
        "overlayLayers":             [
                {
                    "ibiDataLayer": {
                        "map-metadata": {
                            "map_by_field": "STATE_PROV_NAME"
                        }
                    }
                }
            ],
        "baseMapInfo": {
            "customBaseMaps":                  [
                    {
                        "ibiBaseLayer": "gray"
                    }
                ]
        }
    }
}

*END
ENDSTYLE
END
```

The first two of the four generated chart pages are shown in the following image:



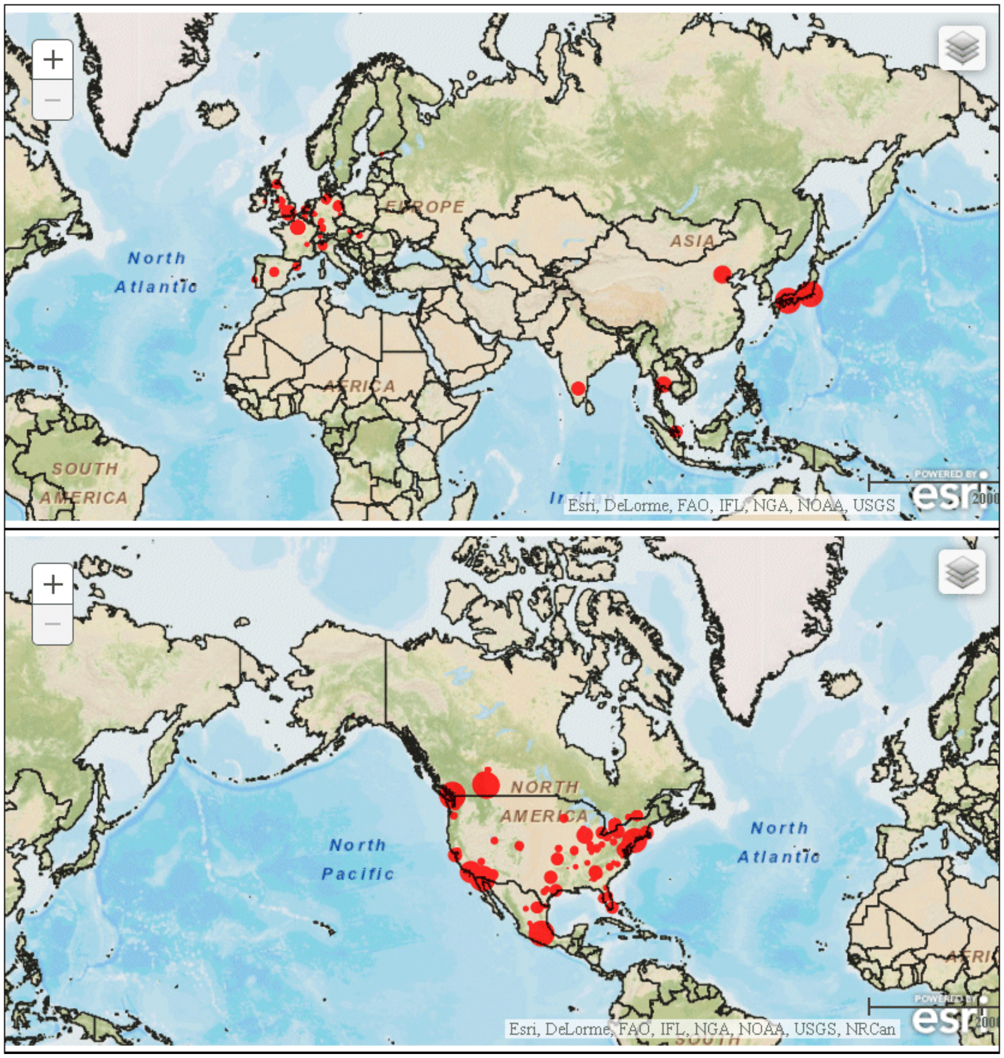


Example: Adding the Page Attribute to an Esri Chart Using Chart Engine Syntax

In the following request, BUSINESS_REGION is added as a high-level sort field to the request and is assigned to the *page* category. WebFOCUS calculates the minimum zoom level and other initial properties separately for each chart based on its geography, so the request adjusts the minimum zoom level in order to fill the bounding box for each chart fully on initial display.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.CITY_POPULATION
  BY BUSINESS_REGION
BY  CITY_LATITUDE
BY CITY_LONGITUDE

WHERE COUNTRY_NAME NE 'Taiwan'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=page, $
TYPE=DATA, COLUMN=CITY_LATITUDE, BUCKET=latitude, $
TYPE=DATA, COLUMN=CITY_LONGITUDE, BUCKET=longitude, $
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=size, $
*GRAPH_JS
"legend": {"visible": false},
"bubbleMarker": {"maxSize":"5%"},
"extensions": {
    "com.esri.map":
           {
     "baseLayer":{ "title" : "Base Layer",
     "basemap": "streets",
     "minZoom" : 2
 },
    "overlayLayers":             [
       {
    "title": "WebFOCUS Request Layer",
    "layerType": "bubble",
    "geometryXY": {
    "y": "lat",
    "x": "lng"},
    "geometrySourceType": "seriesdata"
    },
    {
    "title": "Country Feature Layer",
    "layerType": "featurelayer",
    "geometrySourceType": "esri",
    "geometryLocateField": ["Country"],
    "geometryDataField": "name",
    "url": "http://services.arcgis.com/P3ePLMYs2RVChkJx/arcgis/rest/services/
World_Countries_(Generalized)/FeatureServer/0"
    }
    ],
  }
}
*END
ENDSTYLE
END
```

The first two of the four generated chart pages are shown in the following image:



Generating Leaflet Map Charts

A Leaflet chart requires a mapProperties object in the *GRAPH_JS block of the WebFOCUS StyleSheet. It can include properties that set the map opacity, the zoom level, scale control properties, and the initial position. In addition, it must point to the base layer map tile and define the overlay layer.

Syntax: How to Specify Map Properties for Leaflet Charts

In the mapProperties object in the GRAPH_JS block of the StyleSheet, specify the map viewer engine (leaflet) and the map properties:

```
*GRAPH_JS
"mapProperties": {
"engine": "leaflet",
  "leafLet":{
   map_properties
}
}
  *END
```

where:

map_properties

Are the properties that control the map tiles, map overlays, and map controls in your chart.

You can also use an alternate map tiling engine.

Syntax: How to Define Base Layer Properties for a Leaflet Map

```
"mapProperties":{
"baselayers": [{
"title": "string",
"layerInfo": {
  "maxZoom": number,
  "minZoom": number,
  "attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
  "url": function(){
return "http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/
MapServer/tile/{z}/{y}/{x}";
}
}]
}
```

where:

`"title": "string"`

Is a title for the layer. This title appears when you click the table of contents control displayed at the top right of the map.

`"maxZoom": number`

Is the value -1 (fully automatic) or a number between 1 and 19 that defines the maximum zoom level allowed.

`"minZoom": number`

Is the value -1 (fully automatic) or a number between 1 and 19 that defines the minimum zoom level allowed.

`"url": function()`

Defines the initial zoom and center of the returned map service.

z

Is the initial zoom level, specified in the initPos:level property,

y

Is the latitude for the center of the initial view, specified in the initPos:center property,

x

Is the longitude of the initial view, specified in the initPos:center property,

Note:

- You can disable the zoom (+/-) control, but not remove it altogether, by making minZoom=maxZoom.
- The attribution property defines the copyright statement that appears on the bottom right of the map. The attribution included in the previous code displays the copyrights for the map providers included with WebFOCUS. If you use maps from a different provider, edit the attribution to contain the copyright information for your provider.
- The url property in the previous syntax points to one type of map (World_Street_Map) provided by Esri. If you use maps from a different provider, edit this property to point to your map tile server.

The following are different types of maps provided by Esri and the URL references needed to access them:

```
http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/
tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/
tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/
tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/World_Terrain_Base/MapServer/
tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/Canvas/World_Light_Gray_Base/
MapServer/tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/NatGeo_World_Map/MapServer/
tile/{z}/{y}/{x};
http://services.arcgisonline.com/ArcGIS/rest/services/Ocean_Basemap/MapServer/
tile/{z}/{y}/{x};
```

Since the maps displayed on any map chart are owned by the services that supply them, they require that you include a copyright attribution on your chart, which is one of the properties you must include in the base layer for Leaflet maps.

The following syntax shows the attribution and url properties for using the OpenStreetMap maps installed with WebFOCUS:

```
"attribution": function(){ return '&|copy; <a href="http://www.ibi.com">IBI</a> | ' +
'Map data &|copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors, Open Database Licence';}
    },
"url": function(){
return tdgchart.getScriptPath().replace(/\/tdg\/.*/,'') +
 '/maptiles/osm/{z}/{x}/{y}.png'}
```

The following syntax shows the attribution and url for another type of OpenStreet map provided by Mapbox, installed with WebFOCUS:

```
"attribution": "Map data &|copy; <a href='http://www.openstreetmap.org/
copyright'>OpenStreetMap</a> contributors, Open Database Licence | Map Tiles by <a
href='http://mapbox.com'>Mapbox</a>",

"url": "http://{s}.tiles.mapbox.com/v3/bclc-apec.map-rslgvy56/{z}/{x}/{y}.png",
```

Example: Defining Base Layer Properties for a Leaflet Map

The following request sets an initial position and zoom level and generates the map base layer.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
  "initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },
 "baselayers": [{
  "title": "ArcGIS_World_Street_Map",
  "layerInfo": {
   "maxZoom": 17,
   "minZoom": 0,
   "attribution": function(){return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
  "url": function(){return "http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}";
}
}]
}
},
*END
ENDSTYLE
END
```

On the resulting map, the request output is not represented, as that would go on the overlay layer:

+
−
Leaflet | © Information Builders | Map Tiles: © Esri

Reference: Connecting the Fields in the Location File to the Fields in the WebFOCUS Request

The following code shows a snippet of the beginning of the US.JSON location file, which lists US cities and states. The primary layer (map subdivision) is called *regions*. The position property contains the latitude and longitude of the city:

```
"map_type":"country",
 "country_name":"United States of America",
 "iso_a2":"US",
 "primary_layer":"regions",
 "layers":[{
  "type":"cities",
  "geocode":["city_name"],
  "units":"decimal_degrees",
  "geometry_type":"point",
  "primary_name_field":"city_name",
  "features":[{
   "city_name":"New York",
   "rank":"0",
   "region":"New York",
   "position":[-73.98,40.7499],
   "size":24.2
  },
  {
   "city_name":"Washington, D.C.",
   "rank":"0",
   "region":"District of Columbia",
   "position":[-77.0094,38.8995],
   "size":22
  },
  {
   "city_name":"Los Angeles",
   "rank":"0",
   "region":"California",
   "position":[-118.1799,33.9899],
   "size":23.6
```

The following code shows the beginning of the state definitions. Note that the *type* is *regions* and the fields include *state_name* and *state_abbr*. Each state definition also includes latitudes and longitudes for its borders:

```
"type":"regions",
  "geocode":["state_abbr","state_name","state_fips","iso_3166_2"],
  "units":"decimal_degrees",
  "geometry_type":"polygon",
  "primary_name_field":"state_name",
  "primary_abbr_field":"state_abbr",
  "primary_id_field":"state_abbr",
  "features":[{
   "state_name":"Hawaii",
   "state_abbr":"HI",
   "state_fips":"15",
   "iso_3166_2":"US-HI",
   "borders":[[[-155.9633,19.0815],[-155.9432,19.3442],
[-156.1192,19.7364],[-155.8843,20.0183],[-155.959,20.2268],
[-155.8367,20.3212],[-155.1403,19.9894],[-154.7566,19.5069],
[-154.9396,19.3128],[-155.466,19.0953],[-155.649,18.8705]],
[[-156.5651,20.7389],[-156.7519,20.9136],[-156.6058,21.0925],
[-156.451,20.9687],[-156.2685,21.0037],[-155.9589,20.875],
[-155.9267,20.7817],[-156.0299,20.604],[-156.375,20.5289],
[-156.4845,20.579],[-156.5334,20.4629],[-156.75,20.4543],
[-156.7336,20.5818],[-156.4957,20.6915]],[[-158.3176,21.5391],
[-158.3242,21.6156],[-157.9417,21.7545],[-157.6505,21.485],
[-157.6009,21.2829],[-157.8102,21.1992],[-158.1347,21.2423]],
[[-159.7936,21.9397],[-159.8481,22.076],[-159.6129,22.2722],
[-159.3896,22.2875],[-159.25,22.1757],[-159.2843,21.9406],
[-159.4328,21.8207]],[[-157.015,21.2611],[-156.722,21.226],
[-156.6527,21.125],[-156.8779,20.9933],[-157.3637,21.0923],
[-157.2812,21.2725]],[[-175.9394,27.6882],[-176.0396,27.7582],
[-175.921,27.8214],[-175.9814,27.9132],[-175.7043,27.9827],
[-175.7341,27.7388]],[[-157,20.6968],[-157.1112,20.9312],
[-156.8684,20.9643],[-156.75,20.8076],[-156.875,20.6909]],
[[-160.2589,21.7428],[-160.2799,21.9206],[-160.0634,22.0726],
[-160.031,21.8693]],[[-166.1722,23.587],[-166.3844,23.8319],
[-166.307,23.9248],[-166.1823,23.9023],[-166.1067,23.7425]],
[[-178.3014,28.3283],[-178.4404,28.4051],[-178.375,28.51],
[-178.2384,28.4561]],[[-174.0064,25.9902],[-174.0547,26.1077],
[-173.9579,26.1356],[-173.9033,26.06]],[[-171.7374,25.7065],
[-171.8096,25.7813],[-171.75,25.8379],[-171.6679,25.7812]],
[[-161.9318,23.0053],[-161.9825,23.0912],[-161.8875,23.1103]],
[[-164.723,23.5217],[-164.7611,23.5934],[-164.6826,23.6228],
[-164.6453,23.5515]],[[-160.5707,21.6066],[-160.5918,21.6858],
[-160.5087,21.7032],[-160.4849,21.6324]],[[-168.0235,24.9544],
[-168,25.0522],[-167.9442,24.9898]]]
  },
```

In order to connect the map data with the data in your data source, you must identify the *type* of subdivision you are using and the field in the location file that has the same data as the sort field in your WebFOCUS request. For example, using this location file, if your data source has state names, you must identify *regions* as the type of subdivision to use and *state_name* as the field name to use from the location file.

Syntax: How to Define Overlay Layer Properties for a Leaflet Map

```
"mapProperties":{
"overlayLayers": [{
    "title": "string",
    "dataLookup": "properties.field",
    "layerInfo": {
     "maxZoom": number,
     "minZoom": number,
     "type": "string"
    },
    "type": "string",
    "url": function(){ return tdgchart.getScriptPath() + "map/
location.json"}
   }]
}
```

where:

`"title": "string"`

Is a title for the layer. This title appears when you click the table of contents control displayed at the top right of the map.

`"dataLookup": "properties.field"`

Is the name of the field in the location file that contains the same data as the sort value in your request. Not required for a latitude/longitude-based map layer.

`"layerInfo":`

Defines the minimum zoom, the maximum zoom, and the type of data being returned.

`"maxZoom": number`

Is the value -1 (fully automatic) or a number between 1 and 19 that defines the maximum zoom level allowed.

`"minZoom": number`

Is the value -1 (fully automatic) or a number between 1 and 19 that defines the minimum zoom level allowed.

`"type": "string"`

Is the type of geographical subdivision as listed in the location file.

`"type": "string"`

Defines the type of location file being used. Valid values are:

- "tdg", which supports both the JSON files distributed with WebFOCUS and geoJSON location files.
- "geojson", which supports only geoJSON location files.
- "latlng", which specifies longitude (x) and latitude (y) values will be supplied in the WebFOCUS request.

`"url": function`

Defines the map location file being returned.

location

Is the name of the location file, for example, US.JSON. If you are using geoJSON files or additional territories not supplied with WebFOCUS, change the extension to the correct value and point to where these files are stored.

Note:

- The url attribute points to the location files (the tdgchart engine included with WebFOCUS has this path configured during WebFOCUS installation. If you are using geoJSON files, change the extension to the correct value and point to where those files are stored.

  The following code shows how to use the location file sample_ggsales_regions.geojson, that was installed as a sample during WebFOCUS installation:

  ```
  "url": function() { return tdgchart.getScriptPath().replace(/\/tdg
  \/.*/,"/web_resource/map/") +"sample_ggsales_regions.geojson"}
  ```

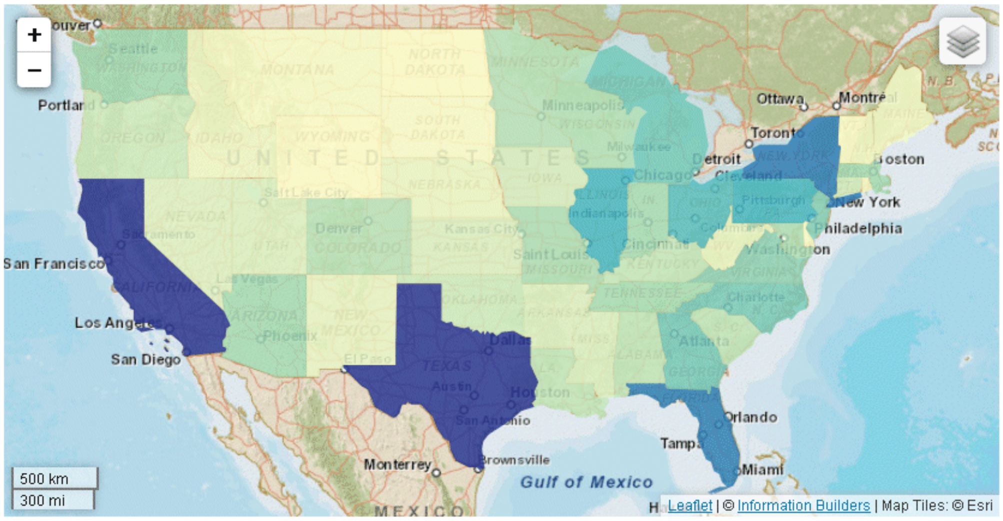
Example: Defining the Overlay Layer for a Leaflet Choropleth Map

The following request defines an overlay layer and the layer and scale controls. The *type* property is set to "regions" and the *dataLookup* property is set to "properties.state_name" because these are the values specified in the location file that match the data in the WF_RETAIL_LITE data source.

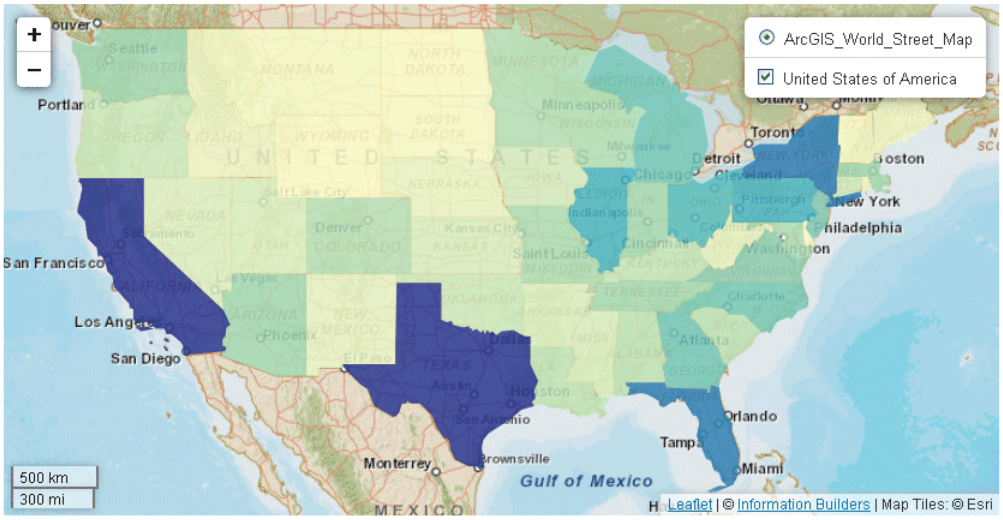
Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },

   "overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
}],
  "controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 17,
"minZoom": 0,
"attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
"url": function(){return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},

*END
ENDSTYLE
END
```

The request output is a choropleth on top of the base layer, in which the median populations are defined by the colors of the polygons representing the different states:



When the mouse hovers over the layer control, the title for each layer displays:



On this choropleth, there is no indication of the population values that the colors represent or the actual values for each state. You can add these objects using the legend and tooltip properties.

Reference: Specifying a Choropleth With a Point Location

If your chart request specifies a choropleth map (ON GRAPH SET LOOKGRAPH CHOROPLETH), but the location field represents a point instead of a polygon, a bubble marker (circle) is drawn to represent the location. The markers will all be the same size. The field assigned to the color category will be used to color the markers.

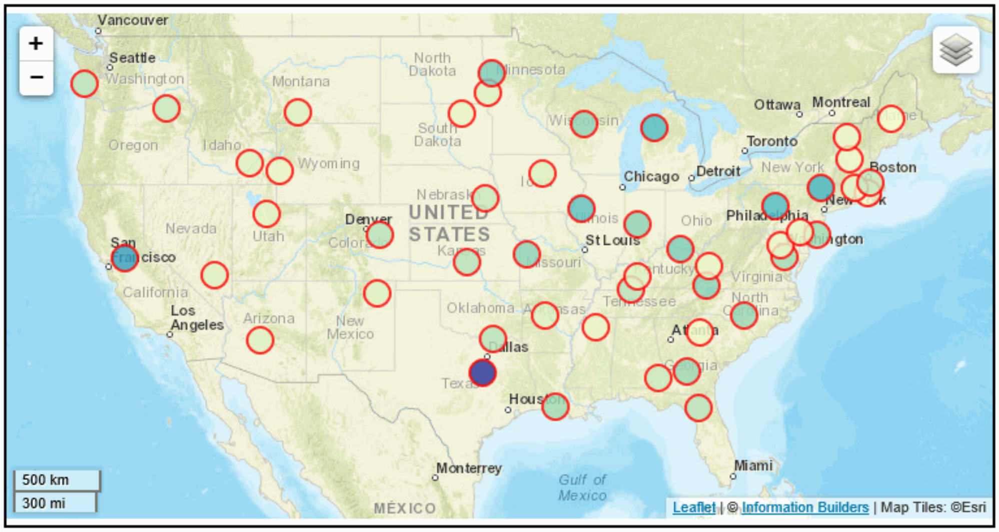
Example: Removing Default Borders From Bubble Markers in a Leaflet Choropleth Chart

By default, in a Leaflet map chart that uses a point location with a choropleth LOOKGRAPH parameter, the markers have a red border. You can remove this border by setting the border width to zero for the series.

The following request specifies a choropleth chart but uses a point location (latitude and longitude).

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
WHERE COUNTRY_NAME EQ 'United States'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
"bubbleMarker":{"maxSize":"10%"},
"mapProperties": {
"leaflet": {
"initPos": {
"center": [37.8, -96],
"level": 4
},
"overlayLayers": [{
"title": "US",
"layerInfo": {
"type": "regions"
},
"type": "latlng",
"url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
}],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 16,
"attribution": function() { return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy;Esri"; }
},
"url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}'; }
}]
}
}
*END
ENDSTYLE
END
```

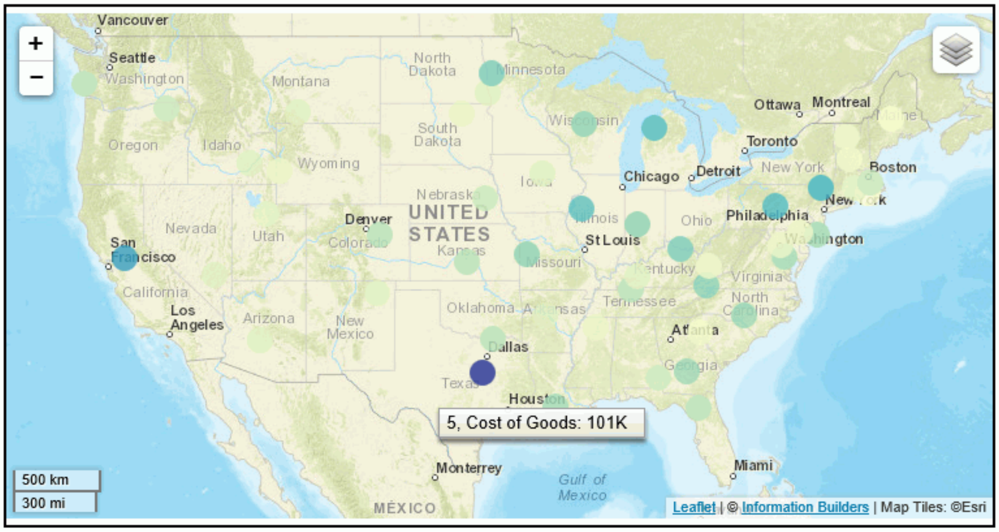
The output is shown in the following image.



The following version of the request uses the series properties to make the border width zero (0).

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
WHERE COUNTRY_NAME EQ 'United States'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
"bubbleMarker":{"maxSize":"10%"},
"mapProperties": {
"leaflet": {
"initPos": {
"center": [37.8, -96],
"level": 4
},
"overlayLayers": [{
"title": "US",
"layerInfo": {
"type": "regions"
},
"type": "latlng",
"url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
}],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 16,
"attribution": function() { return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy;Esri"; }
},
"url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}'; }
}]
}
},
"series": [
                    {
                      "series": "all",
                                 "border": {
                                  "width": 0
                                 }
                    }
                 ]
*END
ENDSTYLE
END
```

The output is shown in the following image.



Defining Initial Position

The following facts describe automatic positioning in map charts:

- By default, the map viewer determines a bounding box that completely encapsulates all of the markers in your data source and chooses a zoom level that nicely bounds your data. For example, if your data is 48 continental US states, the zoom level will be set so that you see only the continental US, with perhaps a bit of Mexico and Canada also visible.
- Also by default, the map viewer sets a center point that is the exact middle of the automatic bounding box. In the continental USA, this puts the mid-West states in the middle of the screen.

Instead, you can specify your own initial position and initial zoom level in the map properties included in your request.

Syntax: How to Define the Initial Position

```
"initPos":{
  "center":[latitude, longitude],
  "level": number}
```

where:

`"center":[latitude, longitude]`

Are the latitude and longitude for the center position of the map.

`"level": *number*`

Is the initial zoom level of the chart. Valid values are numbers from 1 (whole earth) to 19.

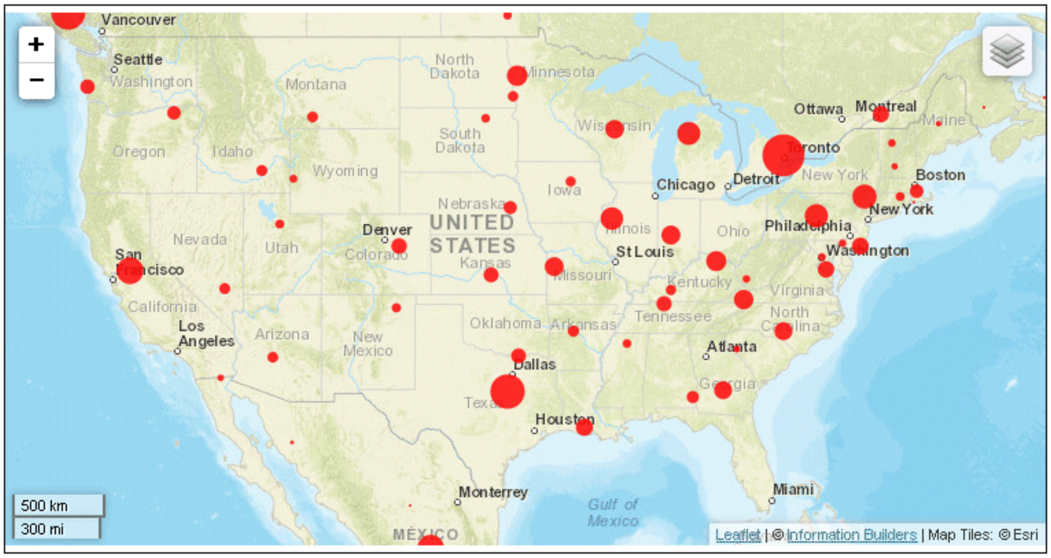
Example: Defining initial Position and Zoom on a Leaflet Map Using Latitude and Longitude

The following request generates a bubblemap based on latitude and longitude. The initial position and zoom are set so that the initial view of the map is the United States.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"bubbleMarker":{"maxSize":"10%"},
"mapProperties": {
  "leaflet": {
   "initPos": {
    "center": [37.8, -96],
     "level": 4
              },
 "overlayLayers": [{
  "title": "US",
  "layerInfo": {
    "type": "regions"
            },
  "type": "latlng",
  "url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
            }],
  "baselayers": [{
   "title": "ArcGIS_World_Street_Map",
   "layerInfo": {
    "maxZoom": 16,
    "attribution": function() {  return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy;
Esri"; }
                },
 "url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}'; }
            }]
    }
}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Adjusting the Opacity on Maps

You can specify an opacity (degree of transparency) for either a choropleth or bubblemap using the mapOpacity property.

Syntax: How to Set the Opacity of a Choropleth or BubbleMap

In the mapProperties block of the StyleSheet, specify the following property:

```
"mapOpacity": number
```

where:

`"mapOpacity": number`

Is a number between 0 (transparent) and 1 (solid). The default value is .82.

Adjusting Elements In and Around the Map Viewer

You can control elements such as the chart legend, the map scale, and the allowed zoom levels of the map.

The legend can be formatted, moved, or hidden entirely. For information, see *Legend Properties* on page 279.

Adjusting the Map Scale and Layers Controls

The scale control can be omitted or set to show miles, kilometers, or both. The layers control can be included or omitted. By default, both controls are shown on the output if the *controls* object is omitted from the leaflet block in the StyleSheet.

Syntax: How to Adjust the Map Scale and Layers Controls

Add the *controls* object to the leaflet block of the StyleSheet:

```
 "leaflet":{
 "controls": [
    {"control": "L.Control.Layers"},
    {"control": "L.Control.Scale",
     "options": {
      "imperial": boolean,
      "metric": boolean}
    }
   ],
   ...
}
```

where:

`"imperial":` *`boolean`*

Controls the display of miles. Valid values are:

- true, which shows miles. This is the default value.
- false, which does not show miles.

`"metric":` *`boolean`*

Controls the display of kilometers. Valid values are:

- true, which shows kilometers. This is the default value.
- false, which does not show kilometers.

For example, the following properties show both miles and kilometers on the scale:

```
"leaflet": {
    "controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
         ],
  ...
    }
```

The following properties cause the scale not to be shown at all:

```
"leaflet": {
    "controls": [
    {"control": "L.Control.Layers"},
    {
         }
   ],
   ...
   }
```

The following properties hide both the layers control and the scale control:

```
"leaflet": {
       "controls": [
                 ],
   ...
        }
```

Incorporating Additional Chart Properties in a Map Chart

You can use some of the general chart properties, as well as properties specific to heatmap and bubble charts, to adjust your chart output.

Adjusting the Heat Scale on Choropleth Maps

You can use the colorScale property to define colors for your choropleth map. You can list any number of colors, using color names, RGB values, or hex values. For example, the following code defines a red, white, and blue color scale.

```
"colorScale": ("colors":["#C4161C", "white", "rgb(0,0,255)"]}
```

You can also define discrete color bands for the map and legend using "colorScale": "colorMode" properties. For information, see *Defining a Color Scale Color Mode* on page 771.

With Reporting Server syntax, the legend is visible by default. You can set "legend": {"visible": false} to remove the legend.

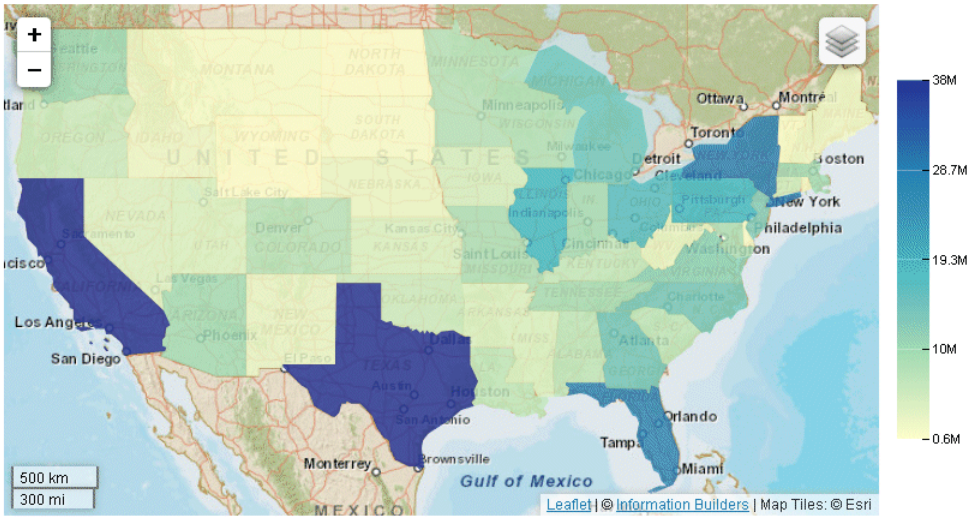
Example: Adding a Heat Scale Legend to a Leaflet Choropleth Chart

The following request makes the legend visible in a Leaflet chart.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
  "legend": {"visible":true},
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },
"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
"controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],

"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 17,
"minZoom": 0,
"attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | "  + "Map Tiles: &|copy;
Esri";}
},
"url": function(){return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},
*END
ENDSTYLE
END
```

The output now has a legend that defines the color for each population range.



Adjusting the Hover Color on a Leaflet Map Chart

As with any other type of chart, with a Leaflet map chart you can define a color or opacity setting for adjusting the display of a marker when the mouse hovers over it. To do this, you use the MouseOverIndicator property.

For example, the following property makes the selected chart area or marker 70% opaque when the mouse hovers over it.

```
"mouseOverIndicator": {
  "color": "70%"
}
```

The following property makes the marker yellow when the mouse hovers over it:

```
"mouseOverIndicator": {
  "color": "yellow"
}
```

For more information, see *Formatting the Mouse Over Indicator (mouseOverIndicator)* on page 804.

Adjusting the Color and Size of Markers on Bubblemaps

You can use series-specific properties to control the color and border of markers on any bubble chart, including a bubblemap.

For example, the following code makes the markers red with transparent borders.

```
"series": [
  {"series":0, "color": "red", "marker":{border: {color: "transparent"}}}
]
```

For information about marker properties, see *Defining the Size, Border, Color, Shape, and Rotation of Series Markers* on page 452.

In addition, you can control the size of bubble markers on a bubblemap or a bubble chart using the bubbleMarker: maxSize property. This property is not part of the mapProperties block, it is a property of bubble charts.

Syntax: How to Set the Maximum Size of Bubbles on a Bubblemap

Use the following property to specify the radius of the largest bubble:

```
"bubbleMarker": {"maxSize": size}
```

where:

`maxSize: size`

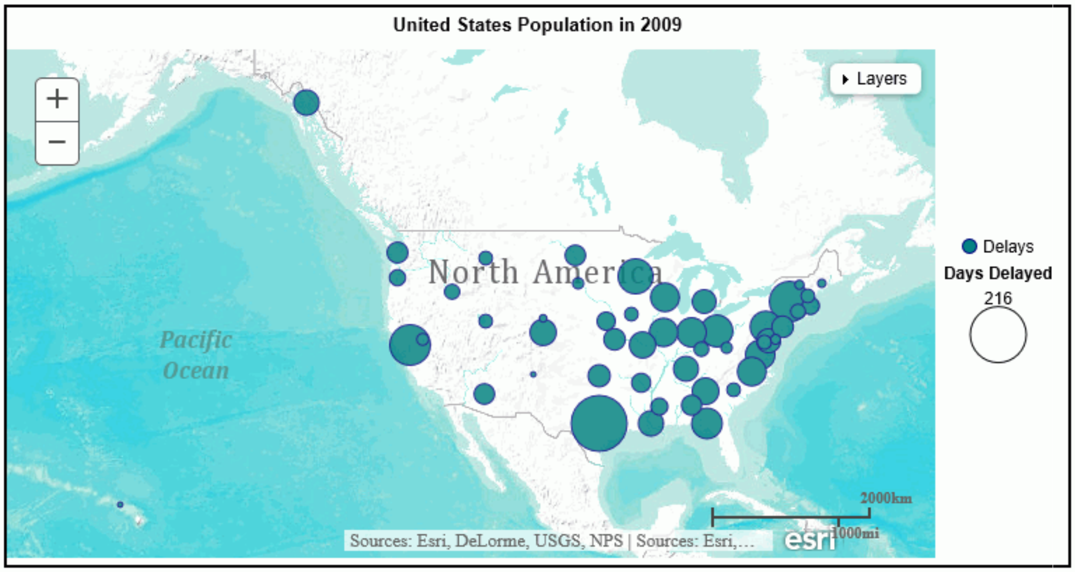
Determines the radius of the largest bubble. It can be a number of pixels or a percent string enclosed in single quotation marks and including a percent symbol (from '1%' to '100%'). If you use a percent string, it takes the minimum of the width and height of the chart container and then calculates the percent.

Example: Formatting Bubbles on an Esri Bubblemap

The following request generates a bubblemap using Reporting Server syntax. Using the "bubbleMarker" object, it sets a maximum size for the bubbles. Using the series marker properties, it makes the bubbles teal with a navy border and defines a label to appear in the legend. The title property assigns a title to the chart.

```
DEFINE FILE WF_RETAIL_LITE
GEOPOINT/A200 = GIS_POINT('4326', STATE_PROV_CAPITAL_LONGITUDE,
STATE_PROV_CAPITAL_LATITUDE);
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY GEOPOINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"bubbleMarker": {"maxSize": "10%"},
"series":[
{"series": 0, "color": "teal", "marker": {"border":{"color": "navy",
"width": 1}},"label": "Delays"}],
"title": {"visible": true, "text": "United States Population in 2009"},
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "GEOPOINT"
                                            }
                                   }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "terrain"
                }
            ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.


United States Population in 2009
Layers
North America
Pacific Ocean
Delays
Days Delayed
216
2000km
1000mi
Sources: Esri, DeLorme, USGS, NPS | Sources: Esri,...
esri

Note: You can also format the bubble markers using chart engine syntax.

Example:

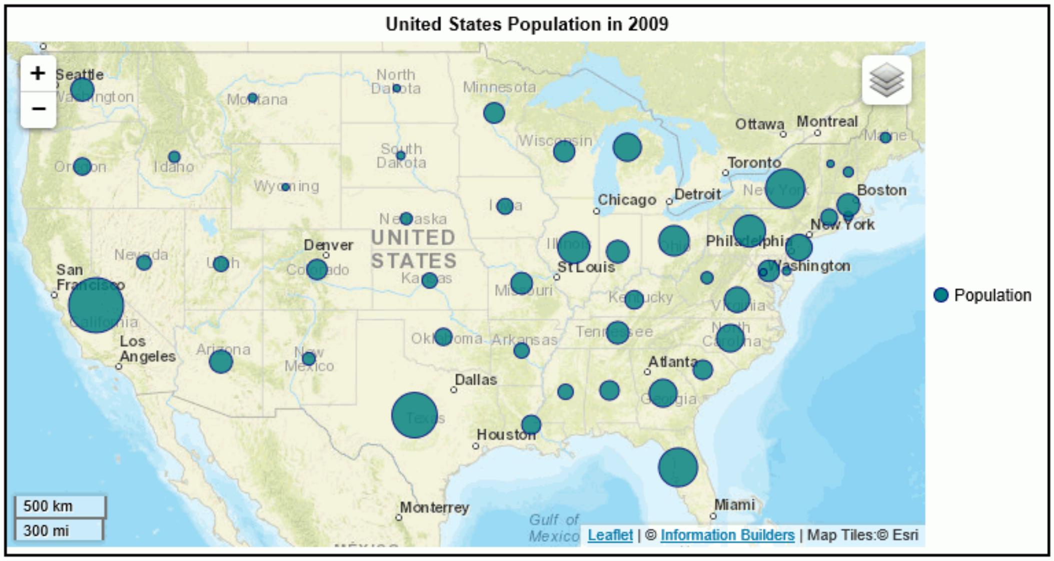
Formatting Bubbles on a Leaflet Bubblemap

The following request generates a bubblemap. Using the "bubbleMarker" object, it sets a maximum size for the bubbles. Using the series marker properties, it makes the bubbles teal with a navy border and defines a label to appear in the legend. The title property assigns a title to the chart.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
WHERE TIME_YEAR EQ 2009
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },

"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
"controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 17,
"minZoom": 0,
"attribution": function(){return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
"url": function(){return  'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},
"legend": {"visible":true},

"bubbleMarker": {maxSize: "10%" },
"series":[{"series":0, "color": "teal", "marker": {"border":{"color": "navy", "width":
1}},"label": "Population"}],
"title": {"visible": true, "text": "United States Population in 2009"} ,
*END
ENDSTYLE
END
```

The output is:



Defining Colors and Tooltips on Choropleth Charts

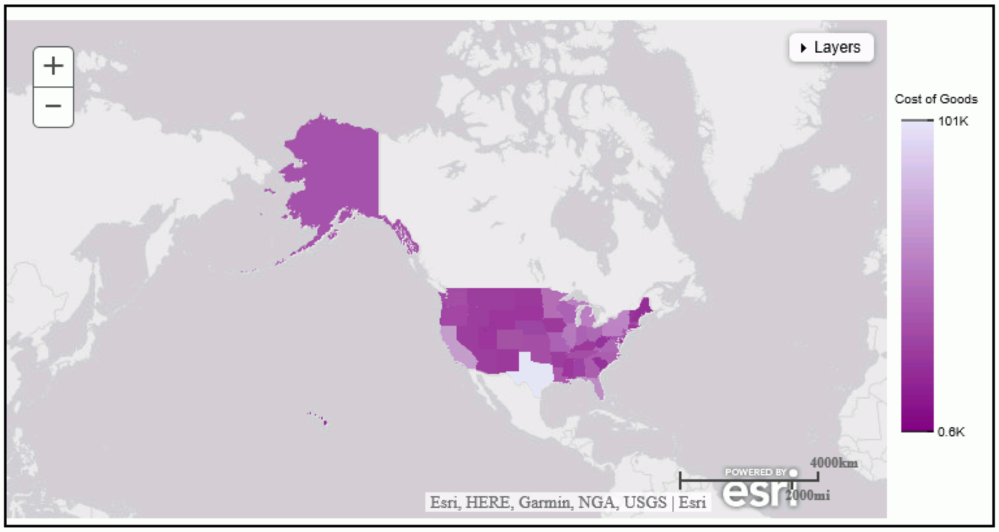
You can define color scales and tooltips on choropleth charts. You can also visualize the colors as discrete color bands. For information, see *Defining a Color Scale Color Mode* on page 771.

Example: Defining Colors and Tooltips on Esri Choropleth Charts

The following request generates a choropleth with a color scale consisting of purple and lavender using Reporting Server syntax. It also defines HTML-style tooltips.

```
DEFINE FILE wf_retail_lite
FIELD1/A30 (GEOGRAPHIC_ROLE=STATE) = STATE_PROV_NAME;
END
GRAPH FILE wf_retail_lite
SUM COGS_US
BY STATE_PROV_NAME
BY FIELD1
WHERE COUNTRY_NAME EQ 'United States'
AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=COGS_US, BUCKET=color, $
*GRAPH_JS_FINAL
"colorScale": {"colors": ["purple", "lavender"] },
"htmlToolTip": {"enabled": true, "snap": true},
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "FIELD1"
                                    }
                                 }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
             ]
          }
                          }
                   }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Note: You can also define colors and tooltips using chart engine syntax.

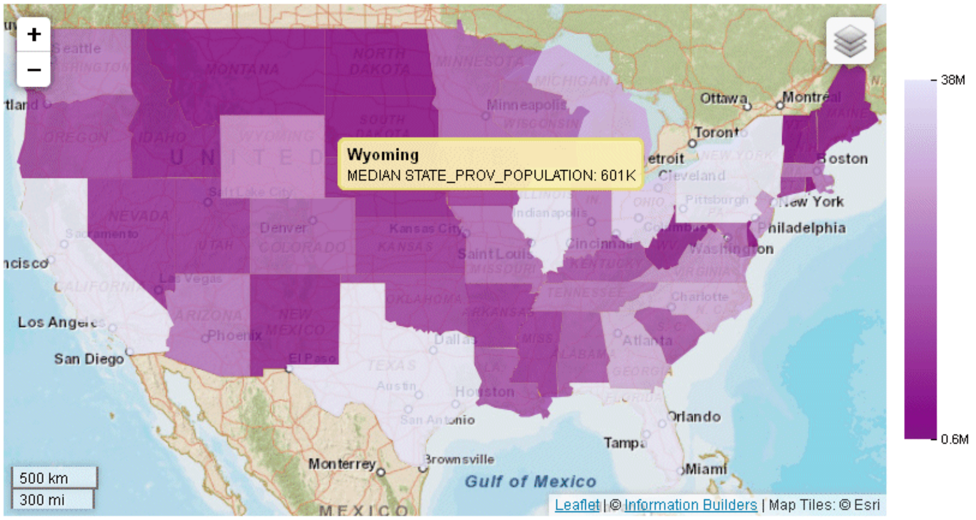
Example: Defining Colors and Tooltips on Leaflet Choropleth Charts

The following request generates a choropleth with a color scale consisting of purple and lavender. It also defines automatic HTML-style tooltips.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET STYLE *
*GRAPH_JS
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },

"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
"controls": [
{"control": "L.Control.Layers"},
{
"control": "L.Control.Scale",
"options": {
"imperial": true,
"metric": true
}
}
],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 17,
"minZoom": 0,
"attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
"url": function(){return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},
"legend":{"visible":true},
"title": {"visible": true, "text": "United States Population in 2010"},

"colorScale": {"colors": ["purple", "lavender"] },
"htmlToolTip": {"enabled": true, "snap": true},
"series": [{"series": "reset", "tooltip":"auto"}]   ,
*END
 ENDSTYLE
END
```

The output is:


Wyoming
MEDIAN STATE_PROV_POPULATION: 601K
38M
0.6M
500 km
300 mi
Leaflet | © Information Builders | Map Tiles: © Esri

Displaying Data Text Labels on a Bubblemap

Data text labels will display on a bubblemap if you set the "dataLabels": "visible" property to true:

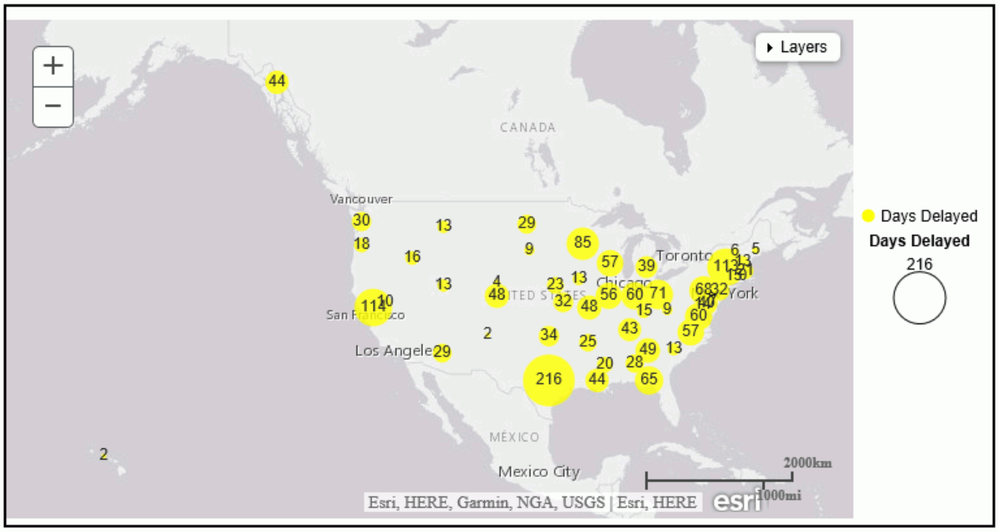
```
"series": [{"series":0,
               "dataLabels": {"visible": true}
                         }
                        ],
```

Example: Showing Data Text Labels on an Esri Bubblemap

The following request generates a bubblemap with data text labels using Reporting Server syntax.

```
DEFINE FILE WF_RETAIL_LITE
GEOPOINT/A200 = GIS_POINT('4326', STATE_PROV_CAPITAL_LONGITUDE,
STATE_PROV_CAPITAL_LATITUDE);
END
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY GEOPOINT
WHERE COUNTRY_NAME EQ 'United States' AND STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=DAYSDELAYED , BUCKET=size, $
*GRAPH_JS_FINAL
"bubbleMarker": {"maxSize": "10%"},
"series": [{"series":0, "color": "yellow", "dataLabels":{"visible":true}}],
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-geometry": {
                       "map_by_field": "GEOPOINT"
                                            }
                                   }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
             ]
          }
                          }
                     }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Note: You can also show data text labels using chart engine syntax.

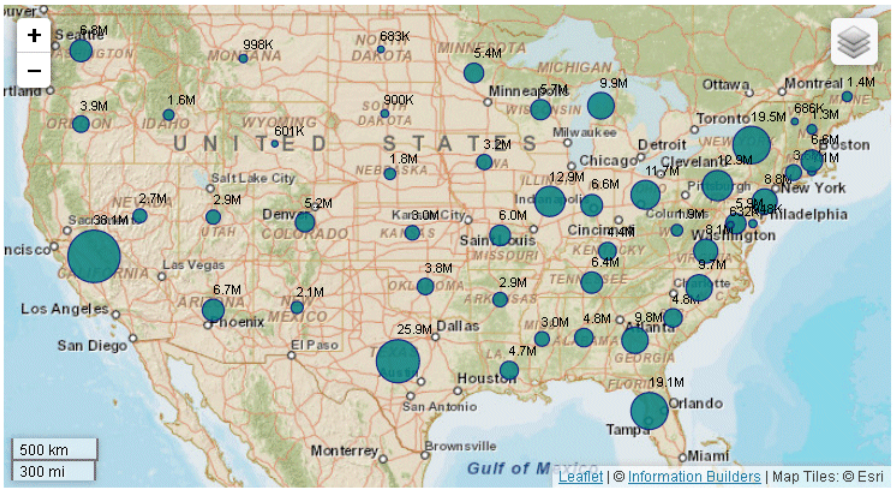
Example: Showing Data Text Labels on a Leaflet Bubblemap

The following request generates a bubblemap with data text labels.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MDN.STATE_PROV_POPULATION
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
WHERE TIME_YEAR EQ 2009
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
 "series": [{"series":0, "color": "teal", "border":{"color": "navy", "width": 1}}],
"dataLabels": {
"visible": true
},
"legend": {"visible":false},
"bubbleMarker": {"maxSize": "10%" },
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
  "controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],
 "baselayers": [{
   "title": "ArcGIS_World_Street_Map",
   "layerInfo": {
     "maxZoom": 17,
     "minZoom": 0,
      "attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
  "url": function(){return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},
*END
ENDSTYLE
END
```

The output is shown on the following image:



Displaying Location Names as Data Text Labels on a Choropleth

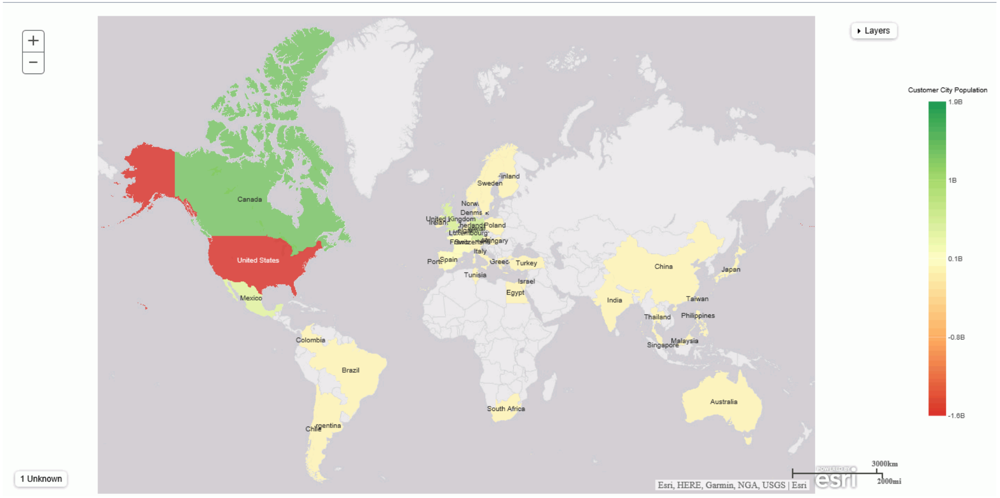
The dataLabels object can display location names using the "content": "name" property.

Example: Displaying Location Names as Data Text Labels on an Esri Map Chart

The following request uses Reporting Server syntax to apply the "content": "name" property of the dataLabels object in order to display country names as data text labels.

```
GRAPH FILE wf_retail_lite
SUM CITY_POPULATION
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/ibi_themes/Warm.sty,$
TYPE=REPORT, CHART-LOOK=com.esri.map, $
TYPE=DATA, COLUMN=CITY_POPULATION, BUCKET=color, $
*GRAPH_JS_FINAL
"series": [{"series":0, "dataLabels":{"visible":true, "content": "name"}}],
"extensions": {
  "com.esri.map": {
      "overlayLayers":
      [
        {
         "ibiDataLayer": {
                  "map-metadata": {
                       "map_by_field": "COUNTRY_NAME"
                                 }
                                 }
         }
      ],
      "baseMapInfo":
         {
          "customBaseMaps":
           [
               {
                "ibiBaseLayer": "gray"
                }
            ]
          }
                          }
                  }
*END
ENDSTYLE
END
```

The output is shown in the following image.



Note: You can also show location names as data text labels using chart engine syntax.

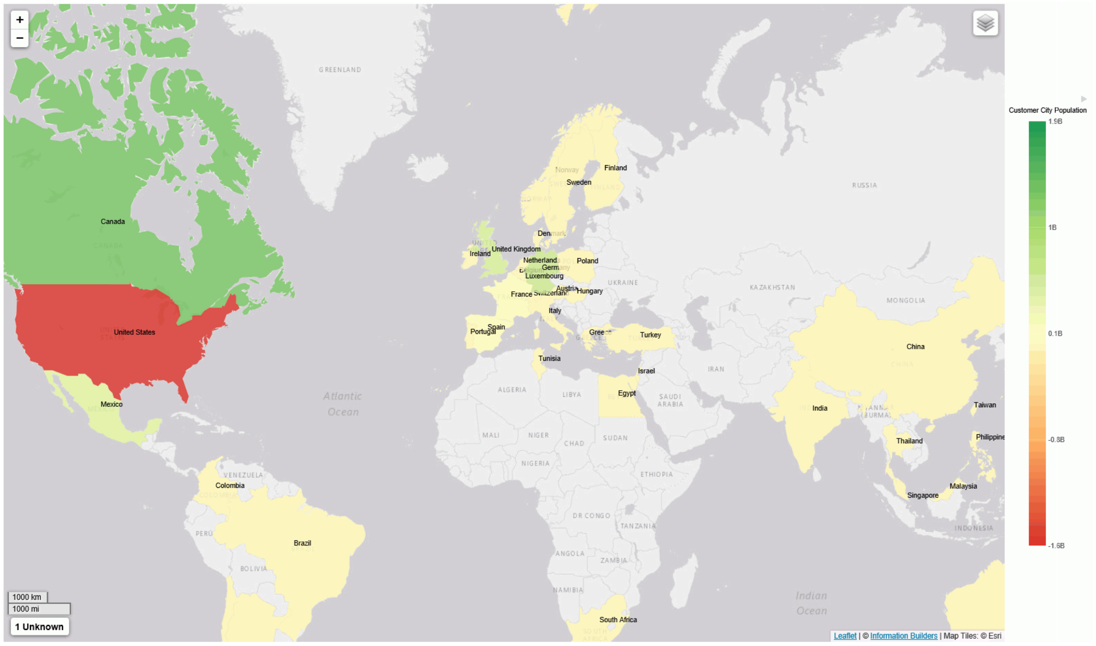
Example: Displaying Location Names as Data Text Labels on a Leaflet Map Chart

The following request uses the "content": "name" property of the dataLabels object to display country names as data text labels.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM CITY_POPULATION
BY COUNTRY_NAME
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=N1, BUCKET=location, $
TYPE=DATA, COLUMN=N2, BUCKET=color, $
*GRAPH_JS_FINAL
"dataLabels": {"visible": true, "content": "name"},
"mapProperties": {
    "leaflet": {
      "baselayers":             [
                {
         "layerInfo": {
            "maxZoom": 16,
            "attribution": function() {  return "&|copy; <a target='_blank'
href='http://www.InformationBuilders.com'>Information Builders</a> | " +
               "Map Tiles: &|copy; Esri"; }
                    },
                 "title": "ArcGIS_World_Street_Map",
                 "url": function() { return 'http://services.arcgisonline.com/ArcGIS/
rest/services/Canvas/World_Light_Gray_Base/MapServer/tile/{z}/{y}/{x}'; }
                }
            ],
        "overlayLayers":             [
                {
                 "dataLookup": "properties.country_name",
                 "layerInfo": {
                     "type": "countries"
                    },
                "type": "tdg",
                "title": "World",
                "url": function() {  return tdgchart.getScriptPath() + 'map/
world.json' }
                }
            ]
    }
},
*END
ENDSTYLE
END
```

The output is shown in the following image.



Chapter 14 Adding Your Own Chart Types to the Chart Library

WebFOCUS supports the ability to add new, custom chart types to its list of built-in charts. These custom chart types are called *extensions* or *plug-ins*. An extension is a block of code that accesses resources external to WebFOCUS. This chapter describes the structure of an extension and the steps necessary to create your own and add it to the chart library.

In this chapter:

Introducing Chart Extensions

Chart extensions are written in JavaScript. The visual part of a visualization can be drawn with HTML, Canvas, or SVG. Extensions can include external CSS and JS libraries (such as d3), which can be used to build almost any visualization. The WebFOCUS Extension API is limited to new, complete chart types only. It is not possible to add features to existing chart types, and it is not possible to modify or extend parts of WebFOCUS outside of the chart area allocated to your extension.

This chapter summarizes the process of writing, configuring, and installing a chart extension. Detailed instructions can be found on the Information Builders GitHub site:

https://github.com/ibi/wf-extensions-chart

WebFOCUS extensions must be placed in the extensions folder under the web_resource folder of your WebFOCUS installation. By default, this is the following location:

`c:\ibi\`*`install_dir`*`\config\web_resource\extensions`

where:

install_dir

Is your WebFOCUS installation directory. For example, the default installation directory is WEBFOCUS82 for WebFOCUS 8.2.

Several sample chart extensions have already been installed in the extensions folder so that you can see their code, their structure, and how they are accessed in the WebFOCUS tools.

Note: The user installing the extension must know how to write JavaScript code for what the chart extension needs to generate. The GitHub site documents how to make the extension conform to the WebFOCUS API and how to install the extension in the WebFOCUS chart library. It does not describe how to write JavaScript code.

Creating a Chart Extension

This section summarizes the build cycle for creating an extension and the structure and components of an extension.

Reference: Build Cycle for Writing an Extension

Creating an extension often involves cycles of writing, running, and then debugging code.

When you make changes to the properties.js file for your extension, you need to clear the WebFOCUS cache in order for those changes to be recognized. Clear the cache using the *Clear cache* link in the WebFOCUS Administration Console.

If you change the .js code for your extension (for example, com.ibi.simple_bar.js), you do not need to make any changes to WebFOCUS. You only need to clear your own browser cache, to ensure that the new JavaScript file is downloaded. The same is true if you change any additional .js files included by your extension.

Reference: Extension Structure

The Simple Bar extension example demonstrates the required and optional files in an extension, and how those files are typically laid out.

You can open com.ibi.simple_bar and com.ibi.simple_bar.js in a text editor to see exactly how an extension is written.

The extension ID (ext_id) is a string in the form com.*your_company*.*extension_name*. The ext_id must be all lowercase, and can include only letters, numbers, underscores and dots. The entire extension lives in a folder named *ext_id*. The core of the extension lives in a file named *ext_id.js*. This file includes code to render the extension as a new chart type within WebFOCUS.

The properties.json file configures your extension to run in WebFOCUS. This file includes all the metadata needed to include your extension in the WebFOCUS user interface, as well as a list of all properties you wish to expose to end users, so they can customize the behavior of your extension.

The extension folder can also include optional additional folders for external css and lib resources. If your extension uses any additional CSS or JavaScript library files, you can keep those resources organized in dedicated folders, such as css and lib, as you choose. External resources are configured and loaded inside the base ext_id.js file of your extension.

Using the Chart Extension API

To see examples of everything that the chart extension API provides, look at *com.ibi.simple_bar.js*. It is divided into two main parts, chart rendering and extension configuration.

If you implement the extension API version 2, you can retrieve the field name and number format associated with a data bucket entry. To use API 2 and get a bucket entry field name or number format, an extension must declare that it implements extension API version 2 using the 'implements_api_version' entry in the info block of properties.json:

```
{
    "info": {
        "implements_api_version": "2.0"
            }
}
```

Rendering Charts

The extension API provides three entry points that you can use as needed by defining your own JavaScript callback functions. They are passed a set of properties in a config object. Some properties are available during the entire rendering process, and some are only available during render callback.

Reference: Chart Rendering Callback Functions

You can define the following three JavaScript callback functions. Only the renderCallback function is always required.

- ❑ **initCallback(successCallback, config)** This optional function is invoked by the engine exactly once during library load time, providing a way to implement document.onload initialization code. This function is passed a successCallback, which you must invoke with *true* if your initialization code succeeded or *false* if was not successful. If you call successCallback(false), no further interaction with your extension will occur, and your extension will render as an empty page.

- **preRenderCallback(config)** This optional function is invoked each time your extension is to be rendered, as the very first step in the overall rendering process. This is a good place to examine and tweak or override any internal chart properties that will affect the subsequent rendering.
- **renderCallback(config)** This required function must contain all of the code that will actually draw your chart. The *config* object will contain the properties described in the following sections.

Each of the three entry point callbacks is passed a config object, which contains a set of useful properties.

Example: Sample renderCallback Function

The following sample renderCallback code renders the Simple Bar extension.

```
function renderCallback(renderConfig) {
  var chart = renderConfig.moonbeamInstance;
  var props = renderConfig.properties;
  var container = d3.select(renderConfig.container)
   .attr('class', 'com_ibi_chart');
  var data = renderConfig.data;
  if (renderConfig.dataBuckets.depth === 1) {
   data = [data];
  }

  var seriesCount = data[0].length;
  var seriesLabels = data[0].map(function(el){return el.labels;});
  data = d3.transpose(data).map(function(el, idx) {
   el = el[0];
   var v = Array.isArray(el.value) ? el.value : [el.value];
   var y0 = 0;
   return v.map(function(d, s) {
    return chart.mergeObjects(d, {y0: y0, y1: y0 += d, seriesID: s, value:
d, labels: seriesLabels[idx]});
   });
  });

  var w = renderConfig.width;
  var h = renderConfig.height;
  var x =
d3.scale.ordinal().domain(pv.range(seriesCount)).rangeRoundBands([0, w],
0.2);
  var ymax = d3.max([].concat.apply([], data), function(d){return d.y1;});
  var y = d3.scale.linear().domain([0, ymax]).range([25, h]);
  var svg = container.selectAll("g")
   .data(data)
   .enter().append('g')
   .attr('transform', function(d, i){return 'translate(' + x(i) + ', 0)';});
  svg.selectAll("rect")
   .data(function(d){return d;})
   .enter().append('rect')
   .attr("width", x.rangeBand())
   .attr("y", function(d) {return h - y(d.y1);})
   .attr("height", function(d){return y(d.y1) - y(d.y0);})
   .attr('tdgtitle', function(d, s, g) {
    // To support tooltips, each chart object that should draw a tooltip
must
    // set its 'tdgtitle' attribute to the tooltip's content string.

    // Retrieve the chart engine's user-defined tooltip content with
getToolTipContent():
    // 's' and 'g' are the series and group IDs for the riser in question.
    // 'd' is this riser's individual datum, and seriesData is the array of
data for this riser's series.
    var seriesData = chart.data[s];
    var tooltip = renderConfig.modules.tooltip.getToolTipContent(s, g, d,
seriesData);
    // getToolTipContent() return values:
    //  - undefined: do not add any content to this riser's tooltip
    //  - the string 'auto': you must define some 'nice' automatic tooltip
content for this riser
    //  - anything else: use this directly as the tooltip content
    if (tooltip === 'auto') {
     if (d.hasOwnProperty('color')) {
      return 'Bar Size: ' + d.value + '<br />Bar Color: ' + d.color;
     }
     return 'Bar Size: ' + d.value;
    }
    return tooltip;
   })
   .attr('class', function(d, s, g) {
    // To support data selection and tooltips, each riser must include a
class name with the appropriate seriesID and groupID
    // Use chart.buildClassName to create an appropriate class name.
    // 1st argument must be 'riser', 2nd is seriesID, 3rd is groupID, 4th
is an optional extra string which can be used to identify the risers in
your extension.
    return chart.buildClassName('riser', s, g, 'bar');
   })
   .attr('fill', function(d) {
    // getSeriesAndGroupProperty(seriesID, groupID, property) is a handy
function
    // to easily look up any series dependent property.  'property' can be
in
    // dot notation (eg: 'marker.border.width').
    return chart.getSeriesAndGroupProperty(d.seriesID, null, 'color');
   });

  svg.append('text')
   .attr('transform', function(d) {return 'translate(' + (x.rangeBand() /
2) + ',' + (h - 5) + ')';})
   .text(function(d, i){return seriesLabels[i];})

  renderConfig.modules.tooltip.updateToolTips();  // Tell the chart engine
your chart is ready for tooltips to be added
  renderConfig.modules.dataSelection.activateSelection();  // Tell the
chart engine your chart is ready for data selection to be enabled
 }
```

Reference: Properties That Are Always Available

The following properties are always available.

Property Name	Description
moonbeamInstance	The chart instance currently being rendered.
data	The data set being rendered.
properties	The block of properties for your extension, as set by the user.
dataBuckets	Optional custom data buckets. For information, see *dataBuckets Block* on page 971.

Reference: Properties Available Only During Render Callback

The following properties are available only during render callback, and are used by your chart rendering code (renderCallback).

Property Name	**Description**
width	Width of the container your extension renders into, in pixels.
height	Height of the container your extension renders into, in pixels.
containerIDPrefix	The ID of the DOM container your extension renders into. Prepend this to *all* IDs your extension generates, to ensure multiple copies of your extension work on one page.
container	DOM node for your extension to render into, either an HTML DIV element or an SVG G element, depending on your chosen containerType extension configuration
rootContainer	DOM node containing the specific chart engine instance being rendered.

Configuring Your Chart Extension

Extension configuration consists of two parts.

- ❑ Chart Engine Configuration configures the extension to interact with the chart engine and chart canvas in WebFOCUS. This part of the extension configuration is defined in the config object that is passed to the chart renderer functions.
- ❑ Chart Interface Configuration interacts with the chart type picker in the user interface and the chart attribute categories. This part of the extension configuration is defined in the properties.json file.

Creating a config Object for Chart Engine Configuration

To configure your extension, create a *config* object with all the information unique to your extension, then register your extension with the extension API.

Reference: Creating a config Object for Your Extension

Required and optional properties in your config object are described in the following table.

Property Name	Description
id	Is the extension ID described in *Extension Structure* on page 960.
name	Is the name for the chart type to be displayed in the user interface.
description	Is a description for the chart type to be displayed in the user interface.
containerType	Is either 'html' or 'svg' (the default).
initCallback	Optional. References your initCallback function, described in *Chart Rendering Callback Functions* on page 961.
preRenderCallback	Optional. References your preRenderCallback function, described in *Chart Rendering Callback Functions* on page 961.
renderCallback	Required. References your renderCallback function, described in *Chart Rendering Callback Functions* on page 961.
resources	Optional. Are additional external resources (CSS and JS) required by this extension.

Example: Sample config Object

The following code is a sample of the config object used with the Simple Bar extension.

```
var config = {
    id: 'com.ibi.simple_bar',     // string that uniquely identifies this
extension
    containerType: 'svg',  // either 'html' or 'svg' (default)
    initCallback: initCallback,  // Refers to your init callback fn
(optional)
    preRenderCallback: preRenderCallback,  // Refers to your preRender
callback fn (optional)
    renderCallback: renderCallback,  // Refers to your render fn (required)
    resources:  {  // Additional external resources (CSS & JS) required by
this extension (optional)
        script: ['lib/d3.min.js'],
        css: ['css/extension.css']
    },
   }
```

Reference: Registering Your Extension

To register your extension with the WebFOCUS extension API, call:

```
tdgchart.extensionManager.register(config);
```

Reference: Tips for Building Your Extension

The easiest way to build your own extension is to clone the Simple Bar example, then tweak it. Assume the ID of the new extension is com.foo.bar:

1. Rename root folder to com.foo.bar. Rename com.ibi.simple_bar.js to com.foo.bar.js.
2. In com.foo.bar.js, delete the inner content of the three callback functions.
3. In com.foo.bar.js, change the entries for each property in config to match the requirements of your extension.
4. Add any external resources you need to *lib* and css, and load them by setting config.resources in com.foo.bar.js.
5. Implement renderCallback in com.foo.bar.js to draw your extension.

Configuring the Chart Interface

Each extension must include a properties.json file, which defines the information needed by WebFOCUS when drawing its user interface.

The properties.json file consists of the following blocks.

- ❑ **info block.** For information, see *info Block* on page 970.

- **properties block.** For information, see *properties Block* on page 970.
- **propertyAnnotations block.** For information, see *propertyAnnotations Block* on page 971.
- **dataBuckets block.** For information, see *dataBuckets Block* on page 971.
- **translations block.** For information, see *translations Block* on page 972.

Example: Sample properties.json File

The following properties.json file is from the Simple Bar extension.

```
{
    // Define some general extension configuration options
    "info": {
     "version": "1.0",  // version number of your extension.
     "implements_api_version": "2.0",  // version number of the WebFocus
    // API used by your extension.
      "author": "Information Builders",
      "copyright": "Information Builders Inc.",
      "url": "https://github.com/ibi/wf-extensions-chart/tree/master/simple_bar%20example",
      "icons": {
      "medium": "icons/medium.png"  // Reference to an image in the
    // extension, used in the WF chart picker
        }
    },
    // Define any properties of your extension that end user may want to change.
    "properties": {
        "exampleProperty": 50
    },
    // Define the possible values for each property in 'properties'.
    "propertyAnnotations": {
        "exampleProperty": "number"
    },
    // Define the available data buckets drawn in WF's 'Query' data bucket tree.
    "dataBuckets":  {
        // Choose whether or not to reuse existing WF data buckets.  All optional.
        "tooltip": false,
        "series_break": true,
        // Define your own custom data buckets.  Optional
        "buckets": [
            {
            "id": "value",
            "type": "measure",
            "count": {"min": 1, "max": 5}
           },
           {
           "id": "labels",
           "type": "dimension",
           "count": {"min": 1, "max": 5}
           }
           ]
          },
    // Define the set of labels used in the WF interface for buckets and
    // chart type picker.
    "translations": {
        "en": {
            "name": "My Simple Bar Chart",
            "description": "This chart is just a simple bar chart, nothing to see here.",
            "icon_tooltip": "This extension does ...",
            "value_name": "Value Bucket",
            "value_tooltip": "Drop a measure here",
            "labels_name": "Label Bucket",
            "labels_tooltip": "Drop a dimension here"
        },
        "fr": {
            "name": "Un Bar Chart tres simple",
            "description": "C'est un Bar Chart vraiment simple",
            "icon_tooltip": "This extension does ...",
            "value_name": "Value Bucket",
            "value_tooltip": "Drop a measure here",
            "labels_name": "Label Bucket",
            "labels_tooltip": "Drop a dimension here"
        }
    }
}
```

Reference: info Block

This block defines several general purpose configuration options, such as files containing chart icons.

icons

This property defines the files to be used as icons that represent the extension inside WebFOCUS tools. An extension can include multiple bitmap icons of different sizes, or a single scalable SVG icon. An SVG icon should not include hardcoded 'width' or 'height' attributes. The syntax for the info block with an icons object is:

```
{
    "info": {
        "icons": {
            "size": "icons/filename"
        }
    }
}
```

where:

size

Is the size for each icon file. Each icon file name is listed as an entry under 'icons' according to its size, which is one of:

- small, which describes an icon that is 32x32 pixels.
- medium, which describes a bitmap icon that is 72x72 pixels, or describes an SVG icon.

icons/filename

Is the name of the file containing the icon.

For example:

```
{
    "info": {
        "icons": {
            "medium": "icons/my_svg_icon.svg"
        }
    }
}
```

Reference: properties Block

This block defines any properties of your extension that the end user may want to change. The user can change these properties in the GRAPH_JS blocks in a WebFOCUS chart procedure.

Reference: propertyAnnotations Block

This block validates the content of the properties block. Everything in properties must appear in propertyAnnotations. The possible types of any non-object (leaf) property in properties must be notated as one of "str", "bool", or "number".

Reference: dataBuckets Block

This block defines the set of chart attribute categories that appear in the Query pane in the WebFOCUS user interface when creating a chart.

If you are using API version 2.0, dataBuckets.buckets is an array instead of an object. Each entry in this array represents the content of one data bucket. The *id* property identifies which bucket this is, and the *fields* array specifies how many entries are in this bucket and the unique information for each (titles, field names, number formats).

The dataBuckets object includes a method named getBucket(). Pass it the name of a bucket and it returns the content of that bucket.

There are two types of buckets, built-in and custom. Built-in buckets provide an easy way to reuse the existing WebFOCUS data bucket logic. There are currently two built-in buckets, tooltip, and series_break. Use any of these buckets by setting the associated dataBuckets property to *true*.

The dataBuckets block includes the following objects for API Version 1.0.

bucket. Each bucket block defines one custom chart attribute category. Each custom bucket requires the following properties:

- **id.** This property corresponds exactly to the dataArrayMap and data properties that will be received by the render function for your chart.
- **type.** This property defines the type of data field this bucket accepts, "measure", "dimension", or "both".
- **count.** Consists of count.min and count.max, which define the minimum and maximum number of fields this bucket can accept. A minimum of 0 means this bucket is optional.

The dataBuckets block includes the following objects for API Version 2.0.

- **depth.** Specifies the number of buckets in the buckets array.
- **buckets.** Specifies the properties of all fields for all buckets.
    - **id.** This property corresponds exactly to the dataArrayMap and data properties that will be received by the render function for your chart.

- **fields.** Is an array of fields for each bucket. For each field, defines the field title, name, and number format:

  ```
  {"title": "fieldtitle", "fieldName": "fieldname", "numberFormat":
  "format"}
  ```

With API Version 2.0, the content of renderConfig.dataBuckets that is passed to each render callback in an extension provides field title and format information. In the following example, the there are two buckets. The labels bucket has one field, CAR.ORIGIN.COUNTRY, whose title is COUNTRY. The value bucket has two fields, CAR.SALES, title SALES, and CAR.BODY.DEALER_COST, title DEALER_COST.

```
"dataBuckets": {
   "getBucket(bucketName)",
   "depth": 2,
 "buckets": [
  {
   "id": "labels",
   "fields": [
   {"title": "COUNTRY", "fieldName": "CAR.ORIGIN.COUNTRY"}
            ]
  },
  {
  "id": "value",
  "fields": [
   {"title": "SALES", "fieldName": "CAR.SALES",
    "numberFormat": "#,###.00"},
   {"title": "DEALER_COST",
    "fieldName": "CAR.BODY.DEALER_COST",
    "numberFormat": "#,###"}
         ]
      }
   ]
}
```

Reference: translations Block

This block defines translations in different languages for every label to be drawn in the WebFOCUS interface. The translation object has one property for each language the extension supports, keyed by ISO-639 two letter locale strings.

Accessing Data for Your Extension

Each time an extension is rendered, the render callback for the extension is passed the current data set using the renderConfig.data argument. The overall structure of the data set is defined by the set of buckets listed in the properties.json file, while the specific content of the data is defined by the data fields the user has added to each bucket.

Defining and Using Buckets in an Extension

The data set is passed into an extension using the data property of the first argument of the render callback, typically named renderConfig. Additional information about the current set of fields in each bucket is in renderConfig.dataBuckets.

A data set is represented in JavaScript as arrays of objects. If an extension defines only custom buckets, the data set will be a flat array of objects. If an extension uses some built-in buckets, the data set may contain deeply nested arrays of arrays. The renderConfig.dataBuckets.depth property will be set to the number of array dimensions in the current data set.

Custom Buckets

Each innermost object within the arrays of data (called a *datum*) will have one property for each data bucket that contains a field. Each property will be the id of a custom bucket, as defined in the dataBuckets.buckets section of properties.json. The type of values of these properties depend on the bucket type. Dimension buckets have string values, while measure buckets have numeric values. If a bucket contains more than one field, the associated property for each innermost object will be an array of string or number values.

Built-in Buckets

An extension can use buckets that are built-in and predefined by WebFOCUS. These buckets will affect more than just the data set. Each bucket will also set specific chart engine properties, to pass in additional information related to that bucket.

Each built in WebFOCUS bucket is either a *standard* bucket or a *break* bucket.

- Standard buckets behave exactly like custom buckets. The data set remains a single array, and each datum object will include an additional property named after the bucket.
- Break buckets divide the data set into additional arrays of data. For each break bucket used, each datum object will be transformed into a full array of datum objects. The number of datum objects in each array will remain unchanged, but the number of arrays or datum arrays will correspond to the number of entries in the break field.

Types of Break Buckets

Break buckets can be of two types:

- ❏ A series-break bucket breaks the data set into one array for each entry in the series break field chosen by the user. A series-break bucket uses series-dependent properties defined in the chart engine, and the data names are now listed in those series-dependent properties. Each entry in the series-break field will generate a corresponding series property object in the chart engine, retrievable with renderConfig.moonbeamInstance.getSeries(*x*), where *x* is an integer for the series to be retrieved. getSeries returns an object with properties such as color and label, which are unique to the chosen series.
- ❏ A matrix-break bucket is used for the sort fields that define the columns and rows in a matrix chart. A matrix-break bucket also adds more array dimensions to the data set. A matrix-break bucket is broken into *column* and *row* sub-buckets. If either the row or column bucket contains any fields, the data set will contain two additional dimensions of data, even if one of the matrix buckets is empty. That is, the data set will either contain neither row nor column data, or both row and column data, never just one or the other. bucket.depth will always be at least three.

The Tooltip Bucket

The tooltip bucket is not a break bucket, and does not add any additional array dimensions to the data set. Instead, tooltip behaves like a custom bucket. Each inner datum object will contain a property named *tooltip*, with a value of type string for dimensions, number for measures, and an array of values for multiple fields in the bucket.

The usefulness of this bucket is that in addition to including tooltip-specific data in the data set, WebFOCUS also generates meaningful tooltip content for each series. This tooltip content is the same content used for all of the built in WebFOCUS chart types. Using the tooltip bucket means the extension does not have to figure out what ought to go into each tooltip.

Example: Sample Series-Break Bucket Definition

This example uses the following sample data.

Car	**Country**	**Seats**
BMW	Germany	5
Audi	Germany	4
Peugeot	France	5

Car	Country	Seats
Alfa Romeo	Italy	4
Maserati	Italy	2
Toyota	Japan	4

The following code defines a series-break bucket.

```
dataBuckets:
        series_break: true,
        buckets: [
            {id: "label", type: "dimension"},
            {id: "value", type: "measure"}
        ]
```

Consider the following fields assigned to each of the buckets:

- "Country" assigned to the "series_break" bucket.
- "Car" assigned to the "label" bucket.
- "Seats" assigned to the "value" bucket.

In the renderConfig function, the renderConfig.data object will be similar to the following, in which the Country values are no longer part of the data array. However, a new array starts for each change in the Country value:

```
 [{labels: "PEUGEOT", value: 5}],
        [{labels: "ALFA ROMEO", value: 4}, {labels: "MASERATI", value: 2],
        [{labels: "TOYOTA", value: 4}],
        [{labels: "AUDI" ,value: 4}, {labels: "BMW", value: 5}]
```

The renderConfig.dataBuckets object will be defined as follows:

```
renderConfig.dataBuckets = {
        depth: 2,
        series_break: {title: "Country"},
        buckets: {
            label: {title: "Car"},
            value: {title: "Seats"}
```

Handling Partial and Null Data in an Extension

In many cases, the end user working with an extension cannot populate all of the extension buckets immediately. An extension must correctly handle these partial data cases, and cannot crash if one or more buckets are empty. It is important to check renderConfig.dataBuckets to see which buckets have been populated, and act accordingly.

In addition, data sets are often incomplete, missing some values for a given combination of dimensions and measures. These missing values may show up in the data set as null entries within an array (instead of datum objects), or they may show up as entirely empty arrays. It is important to detect and handle these missing data cases, and render a visualization appropriate for such missing data.

Most extensions require some minimum number of populated buckets before anything can be rendered. Use the count.min properties of each dataBuckets.bucket entry in properties.json to define these minimum requirements. If the fields in all buckets do not meet the minimum counts, then the renderCallback for the extension will not be called. Instead, the noDataPreRenderCallback for the extension is called. This allows the extension to render in a special *no data* mode. In this mode, the extension should render in grey scale, using renderCallback.baseColor as the main color. This should be a very simplified, sample rendering of the extension.

Example: Sample noDataPreRenderCallback Function

The following noDataPreRenderCallback function is from the Simple Bar sample extension.

```
function noDataRenderCallback(renderConfig) {
  var grey = renderConfig.baseColor;
  renderConfig.data = [{value: [3, 3]}, {value: [4, 4]}, {value: [5, 5]},
{value: [6, 6]}, {value: [7, 7]}];
  renderConfig.moonbeamInstance.getSeries(0).color = grey;
  renderConfig.moonbeamInstance.getSeries(1).color =
pv.color(grey).lighter(0.18).color;
  renderCallback(renderConfig);
 }
```

Installing a Chart Extension

1. Find the extensions folder for your local WebFOCUS installation. This is typically the following folder.

   ```
   C:\ibi\install_dir\config\web_resource\extensions
   ```

   where:

   install_dir

   Is your WebFOCUS installation directory.

   Note: The WebFOCUS Extension section of the Information Builders GitHub page maintains a list of publicly available and supported extensions. To install one of those, click the extension you want to install, then right click the zip file for that extension, for example *com.ibi.xyz.zip*, and choose *Save link as...*

2. Unzip the downloaded zip file into the WebFOCUS extensions folder. For example, for the *com.ibi.xyz.zip* zip file, this should create the following folder.

```
C:\ibi\install_dir\config\web_resource\extensions\com.ibi.xyz
```

If you are installing your own extension from your own environment, copy or download it to the WebFOCUS extensions folder, using the same naming conventions for the folder and the extension ID as described for the sample extensions.

3. Edit C:\ibi\install_dir\config\web_resource\extensions\html5chart_extensions.json. Create a new line for the new extension in the form:

   ```
   "com.ibi.abc": {"enabled": true},
   ```

   where:

   abc

   Is the name of the extension.

4. In the WebFOCUS Administration Console, click *Clear cache*. This will force WebFOCUS to reload all extensions.

Following is a sample html5chart_extensions.json.

```
{
        "com.ibi.simple_bar": {enabled: true},
        "com.ibi.liquid_gauge": {enabled: false},
        "com.ibi.sankey": {enabled: true}
    }
```

Note: The WebFOCUS Administration Console provides a user interface for installing chart extensions. For information, see Chapter 3, *Configuring the WebFOCUS Client*, in the *WebFOCUS Security and Administration* manual.

Reference: Preserving Custom Chart Types When Reinstalling the WebFOCUS Client

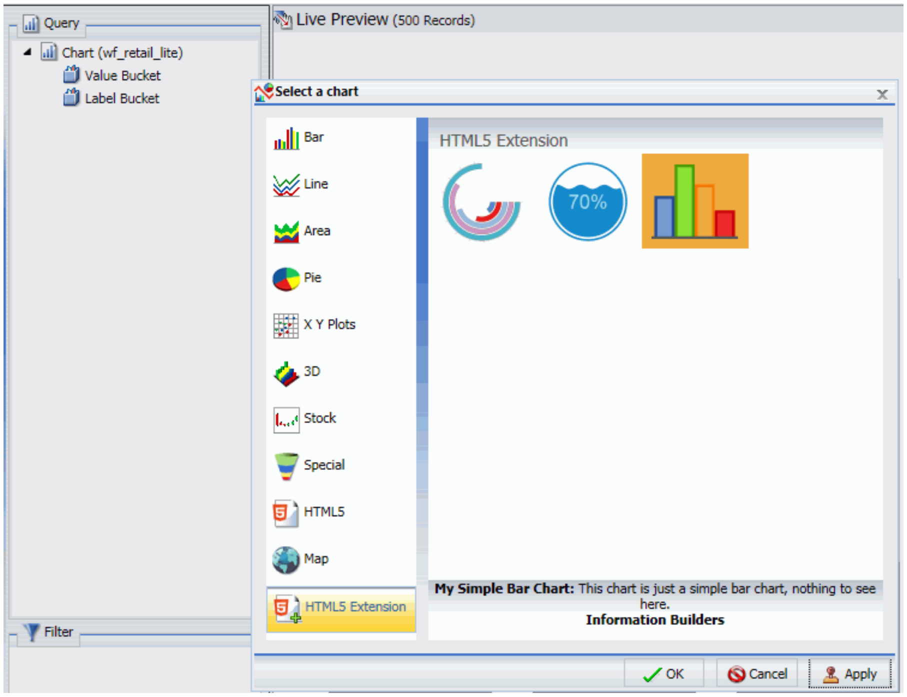
If you reinstall the WebFOCUS Client, your extensions folder will be overwritten. Therefore, if you have installed any custom chart extensions, you should preserve them by copying them to another location prior to reinstalling the WebFOCUS Client and copying them back to the extensions folder after reinstalling the WebFOCUS Client.

You will also have to copy the entries for your custom extensions into the new html5chart_extensions.json file installed with the new version of the WebFOCUS Client.

Note: The extensions that are delivered as part of WebFOCUS will be reinstalled automatically, so you should not preserve those extensions. In that way, if any enhancements have been made to those extensions, you will automatically have access to the enhanced versions when you reinstall the WebFOCUS Client.

Using Your Extension in a WebFOCUS Request

If you have installed and configured your extension as described, your extension will be available for use in the WebFOCUS tools as a chart type in the *Other* format category under *HTML5 Extension*, as shown in the following image.



My Simple Bar Chart: This chart is just a simple bar chart, nothing to see here.
Information Builders

The attribute categories you defined in the dataBuckets object of your extension are available in the query pane.

In the FOCEXEC:

- The LOOKGRAPH value is EXTENSION.
- The actual extension to use is identified in the chartType property of the *GRAPH_JS block in the StyleSheet. For example:

  ```
  *GRAPH_JS
  chartType: "com.ibi.simple_bar",
  }
  ```

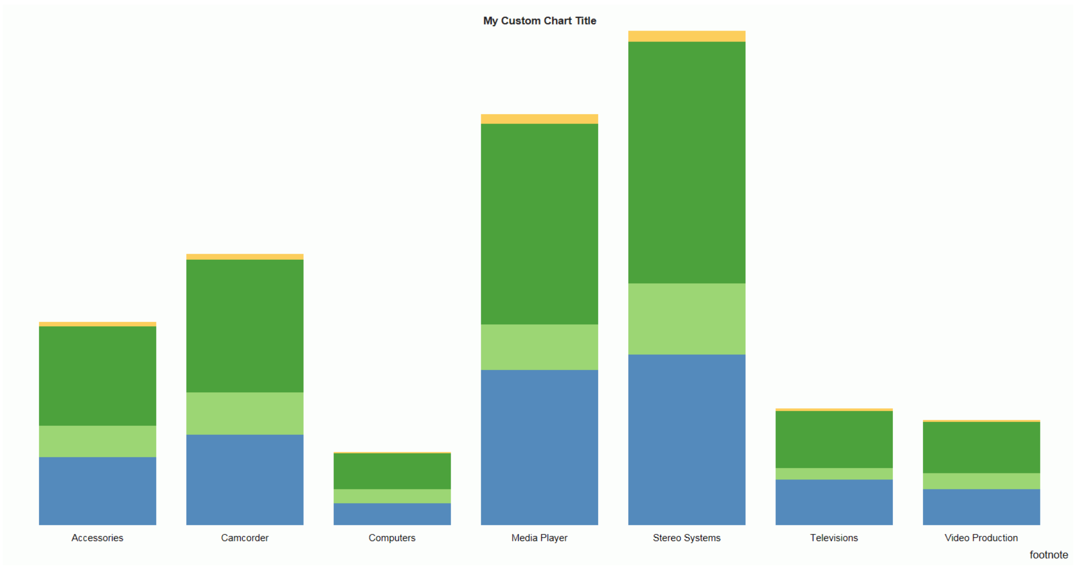
- Each custom attribute category name is prepended with a greater-than character (>). For example:

```
   TYPE=DATA, COLUMN=N1, BUCKET= >labels, $
   TYPE=DATA, COLUMN=N2, BUCKET= >value, $
   TYPE=DATA, COLUMN=N3, BUCKET= >value, $
   TYPE=DATA, COLUMN=N4, BUCKET= >value, $
   TYPE=DATA, COLUMN=N5, BUCKET= >value, $
```

The following is a sample request using the Simple Bar extension.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
GROSS_PROFIT_US
REVENUE_US
DISCOUNT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH EXTENSION
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/
ENWarm.sty,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET= >labels, $
TYPE=DATA, COLUMN=COGS_US, BUCKET= >value, $
TYPE=DATA, COLUMN=GROSS_PROFIT_US, BUCKET= >value, $
TYPE=DATA, COLUMN=REVENUE_US, BUCKET= >value, $
TYPE=DATA, COLUMN=DISCOUNT_US, BUCKET= >value, $
*GRAPH_JS
chartType: "com.ibi.simple_bar",
*END
ENDSTYLE
END
```

Run the chart. The output is shown in the following image.



Appendix A

WebFOCUS Chart Parameters

This topic describes WebFOCUS chart parameters and defaults.

In this appendix:

Controlling Chart Type Using the JSON chartType Property

This JSON property defines the chart type. However, it is recommended that you use the WebFOCUS LOOKGRAPH parameter to establish the chart type, when a corresponding LOOKGRAPH value exists. For information about the LOOKGRAPH parameter, see *Controlling Chart Type Using LOOKGRAPH* on page 40.

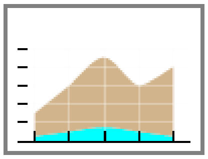
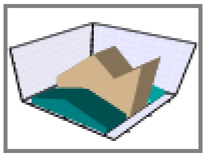
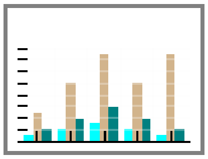
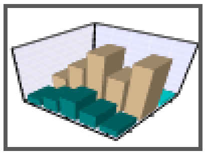
```
"chartType": "string"
```

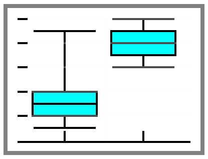
where:

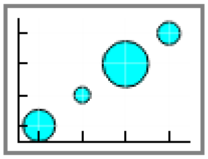
`"chartType": "string"`

Is a string enclosed in double quotation marks (") that defines the chart type. The default value is "bar".

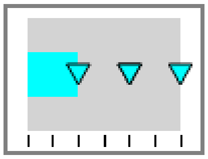
The valid chartType values, with illustrations of each chart type follow:

			
area	area3d	bar	bar3d

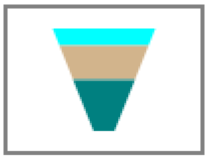

boxplot

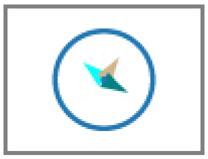

bubble

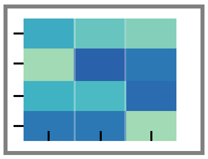
bubblemap

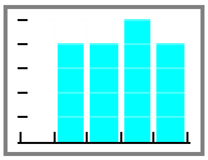

bullet

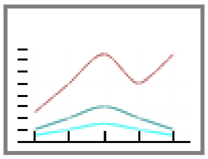
choropleth

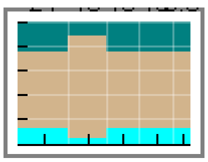

funnel

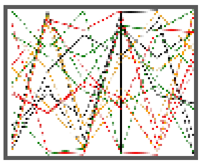

gauge

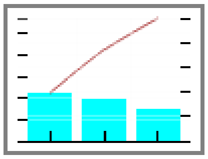

heatmap

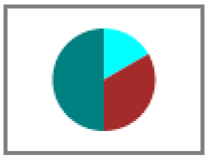

histogram

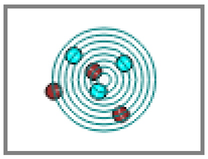

line

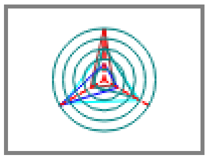

mekko

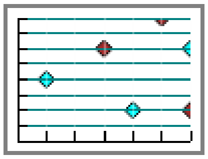

parabox


pareto

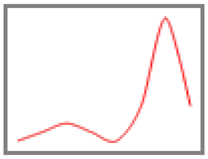

pie


polar

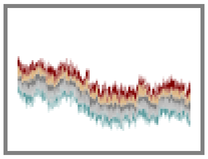

radar



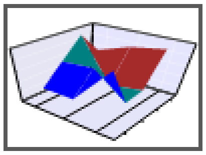
scatter



sparkline



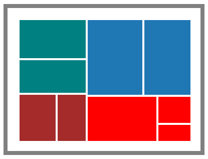
streamgraph



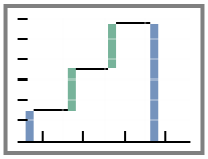
surface3d

Label4
Label6
Label3
Label1
Label5
Label2

tagcloud



treemap



waterfall

Creating Multiple Graphs

You can create multiple graphs by including secondary sort dimensions (fields).

By default, the number of graphs created depends on the number of values in the fields you designate in the sort (BY, ACROSS) phrases. You can change this default using the GRMERGE parameter:

- With GRMERGE OFF (the default), if a request contains two BY fields, there will be as many graphs as there are values in the first BY field. The second BY field will determine the x-axis. For example, if you have selected a BY field with two values, two graphs will be generated. If you have selected a field with ten values, ten graphs will be generated. If there is one BY phrase and one ACROSS phrase, as many graphs will display as there are values in the BY field. The ACROSS field will determine the x-axis. You can select the second horizontal category by including multiple BY phrases or an ACROSS and BY phrase in the same request.
- With GRMERGE ON, WebFOCUS creates one merged graph.
- With GRMERGE ADVANCED, WebFOCUS uses three parameters to determine:
    - How many graphs to generate (GRMULTIGRAPH).
    - How many sort fields should be placed on the graph legend (GRLEGEND).
    - How many sort fields should be placed on the x-axis (GRXAXIS).

Multiple graphs can be displayed in either merged format or in columns. For details, see *How to Merge Multiple Graphs* on page 984 and *How to Display Multiple Graphs in Columns* on page 987.

Syntax: How to Merge Multiple Graphs

```
SET GRMERGE={ON|OFF|ADVANCED}
```

where:

`ON`

Turns on the merge graph option.

`OFF`

Turns off the merge graph option. This is the default.

`ADVANCED`

Turns on the advanced merge option. This option uses three parameters to determine how to merge the graphs:

- GRMULTIGRAPH, which specifies how many sort fields to use to create multiple graphs.
- GRLEGEND, which specifies how many sort fields to place on the graph legend.
- GRXAXIS, which specifies how many sort fields to display on the x-axis. GRXAXIS must be at least 1 in order to plot the graph. A value greater than one creates nested X-axes.

Note: The sum of the sort fields used by GRMULTIGRAPH, GRLEGEND, and GRXAXIS must equal the number of sort fields in the graph request.

The syntax for the GRMULTIGRAPH, GRLEGEND, and GRXAXIS parameters is:

`ON GRAPH SET GRMULTIGRAPH` *n*

Specifies how many sort fields (0 through 2) to use to break the output into multiple graphs. The outermost sort fields are used to separate the graphs. When *n* is greater than zero, this is similar to GRMERGE=OFF, but allows an additional sort field.

`ON GRAPH SET GRLEGEND` *n*

Specifies how many of the remaining outermost sort fields (0 through 2), after the ones used for GRMULTIGRAPH, to add to the graph legend. When *n* is greater than zero, this is similar to GRMERGE=ON, but allows an additional sort field.

`ON GRAPH SET GRXAXIS` *`n`*

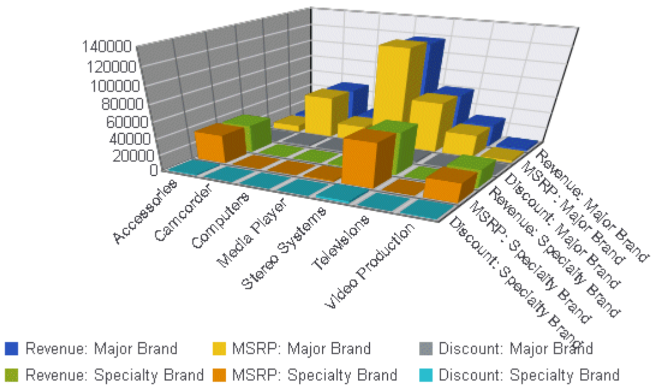
Specifies how many of the remaining sort fields (1 through 3) to display on the x-axis. When *n* is greater than 1, this creates nested x-axes.

Example: Merging Multiple Graphs With GRMERGE ON

The following illustrates a graph with two horizontal, or x-axes, categories (PRODUCT_CATEGORY and BRANDTYPE) that have been merged.

```
SET GRMERGE=ON
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US MSRP_US DISCOUNT_US
ACROSS PRODUCT_CATEGORY
BY BRANDTYPE
ON GRAPH SET LOOKGRAPH 3D_BAR
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```

The output is:

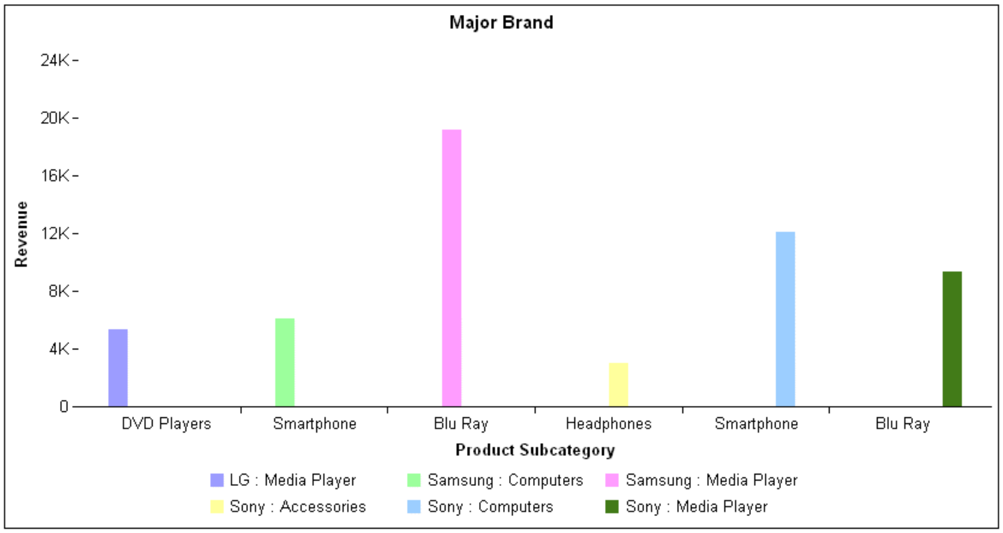


Example: Merging Multiple Graphs With GRMERGE ADVANCED

The following example generates a vertical bar graph that separates the outermost sort field (BRANDTYPE) onto separate graphs, distinguishes the next two sort fields (BRAND and PRODUCT_CATEGORY) by combining them on the graph legend, and places the PRODUCT_SUBCATEG sort field on the x-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM  REVENUE_US
BY BRANDTYPE BY BRAND BY PRODUCT_CATEGORY BY PRODUCT_SUBCATEG
WHERE BRAND EQ 'Sony' OR 'Samsung' OR 'LG'
WHERE PRODUCT_SUBCATEG EQ 'Blu Ray' OR 'Smartphone' OR 'DVD Players' OR
'Headphones'
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 2
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH PCHOLD FORMAT JSCHART
END
```

The first chart is for region Major Brand. The legend distinguishes Brand-Category combinations by color, and the Subcategory sort field is repeated on the x-axis for each Brand-Category combination:



Syntax: How to Display Multiple Graphs in Columns

```
SET GRWIDTH=nn
```

where:

nn

Is the number of columns in which to display multiple graphs. This may be any value from 0-512. The default is 0.

All values from 1-512 will display graphs in an HTML table with the corresponding number of columns. The default value of 0 will display the graphs one under the other in a Java applet.

Customizing Graphs Using SET Parameters

The GRAPH environment includes a set of parameters that control the appearance of the graph and offer some additional control when you run the request.

For example, the BSTACK parameter enables you to specify that the bars on a bar graph are to be stacked, rather than placed side by side.

Syntax: How to Use SET Parameters With GRAPH Requests

To set the parameters that control the GRAPH environment, use the appropriate variation of the SET parameter.

```
SET parameter=value,parameter=value...
```

For a list of supported GRAPH parameters, see *Values and Functions of SET Parameters for Graphs* on page 988.

Note:

- Repeat the command SET on each new line.
- When entering more than one parameter on a line, separate them with commas (,).
- You can use unique truncations of parameter names. You must make sure that they are unique.

Example: Using SET Parameters With GRAPH Requests

The following shows how to set the height (y-axis) and width (x-axis) for a graph.

```
SET HAXIS=75,VAXIS=40
GRAPH FILE filename
.
.
.
END
```

Reference: Values and Functions of SET Parameters for Graphs

Graph SET Parameter	Values	Parameter Function
3D	<u>ON</u>|OFF	When set to ON, a three-dimensional chart is produced. When set to OFF, a two-dimensional chart is produced. ON is the default.
AUTOTICK	<u>ON</u>|OFF	When set to ON, tick mark intervals are automatically set. ON is the default. (See also HTICK and VTICK.)
BARNUMB	ON|<u>OFF</u>	When set to ON, places the summary values at the ends of the bars on bar graphs or slices on pie graphs. OFF is the default.
BSTACK	ON|<u>OFF</u>	When set to ON, specifies that the bars on a bar graph are to be stacked rather than placed side by side. OFF is the default.

Graph SET Parameter	Values	Parameter Function
EMBEDHEADING	ON|OFF	When set to ON, embeds the heading in the chart. When set to OFF, places the heading above the chart on the HTML page. OFF is the default.
GRAPHDEFAULT	ON|OFF	When set to ON, WebFOCUS default styles are used. When set to OFF, defaults are taken from the legacy graph API. ON is the default value.
GRAPHEDIT	*graphedit*	As of WebFOCUS 8.0, this parameter has been deprecated. For information about editing charts, see the *WebFOCUS InfoAssist User's Manual*.
GRID	ON|OFF	When set to ON, specifies that a grid is to be drawn on the graph at the horizontal and vertical class marks (see also VGRID). OFF is the default.
HAUTO	ON|OFF	When set to ON, specifies automatic scaling of the horizontal axis unless overridden by the user. If set to OFF, user must supply values for HMAX and HMIN. ON is the default.

Graph SET Parameter	Values	Parameter Function
HAXIS		Specifies the width in characters of the horizontal axis. This parameter can be adjusted for graphs generated offline. HAXIS is ignored for online displays since the width of the graph is automatically adjusted to the width of the display area.
HCLASS	*nnn*	Specifies the horizontal interval mark when AUTOTICK=OFF (see also HTICK).
HMAX	*nnn*	Specifies the maximum value on the horizontal axis when the automatic scaling is not used (HAUTO=OFF).
HMIN	*nnn*	Specifies the minimum value on the horizontal axis when the automatic scaling is not used (HAUTO=OFF).
HSTACK	ON|OFF	When set to ON, specifies that the bars on a histogram are to be stacked, rather than placed side by side. OFF is the default.
HTICK	*nnn*	Specifies the horizontal axis tick mark interval when AUTOTICK is OFF (see also HCLASS).
LOOKGRAPH	*option*	Specifies the graph type. For more information, see *Controlling Chart Type Using LOOKGRAPH* on page 40.

Graph SET Parameter	Values	Parameter Function
VAUTO	ON|OFF	When set to ON, specifies automatic scaling of the vertical axis unless overridden by the user. If set to OFF, the user must supply values for VMAX and VMIN. ON is the default.
VAXIS		Specifies page length in lines. This parameter can be adjusted for graphs generated offline. VAXIS is ignored for online displays since the height of the graph is automatically adjusted to the display area.
VCLASS	*nnn*	Specifies the vertical interval mark when AUTOTICK=OFF (see also VTICK).
VGRID	ON|OFF	When set to ON, specifies that a grid is to be drawn on the graph at the horizontal and vertical class marks (see also GRID). OFF is the default.
VMAX	*nnn*	Specifies the maximum value on the vertical axis when the automatic scaling is not used (VAUTO=OFF).
VMIN	*nnn*	Specifies the minimum value on the vertical axis when the automatic scaling is not used (VAUTO=OFF).
VTICK	*nnn*	Specifies the vertical axis tick mark interval when AUTOTICK is OFF (see also VCLASS).

Graph SET Parameter	Values	Parameter Function
`VZERO`	`ON|OFF`	Determines whether missing values along the y-axis are stored or ignored. If set to ON, missing data along the y-axis is treated as zero. If set to OFF, missing data along the y-axis is ignored and values are not stored in the plot matrix. OFF is the default.

Appendix B

Converting Requests to Chart Attribute Syntax

This topic describes how to convert traditional WebFOCUS chart requests to use chart attribute syntax.

In this appendix:

Overview of Conversion Rules

Traditional WebFOCUS chart syntax supports both BY and ACROSS sort phrases. Chart attribute syntax supports only BY sort phrases, so any ACROSS phrase in the request must be changed to BY. This does not affect the chart output in any way.

In addition, if the traditional chart request does not generate an HTML5 chart, you must change the chart output format to JSCHART. For example, suppose the request generates chart output as format PNG:

```
ON GRAPH PCHOLD FORMAT PNG
```

You must change the format to generate an HTML5 chart using the following syntax:

```
ON GRAPH PCHOLD FORMAT JSCHART
```

By default, multiple sort fields in a request generate multiple charts. With traditional WebFOCUS chart syntax, multiple charts can be combined using several WebFOCUS parameters to specify how to assign sort fields to chart components. The GRMERGE parameter can activate simple or advanced merging. With GRMERGE ON, one chart is generated. With GRMERGE ADVANCED, the following parameters are used to determine how many charts are generated:

- GRMULTIGRAPH, which specifies how many sort fields to use to create multiple graphs.
- GRLEGEND, which specifies how many sort fields to place on the graph legend.
- GRXAXIS, which specifies how many sort fields to display on the x-axis. A value greater than 1 generates nested x-axes, except in the case of a bubblemap based on latitude and longitude sort fields. You can control whether nested or concatenated x-axis labels are generated by specifying a concatenation symbol for the x-axis labels, as described in *Axis Properties* on page 325.

The number of sort fields in the request must be the same as the combined number of merge fields. A few chart types do not support merge parameters. With some chart types, a specific merge parameter may not be supported, and with some chart types GRLEGEND and GRXAXIS are interchangeable and are only needed to make the number of sort fields equal to the number of merge fields. These instances are described in the following sections.

Chart attribute syntax eliminates the need for merge parameters. Instead, additional sort fields in a request can be assigned to attribute categories that control how many charts are generated and how they are merged. These assignments are described in the following sections. For more information about chart attribute syntax, see *WebFOCUS Chart Attribute Syntax* on page 143.

Note: Some of the chart requests in this chapter generate more merge parameters than the WebFOCUS tools generate. For example, a request may set GRLEGEND to 2, when the tools only allow 1 GRLEGEND field. The purpose of this difference is to illustrate parameter conversions to chart attribute categories, not actual chart requests generated by the tools.

Converting Bar, Line, and Area Chart Requests to Chart Attribute Syntax

Traditional bar, line, and area charts have separate LOOKGRAPH parameter values for vertical and horizontal orientations and for side-by-side, stacked, absolute, and percent layouts.

With chart attribute syntax, one LOOKGRAPH value is used for all layouts and orientations of the same type of chart, and the layout and orientation are specified in the TYPE=REPORT declaration of the WebFOCUS StyleSheet.

For bar, line, and area charts, you must convert the LOOKGRAPH parameter to the correct value, make sure the orientation and layout are assigned correctly in the WebFOCUS StyleSheet, and assign the measures and sort fields to the correct attribute categories.

Note that chart attribute syntax clusters similar x-axis values together, so the output looks different from the output generated by the GRMERGE syntax, in which the combinations of sort field values on the x-axis are not clustered.

Reference: LOOKGRAPH Conversions for Bar Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with the new CHART-ORIENTATION and CHART-SERIES-LAYOUT values.

LOOKGRAPH Parameter Values		**TYPE=REPORT StyleSheet Attributes**	
HBAR	BAR	horizontal	side-by-side (default)
VBAR, BAR	BAR	vertical (default)	side-by-side (default)
HBRSTK1	BAR	horizontal	stacked
VBRSTK1, STACK	BAR	vertical (default)	stacked
HBRSTKPC	BAR	horizontal	percent
VBRSTKPC	BAR	vertical (default)	percent

Reference: LOOKGRAPH Conversions for Line Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with the new CHART-ORIENTATION and CHART-SERIES-LAYOUT values.

LOOKGRAPH Parameter Values		**TYPE=REPORT StyleSheet Attributes**	
HLINE	LINE	horizontal	absolute (default)
VLINE, LINE	LINE	vertical (default)	absolute (default)
HLINSTK	LINE	horizontal	stacked
VLINSTK	LINE	vertical (default)	stacked

LOOKGRAPH Parameter Values		TYPE=REPORT StyleSheet Attributes	
HLNSTKPC	LINE	horizontal	percent
VLNSTKPC	LINE	vertical (default)	percent

Reference: LOOKGRAPH Conversions for Area Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with the new CHART-ORIENTATION and CHART-SERIES-LAYOUT values.

LOOKGRAPH Parameter Values		TYPE=REPORT StyleSheet Attributes	
HAREA	AREA	horizontal	absolute (default)
VAREA	AREA	vertical (default)	absolute (default)
HAREASTK	AREA	horizontal	stacked
VAREATK	AREA	vertical (default)	stacked
HARESTKP	AREA	horizontal	percent
VARESTKP	AREA	vertical (default)	percent

Reference: Attribute Category Assignments for Bar, Line, and Area Charts

The following table lists the attribute category conversions for bar, line, and area charts.

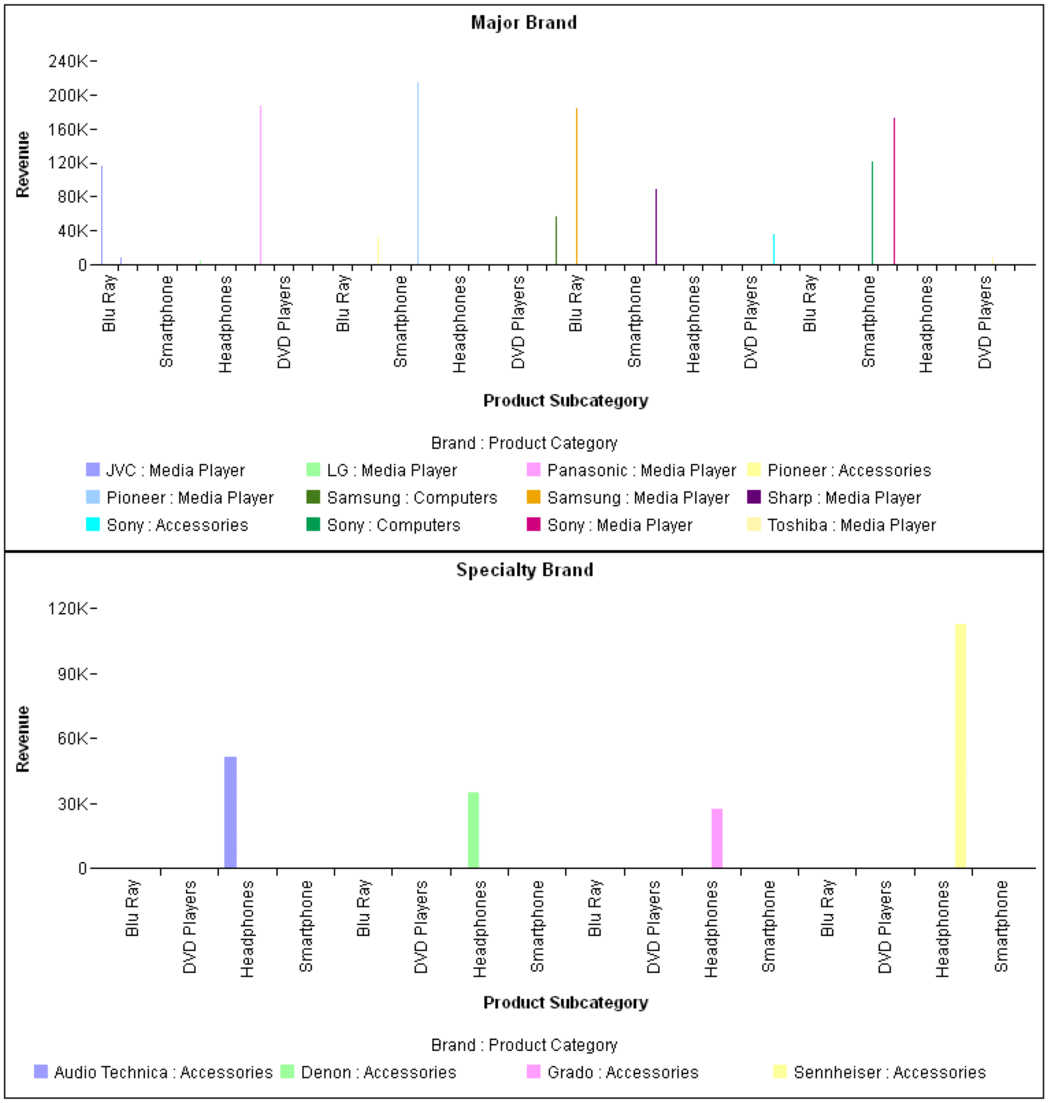
Type of Column or Parameter	Attribute Category
measure field	y-axis
GRXAXIS sort field	x-axis
GRLEGEND sort field	color
GRMULTIGRAPH sort field	page

Example: Converting a Bar Chart Request to Chart Attribute Syntax

The following example generates a vertical bar chart that separates the outermost sort field (BRANDTYPE) onto separate charts, distinguishes the next two sort fields (BRAND and PRODUCT_CATEGORY) by combining them on the graph legend, and places the PRODUCT_SUBCATEG sort field on the x-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY BRANDTYPE
BY BRAND
BY PRODUCT_CATEGORY
ACROSS PRODUCT_SUBCATEG
WHERE PRODUCT_SUBCATEG EQ 'Blu Ray' OR 'Smartphone' OR 'DVD Players' OR
'Headphones'
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 2
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH PCHOLD FORMAT JSCHART
END
```

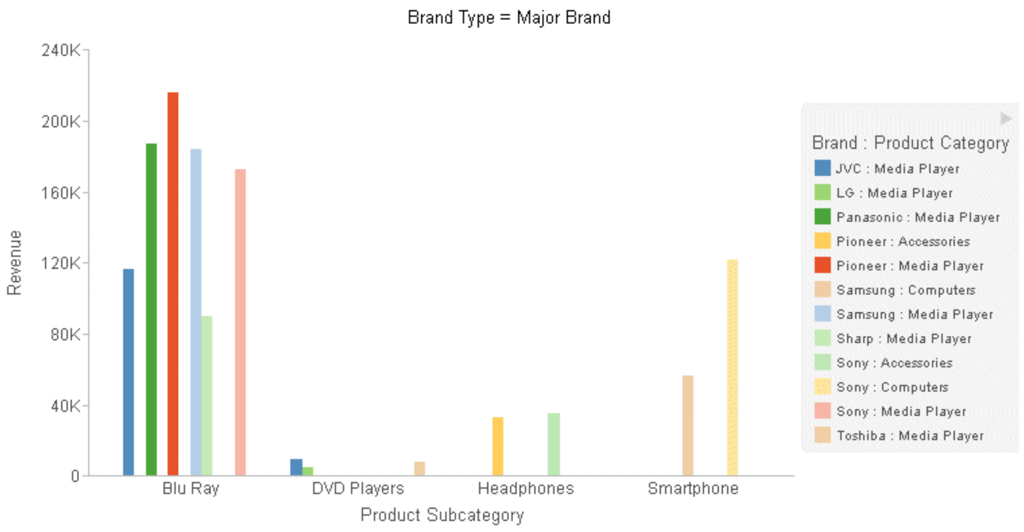
The output is shown in the following image:

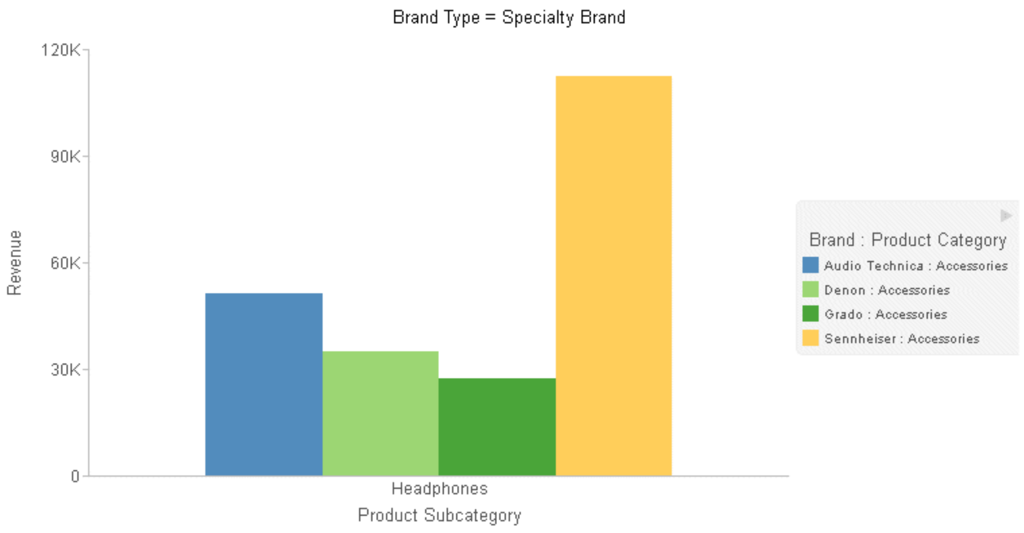


The following is the same request converted to chart attribute syntax. The BRANDTYPE sort field is assigned to the page category, the BRAND and PRODUCT_CATEGORY sort fields are assigned to the color category, the PRODUCT_SUBCATEG sort field is assigned to the x-axis category, and the REVENUE_US measure is assigned to the y-axis category. The ACROSS phrase is changed to a BY phrase, and the chart type is BAR:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"Brand Type = <BRANDTYPE"
SUM REVENUE_US
BY BRANDTYPE
BY BRAND
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
WHERE PRODUCT_SUBCATEG EQ 'Blu Ray' OR 'Smartphone' OR 'DVD Players' OR
'Headphones'
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=page,$
TYPE=DATA, COLUMN=BRAND, BUCKET=COLOR,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color,$
TYPE=DATA, COLUMN=PRODUCT_SUBCATEG, BUCKET=x-axis,$
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis,$
END
```

The output shown in the following image looks different from the GRMERGE output, since chart attribute syntax clusters similar x-axis values together. For example, the media players from all of the brands are displayed as one cluster of bars:



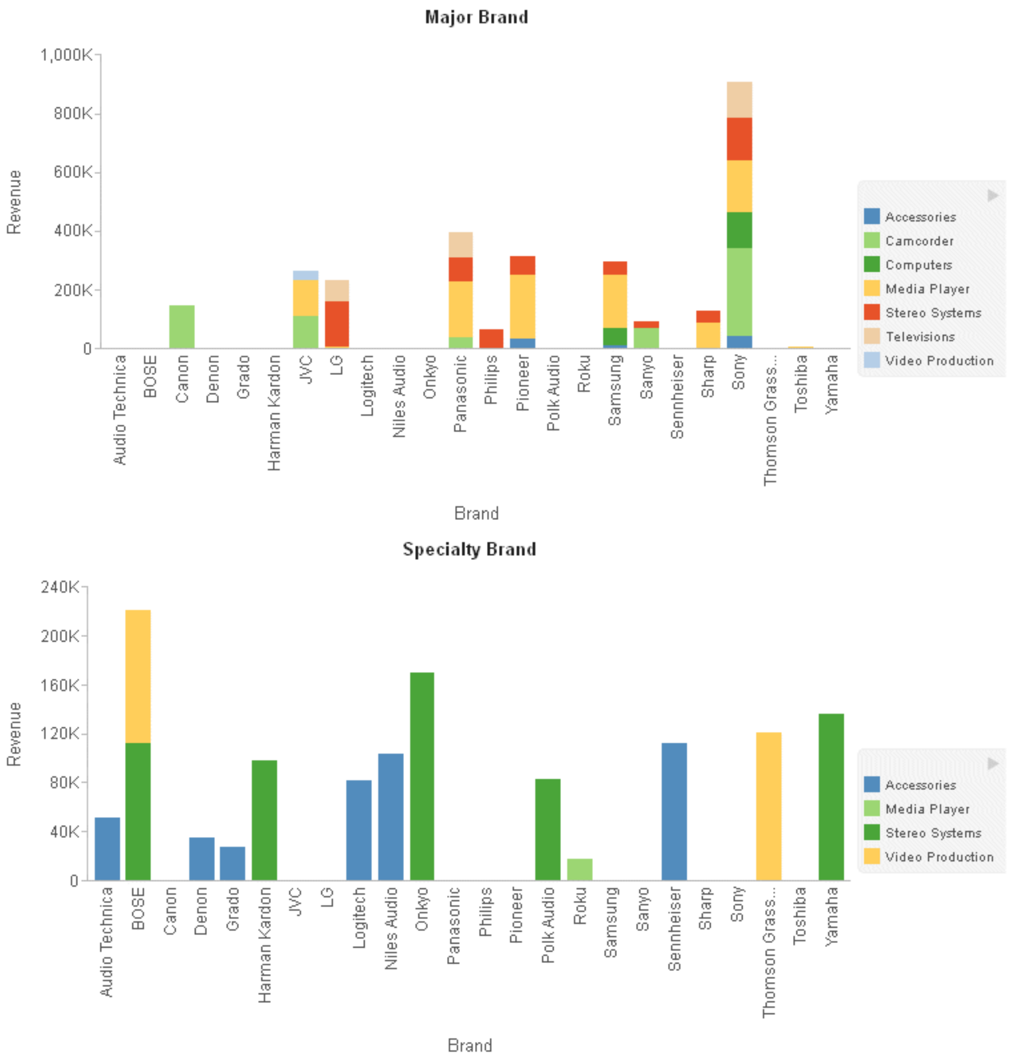


Example: Converting a Stacked Bar Chart Request to Chart Attribute Syntax

The following request generates multiple stacked bar charts. A new chart is generated for each value of BRANDTYPE (the GRMULTIGRAPH field), the field PRODUCT_CATEGORY is the GRLEGEND field, and the field BRAND is the GRXAXIS field:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ACROSS BRAND
ON GRAPH SET AUTOFIT ON
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH VBRSTK1
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

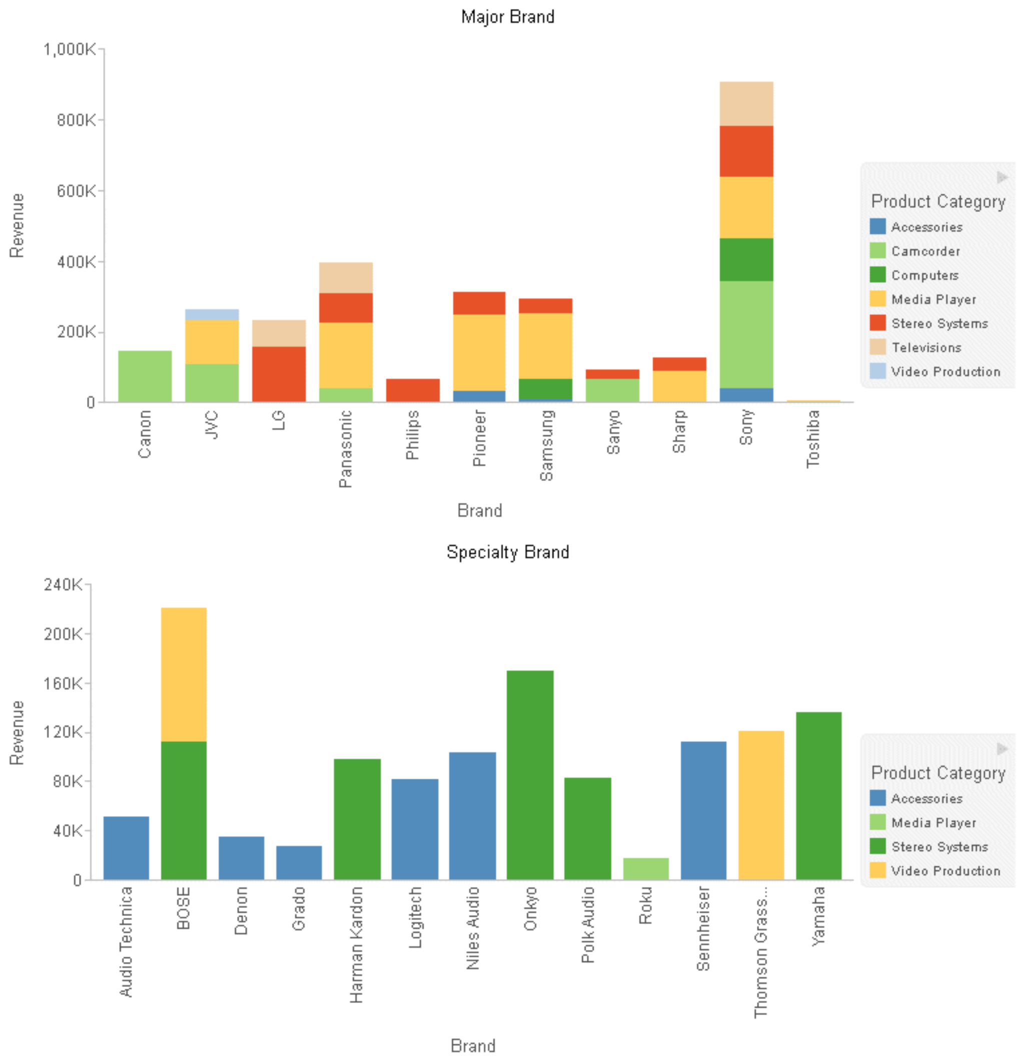
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The CHART-SERIES-LAYOUT is stacked. The measure REVENUE_US is assigned to the y-axis category, the BRANDTYPE sort field is assigned to the page category, the PRODUCT_CATEGORY sort field is assigned to the color category, and the BRAND sort field is assigned to the x-axis category. The ACROSS sort phrase from the original request is changed to a BY sort phrase. The chart type is BAR:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM REVENUE_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
BY BRAND
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET AUTOFIT ON
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=report, chart-series-layout=stacked,$
type=data, column=brandtype, bucket=page,$
type=data, column=brand, bucket=x-axis,$
type=data, column=product_category, bucket=color,$
type=data, column=revenue_us, bucket=y-axis,$
END
```

The output is shown in the following image:



Converting Boxplot Requests to Chart Attribute Syntax

A boxplot shows data broken into quartiles. In a boxplot, each series and group requires five values in order to visualize this breakdown.

With traditional syntax, the measures need to be in ascending order so that they can be assigned to the correct quartile. With chart attribute syntax, the category assignments make this requirement not necessary.

Reference: LOOKGRAPH Conversions for Boxplots

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with additional properties that may be needed for the chart type.

LOOKGRAPH Parameter	
BOXPLOT	BOXPLOT

Reference: Attribute Category Assignments for Boxplots

The following table lists the attribute category conversions for boxplots.

Type of Column or Parameter	**Attribute Category**
low value measure field	min
box bottom measure field	lower
median measure field	median
box top measure field	upper
high value measure field	max
sort field	x-axis

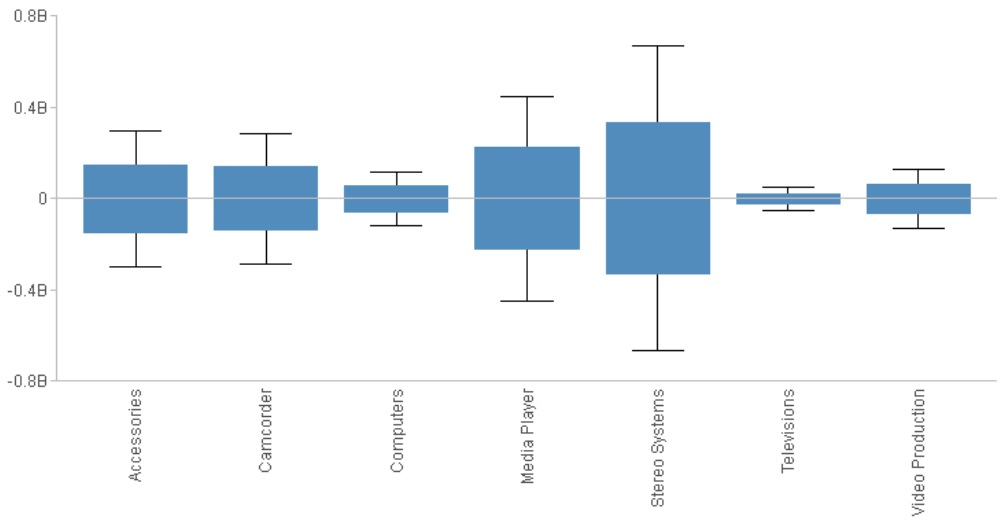
Note: Merge parameters are not supported with boxplots.

Example: Converting a Boxplot Request to Chart Attribute Syntax

The following request generates a boxplot.

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US -100000;
DIFF2 = COGS_US -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"boxplotProperties": {"drawHatAsBox": false}
*END
ENDSTYLE
END
```

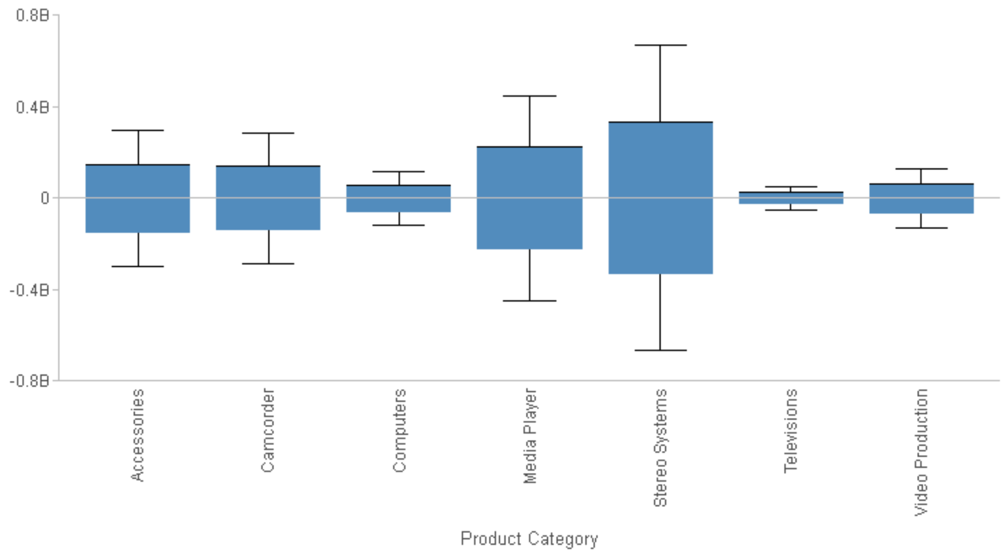
The output is shown in the following image.



The following version of the request uses chart attribute syntax.

```
DEFINE FILE WF_RETAIL_LITE
DIFF1 = COGS_US -100000;
DIFF2 = COGS_US -200000;
DIFF3 = COGS_US +100000;
DIFF4 = COGS_US +200000;
END
GRAPH FILE WF_RETAIL_LITE
SUM DIFF1 DIFF2 MDN.COGS_US DIFF3 DIFF4
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BOXPLOT
ON GRAPH SET STYLE *
type=data, column=product_category, bucket=x-axis,$
type=data, column=diff2, bucket=min,$
type=data, column=diff1, bucket=lower,$
type=data, column=C4, bucket=median,$
type=data, column=diff3, bucket=upper,$
type=data, column=diff4, bucket=max,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"boxplotProperties": {"drawHatAsBox": false}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Converting Bubble Chart Requests to Chart Attribute Syntax

Bubble charts are scatter charts in which the size of each marker varies depending on the value of a measure.

Reference: LOOKGRAPH Conversions for Bubble Charts

The following table lists the traditional LOOKGRAPH value and the new LOOKGRAPH value.

LOOKGRAPH Parameter	
BUBBLE	BUBBLE

Reference: Attribute Category Assignments for Bubble Charts

The following table lists the attribute category conversions for bubble charts.

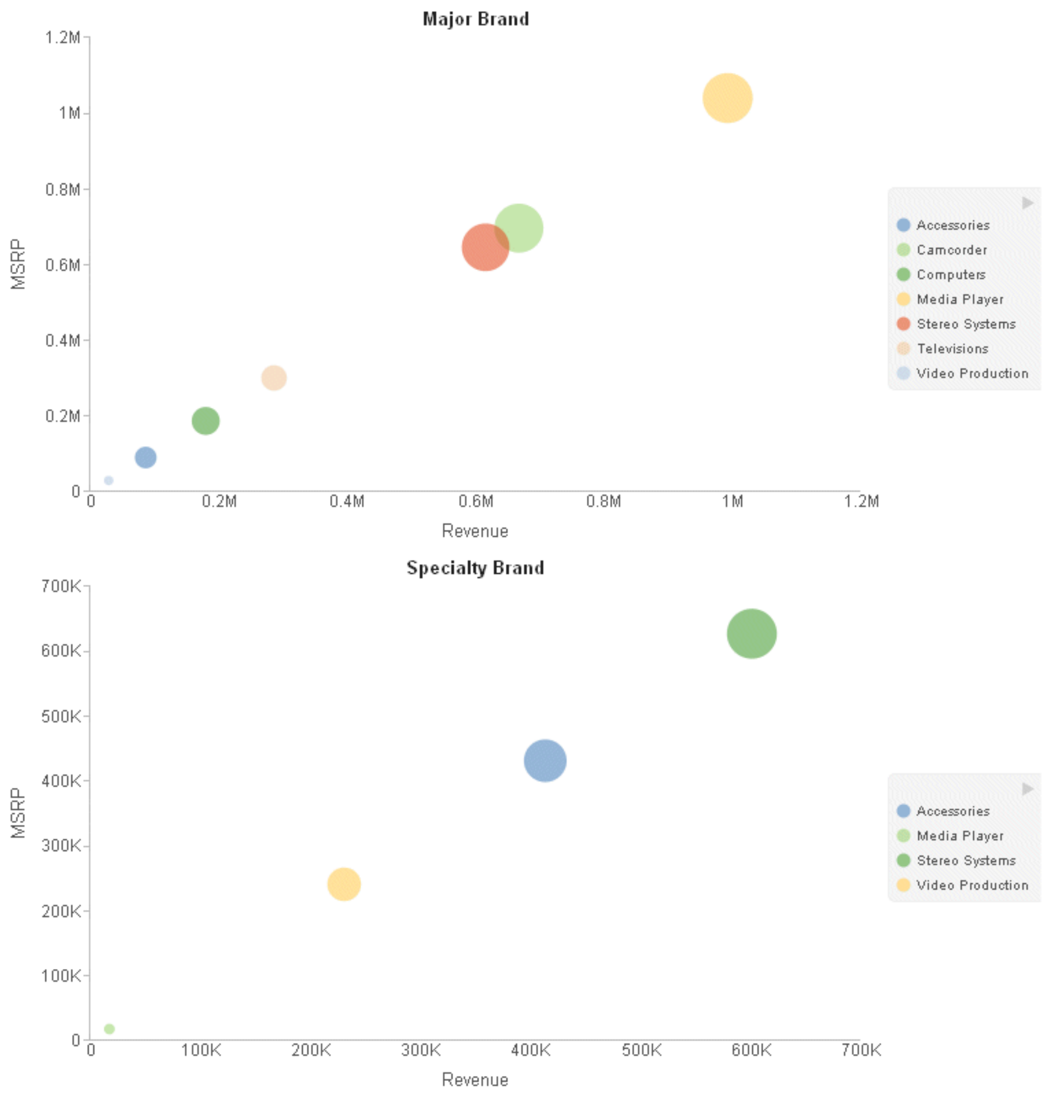
Type of Column or Parameter	**Attribute Category**
measure field	y-axis
measure field	x-axis
measure field	size
GRLEGEND or GRXAXIS sort field	color
GRMULTIGRAPH sort field	page

Example: Converting a Bubble Chart Request to Chart Attribute Syntax

The following example generates multiple bubble charts using traditional syntax (LOOKGRAPH value is BUBBLE). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate bubble charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY sort field is the GRLEGEND sort field. By default, the legend is not visible, so the JSON visible:true property is set for the legend:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
MSRP_US
GROSS_PROFIT_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"legend": {"visible": true}
*END
ENDSTYLE
END
```

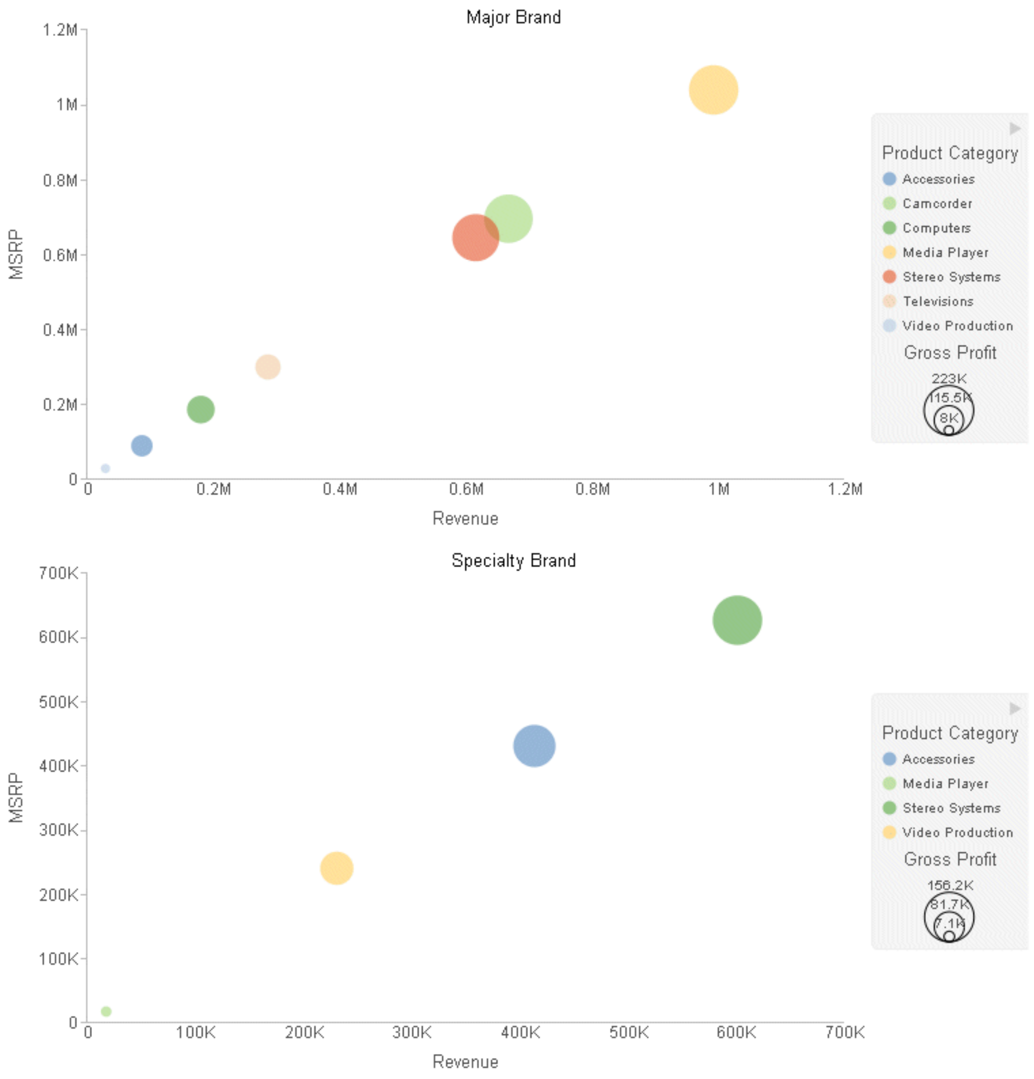
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is BUBBLE. The measures (REVENUE_US, MSRP_US, and GROSS_PROFIT_US) are assigned to the x-axis, y-axis, and size attribute categories, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (PRODUCT_CATEGORY) is assigned to the color attribute category.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
SUM REVENUE_US
MSRP_US
GROSS_PROFIT_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET LOOKGRAPH BUBBLE
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN = BRANDTYPE, BUCKET=PAGE,$
TYPE=DATA, COLUMN = PRODUCT_CATEGORY, BUCKET=COLOR,$
TYPE=DATA, COLUMN = REVENUE_US, BUCKET=x-axis,$
TYPE=DATA, COLUMN = MSRP_US, BUCKET=y-axis,$
TYPE=DATA, COLUMN = GROSS_PROFIT_US, BUCKET=SIZE,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
END
```

The output is shown in the following image:



Converting Funnel Chart Requests to Chart Attribute Syntax

Funnel charts slice a measure based on a sort field. The defaults for several properties vary depending on whether the request uses traditional WebFOCUS syntax, chart attribute syntax, or a StyleSheet.

Reference: LOOKGRAPH Conversions for Funnel Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with additional properties that may be needed for the chart type.

LOOKGRAPH Parameter	
FUNNEL	FUNNEL

Reference: Attribute Category Assignments for Funnel Charts

The following table lists the attribute category conversions for funnel charts.

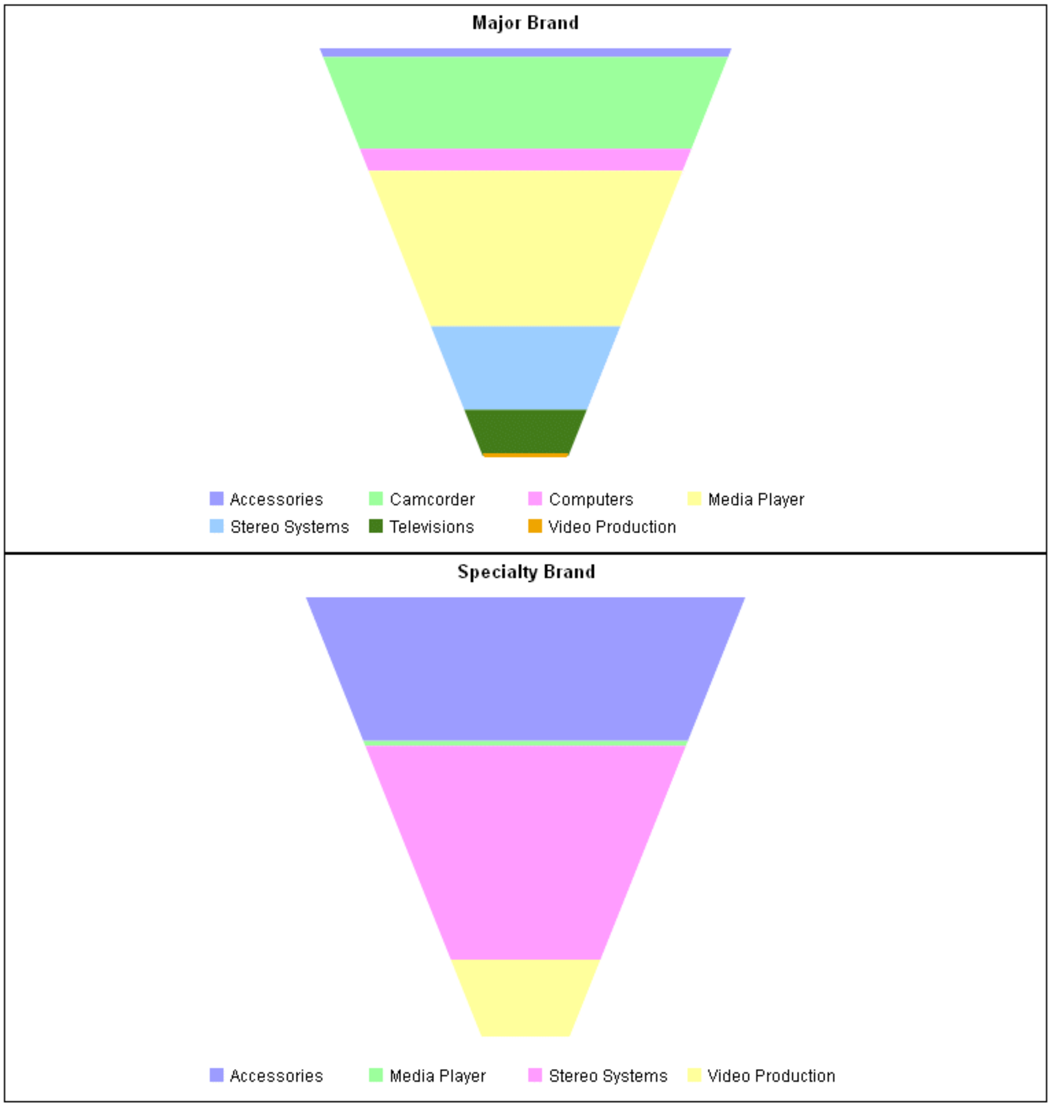
Type of Column or Parameter	**Attribute Category**
measure field	measure
Either GRLEGEND or GRXAXIS sort field (only 1 is supported)	color
GRMULTIGRAPH sort field	page

Example: Converting a Funnel Chart Request to Chart Attribute Syntax

The following example generates a funnel chart using traditional syntax (LOOKGRAPH value is FUNNEL). Separate funnel charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY sort field is the GRLEGEND sort field:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET AUTOFIT ON
END
```

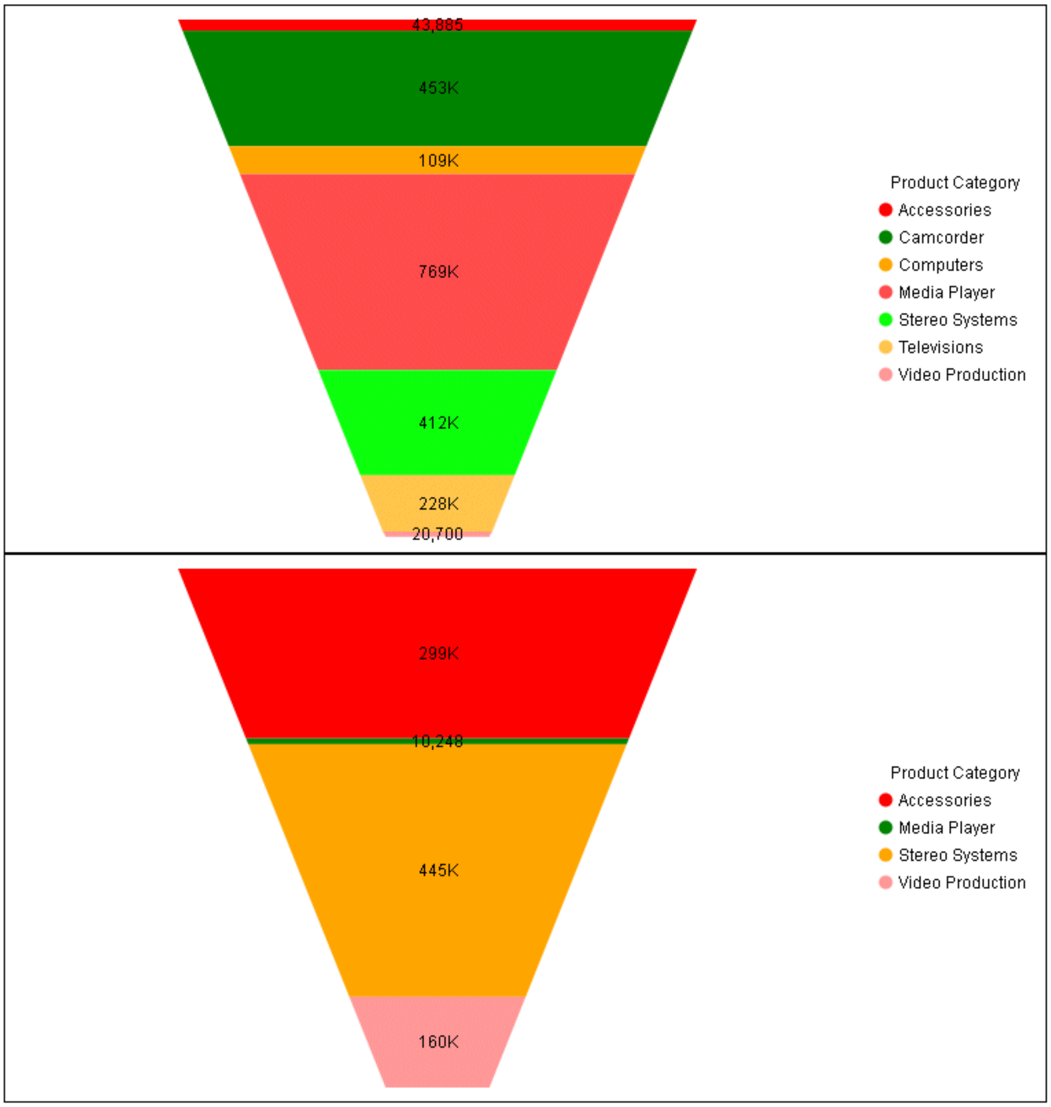
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is FUNNEL. The measure (COGS_US) is assigned to the measure attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (PRODUCT_CATEGORY) is assigned to the color attribute category. Some of the default properties, such as the colors and generation of data text labels, are different from the defaults for the traditional syntax.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH FUNNEL
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=measure,$
type=data, column=brandtype, bucket=page,$
type=data, column=product_category, bucket=color,$
ENDSTYLE
END
```

The output is shown in the following image:



Converting Map Chart Requests to Chart Attribute Syntax

Choropleth charts display maps in which the polygons that represent locations are colored by the value of a measure. Bubblemaps place proportional symbols on map locations or points (defined by latitude and longitude values).

Traditional WebFOCUS syntax only supports the Leaflet map engine for HTML5 charts. WebFOCUS chart attribute syntax supports both Leaflet and Esri map charts. This section will show how to convert the traditional WebFOCUS syntax to chart attribute syntax, not how to convert the map engine. For information about map charts, see *Map Support* on page 817.

Reference: LOOKGRAPH Conversions for Map Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values.

LOOKGRAPH Parameter	
CHOROPLETH	CHOROPLETH
BUBBLEMAP	BUBBLEMAP

Reference: Attribute Category Assignments for Map Charts

The following table lists the attribute category conversions for map charts.

Type of Column or Parameter	**Attribute Category**
measure field (choropleth)	color
measure field (bubblemap)	size
GRMULTIGRAPH sort field	page
GRLEGEND - not supported for maps	N/A
GRXAXIS sort field Choropleths can have only a location sort field, so GRXAXIS is always 1. Bubblemaps can have either a location sort field or latitude and longitude sort fields, so GRXAXIS can be 1 or 2,	When GRXAXIS is 1, convert to *location* attribute category. When GRXAXIS is 2, convert to *latitude* and *longitude* attribute categories.

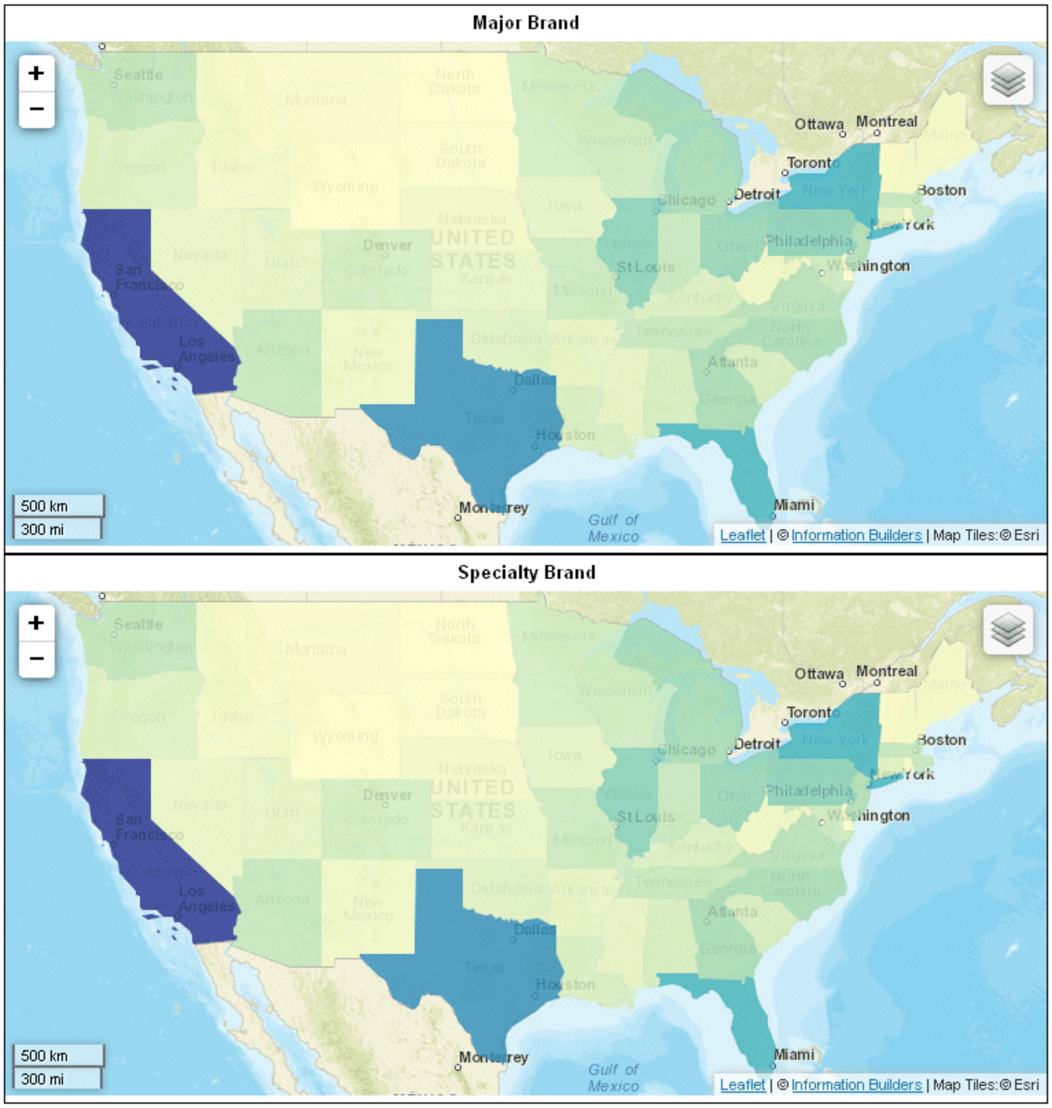
Example: Converting a Choropleth Chart Request to Chart Attribute Syntax

The following example generates multiple choropleth charts using traditional syntax (LOOKGRAPH value is CHOROPLETH). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate choropleth charts are generated for each value of BRANDTYPE. The measure (MIN.STATE_PROV_POPULATION) is used for the color of the polygons, and the sort field (STATE_PROV_NAME) is the location field (GRXAXIS field).

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.STATE_PROV_POPULATION
BY BRANDTYPE
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 1
ON GRAPH SET STYLE *
*GRAPH_JS
"mapProperties": {
"engine": "leaflet",
"leaflet": {
"initPos": {
"center": [37.8, -96],
"level": 4
},
"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
}],
"controls": [
{"control": "L.Control.Layers"},
{
"control": "L.Control.Scale",
"options": {
"imperial": true,
"metric": true }
}
],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 17,
"minZoom": 0,
"attribution": function(){ return "&|copy; <a target='_blank'href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles:&|copy; Esri";}
},
"url": function(){return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
},
*END
ENDSTYLE
END
```

The output is shown in the following image:

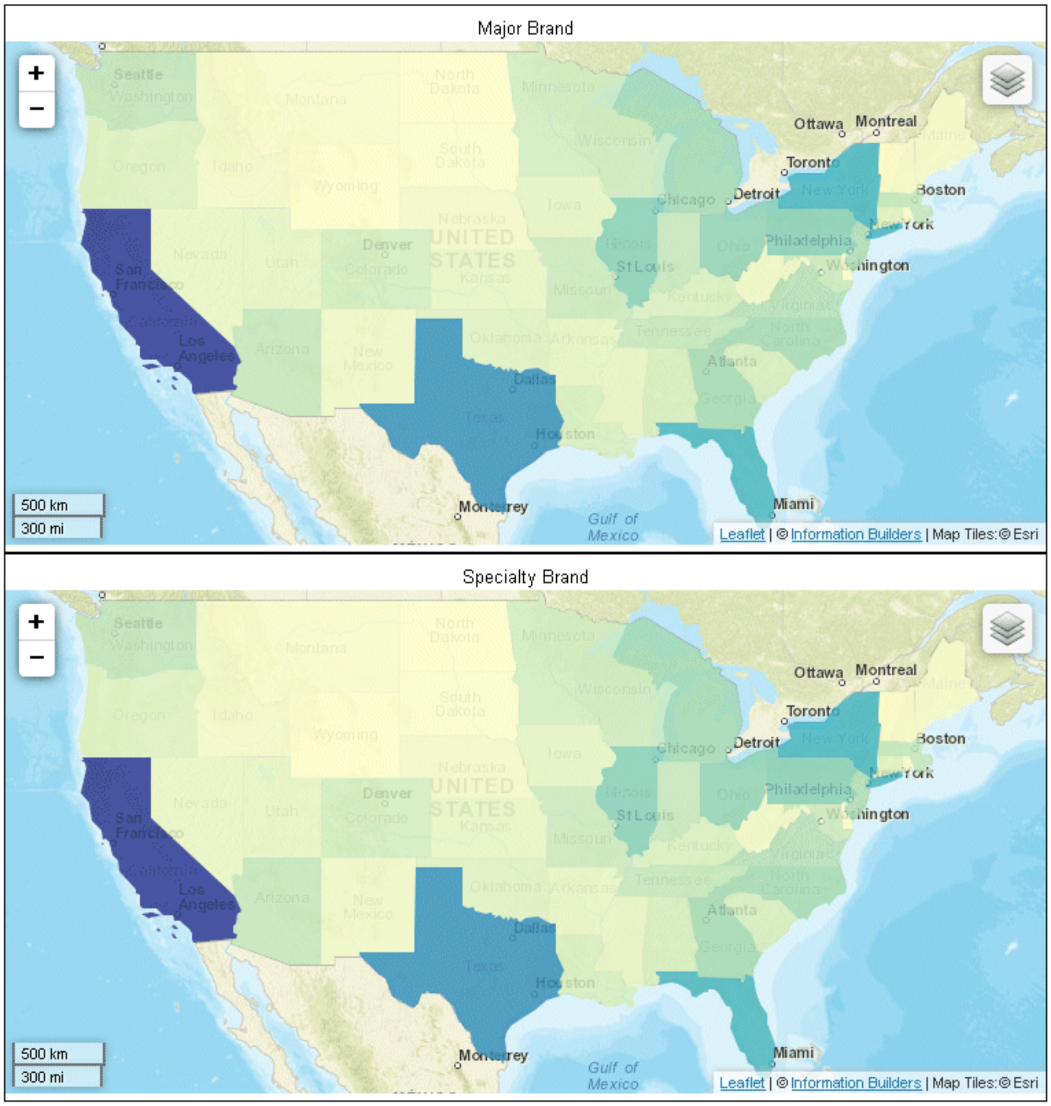


The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is CHOROPLETH. The measure (MIN.STATE_PROV_POPULATION) is assigned to the color attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (STATE_PROV_NAME) is assigned to the location attribute category. The visible:false property is set for the legend in order to make the output similar to that of the default traditional syntax.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
SUM MIN.STATE_PROV_POPULATION
BY BRANDTYPE
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH CHOROPLETH
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=PAGE,$
TYPE=DATA, COLUMN=MIN.STATE_PROV_POPULATION, BUCKET=color,$
TYPE=DATA, COLUMN=STATE_PROV_NAME, bucket=location,$
*GRAPH_JS
"legend":{"visible":false},
"legend":{"visible":false},
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },
"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
  "controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],
 "baselayers": [{
 "title": "ArcGIS_World_Street_Map",
 "layerInfo": {
 "maxZoom": 17,
 "minZoom": 0,
 "attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
  "url": function(){ return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';}
}]
}
},
*END
ENDSTYLE
END
```

The output is shown in the following image:



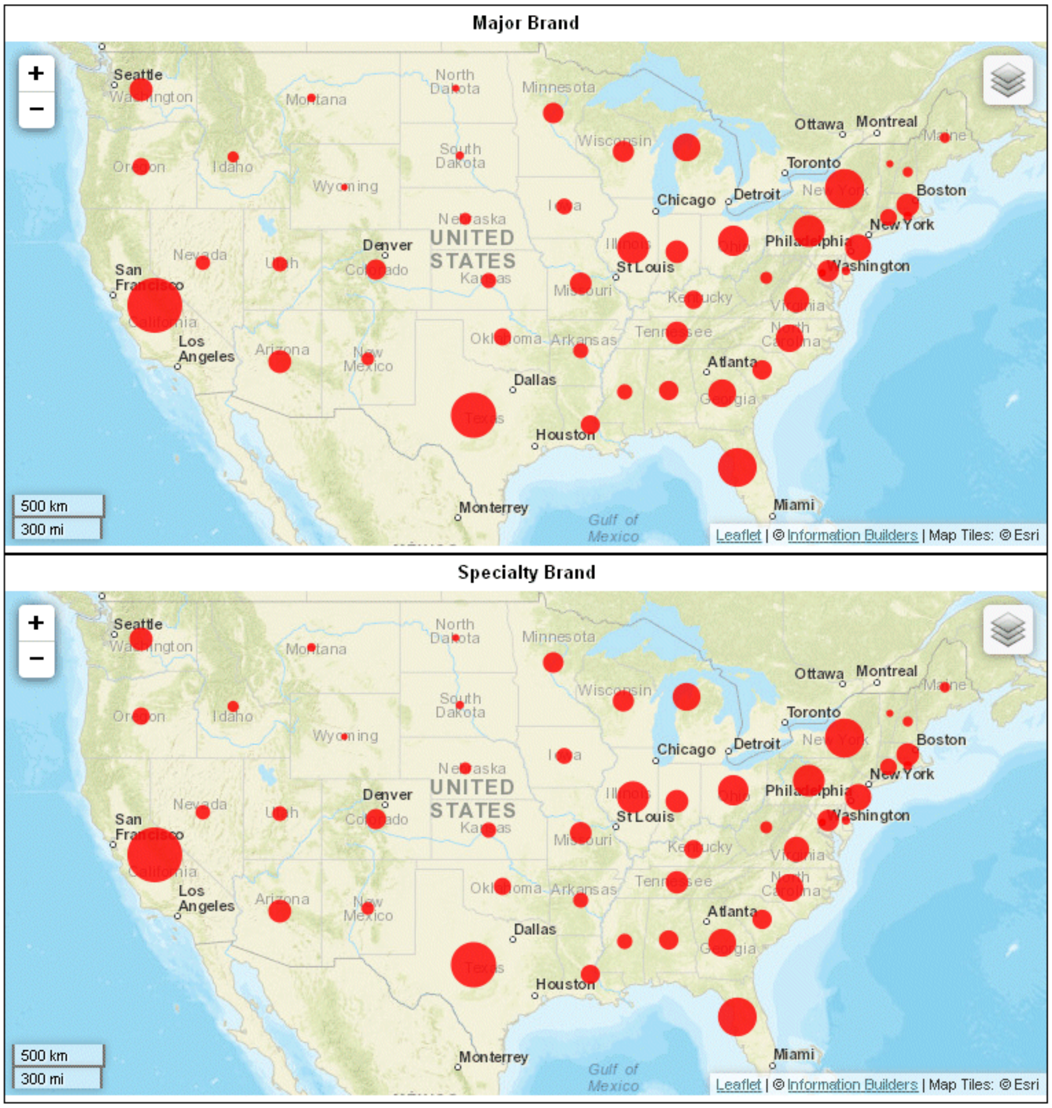
Example: Converting a Location-Based Bubblemap Chart Request to Chart Attribute Syntax

The following example generates multiple bubblemap charts using traditional syntax (LOOKGRAPH value is BUBBLEMAP). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate bubblemap charts are generated for each value of BRANDTYPE. The measure (MIN.STATE_PROV_POPULATION) is used for the size of the proportional symbols, and the sort field (STATE_PROV_NAME) is the location field (GRXAXIS field).

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
SUM MIN.STATE_PROV_POPULATION
BY BRANDTYPE
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET AUTOFIT ON
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 1
ON GRAPH SET STYLE *
*GRAPH_JS
"bubbleMarker":{maxSize:"10%"},
"mapProperties": {
  "engine": "leaflet",

"leaflet": {
"initPos": {
                  "center": [37.8, -96],
                  "level": 4
            },
"overlayLayers": [{
"title": "United States of America",
"dataLookup": "properties.state_name",
"layerInfo": {
"maxZoom": -1,
"minZoom": -1,
"type": "regions"
},
"type": "tdg",
"url": function(){ return tdgchart.getScriptPath() + 'map/US.json'}
}],
  "controls": [
    {"control": "L.Control.Layers"},
    {
     "control": "L.Control.Scale",
     "options": {
      "imperial": true,
      "metric": true }
    }
   ],
 "baselayers: [{
 "title": "ArcGIS_World_Street_Map",
 "layerInfo": {
 "maxZoom": 17,
 "minZoom": 0,
 "attribution": function(){ return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy; Esri";}
},
  "url": function(){ return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';}
}]
}
},
*END
ENDSTYLE
END
```

The output is shown in the following image:

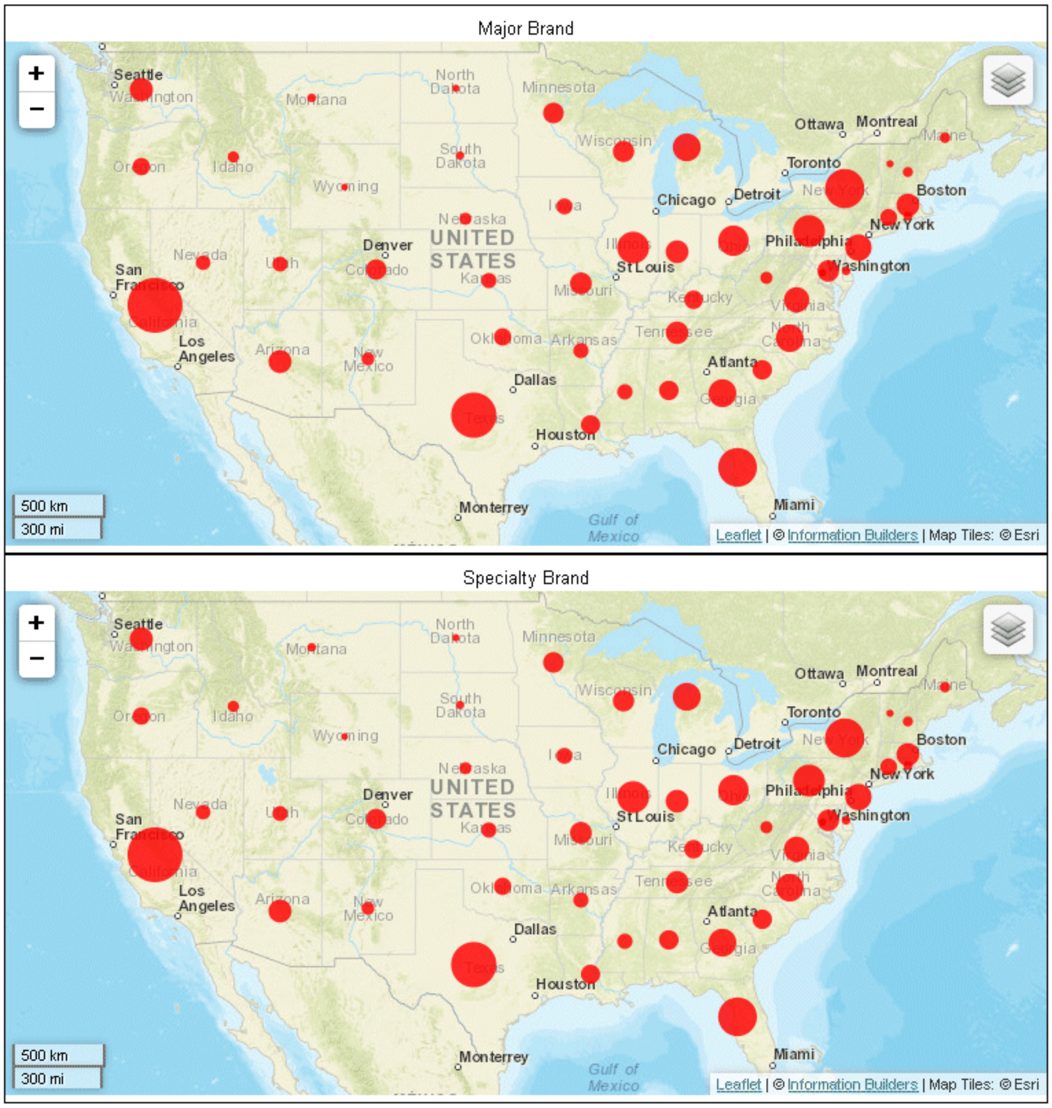


The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is BUBBLEMAP. The measure (MIN.STATE_PROV_POPULATION) is assigned to the size attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (STATE_PROV_NAME) is assigned to the location attribute category. The visible:false property is set for the legend in order to make the output similar to that of the traditional syntax.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
SUM MIN.STATE_PROV_POPULATION
BY BRANDTYPE
BY STATE_PROV_NAME
WHERE COUNTRY_NAME EQ 'United States'
WHERE STATE_PROV_NAME NE 'Puerto Rico'
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET AUTOFIT ON
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=PAGE,$
TYPE=DATA, COLUMN=STATE_PROV_NAME, BUCKET=LOCATION,$
TYPE=DATA,COLUMN= MIN.STATE_PROV_POPULATION, BUCKET=SIZE,$
*GRAPH_JS
"legend":{"visible":false},
"bubbleMarker":{"maxSize":"10%"},"mapProperties": {
"leaflet": {
"initPos": {
"center": [37.8, -96],
"level": 4
},
"overlayLayers": [{
"title": "US",
"layerInfo": {
"type": "regions"
},
"type": "tdg",
"url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
}],
"baselayers": [{
"title": "ArcGIS_World_Street_Map",
"layerInfo": {
"maxZoom": 16,
"attribution": function() { return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles:&|copy; Esri"; }
},
"url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}';
}
}]
}
}
*END
ENDSTYLE
END
```

The output is shown in the following image:



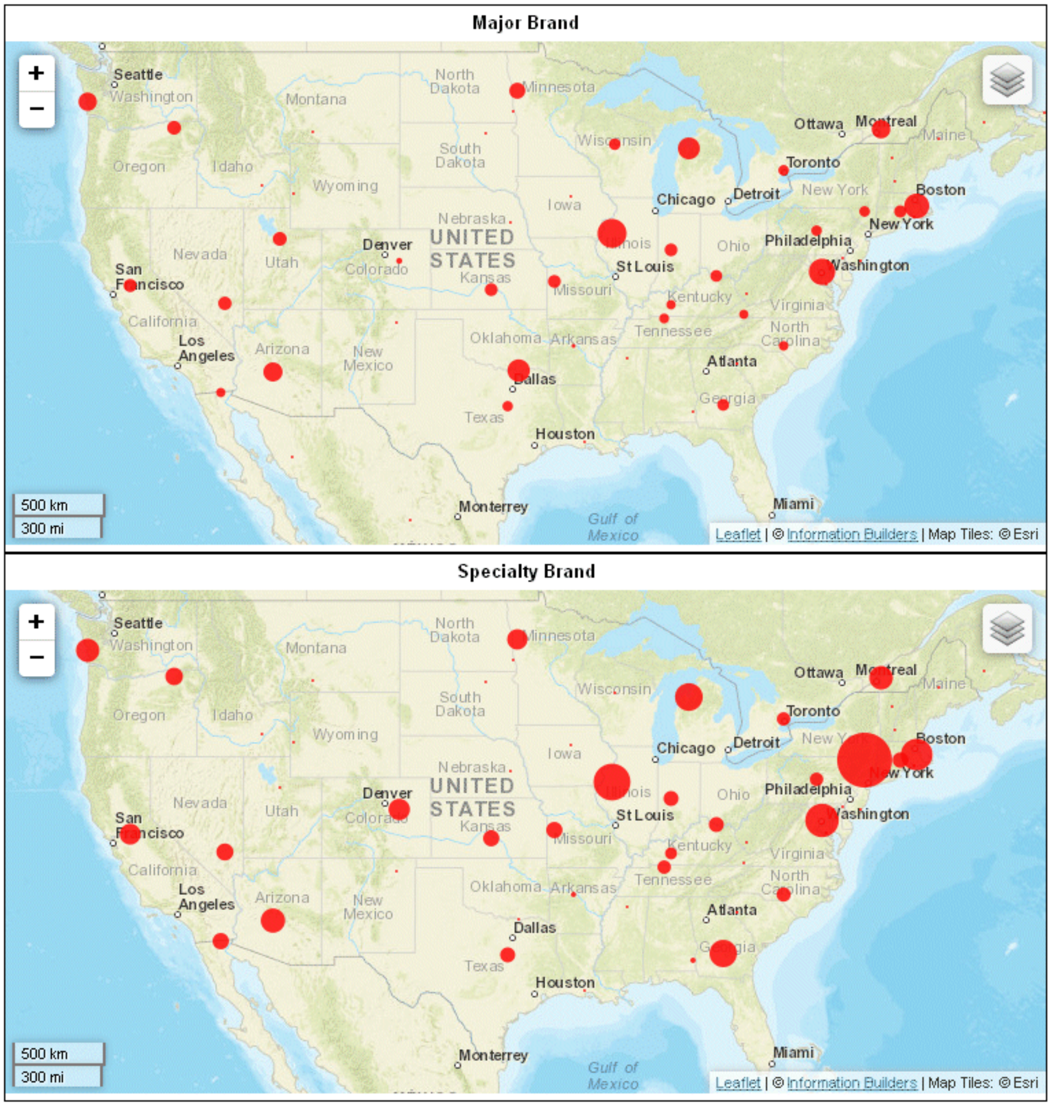
Example: Converting a Latitude and Longitude-Based Bubblemap Chart Request to Chart Attribute Syntax

The following example generates multiple bubblemap charts using traditional syntax (LOOKGRAPH value is BUBBLEMAP). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate bubblemap charts are generated for each value of BRANDTYPE. The measure (MIN.CITY_POPULATION) is used for the size of the proportional symbols, and the latitude and longitude sort fields (STATE_PROV_LATITUDE and STATE_PROV_LONGITUDE) are used for the GRXAXIS value (2).

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE wf_retail_lite
SUM MIN.CITY_POPULATION
BY BRANDTYPE
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 2
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
*GRAPH_JS
"bubbleMarker":{"maxSize":"10%"},
"mapProperties": {
  "leaflet": {
   "initPos": {
    "center": [37.8, -96],
     "level": 4
              },
   "overlayLayers": [{
      "title": "US",
      "layerInfo": {
       "type": "regions"
                },
      "type": "latlng",
       "url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
            }],
   "baselayers": [{
    "title": "ArcGIS_World_Street_Map",
    "layerInfo": {
     "maxZoom": 16,
     "attribution": function() {  return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy;
Esri"; }
                },
     "url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}'; }
            }]
    }
}
*END
ENDSTYLE
END
```

The output is shown in the following image:

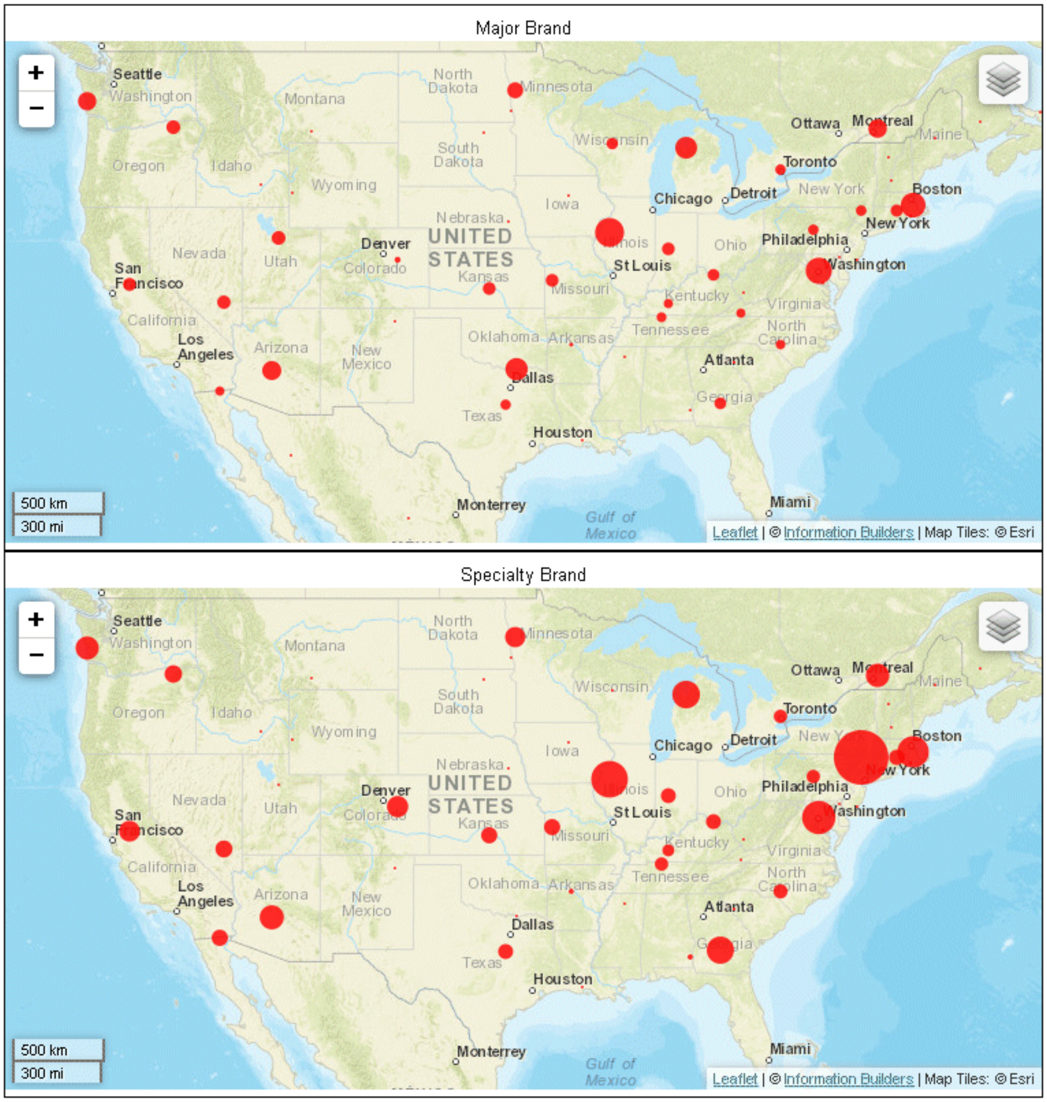


The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is BUBBLEMAP. The measure (MIN.CITY_POPULATION) is assigned to the size attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort fields (STATE_PROV_LATITUDE and STATE_PROV_LONGITUDE) are assigned to the latitude and longitude attribute categories. The visible:false property is set for the legend in order to make the output similar to that of the default traditional syntax, and an embedded heading is added to display the brand types.

Note: Due to their length, certain lines of code in the example below may wrap onto the next line of text. Wrapping may create breaks within strings or URL references, which may cause errors when run. If you copy and paste this example, be sure to remove these line breaks before running it.

```
GRAPH FILE wf_retail_lite
HEADING CENTER
"<BRANDTYPE "
SUM MIN.CITY_POPULATION
BY BRANDTYPE
BY STATE_PROV_LATITUDE
BY STATE_PROV_LONGITUDE
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH BUBBLEMAP
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=PAGE,$
TYPE=DATA, COLUMN=MIN.CITY_POPULATION, BUCKET=SIZE,$
TYPE=DATA, COLUMN=STATE_PROV_LATITUDE, BUCKET=LATITUDE,$
TYPE=DATA, COLUMN=STATE_PROV_LONGITUDE, BUCKET=LONGITUDE,$
*GRAPH_JS
"legend":{"visible":false},
"bubbleMarker":{"maxSize":"10%"},
"mapProperties": {
  "leaflet": {
   "initPos": {
    "center": [37.8, -96],
     "level": 4
              },
   "overlayLayers": [{
      "title": "US",
      "layerInfo": {
       "type": "regions"
                },
      "type": "latlng",
       "url": function(){return tdgchart.getScriptPath() + 'map/US.json'}
            }],
   "baselayers": [{
    "title": "ArcGIS_World_Street_Map",
    "layerInfo": {
     "maxZoom": 16,
     "attribution": function() {  return "&|copy; <a target='_blank' href='http://
www.InformationBuilders.com'>Information Builders</a> | " + "Map Tiles: &|copy;
Esri"; }
                },
     "url": function() { return 'http://services.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}'; }
            }]
    }
}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Converting Mekko Chart Requests to Chart Attribute Syntax

A mekko chart is a percent bar chart, except that the width of each bar riser is based on the overall value of the stack.

Reference: LOOKGRAPH Conversions for Mekko Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with additional properties that may be needed for the chart type.

LOOKGRAPH Parameter	
MEKKO	MEKKO

Reference: Attribute Category Assignments for Mekko Charts

The following table lists the attribute category conversions for mekko charts.

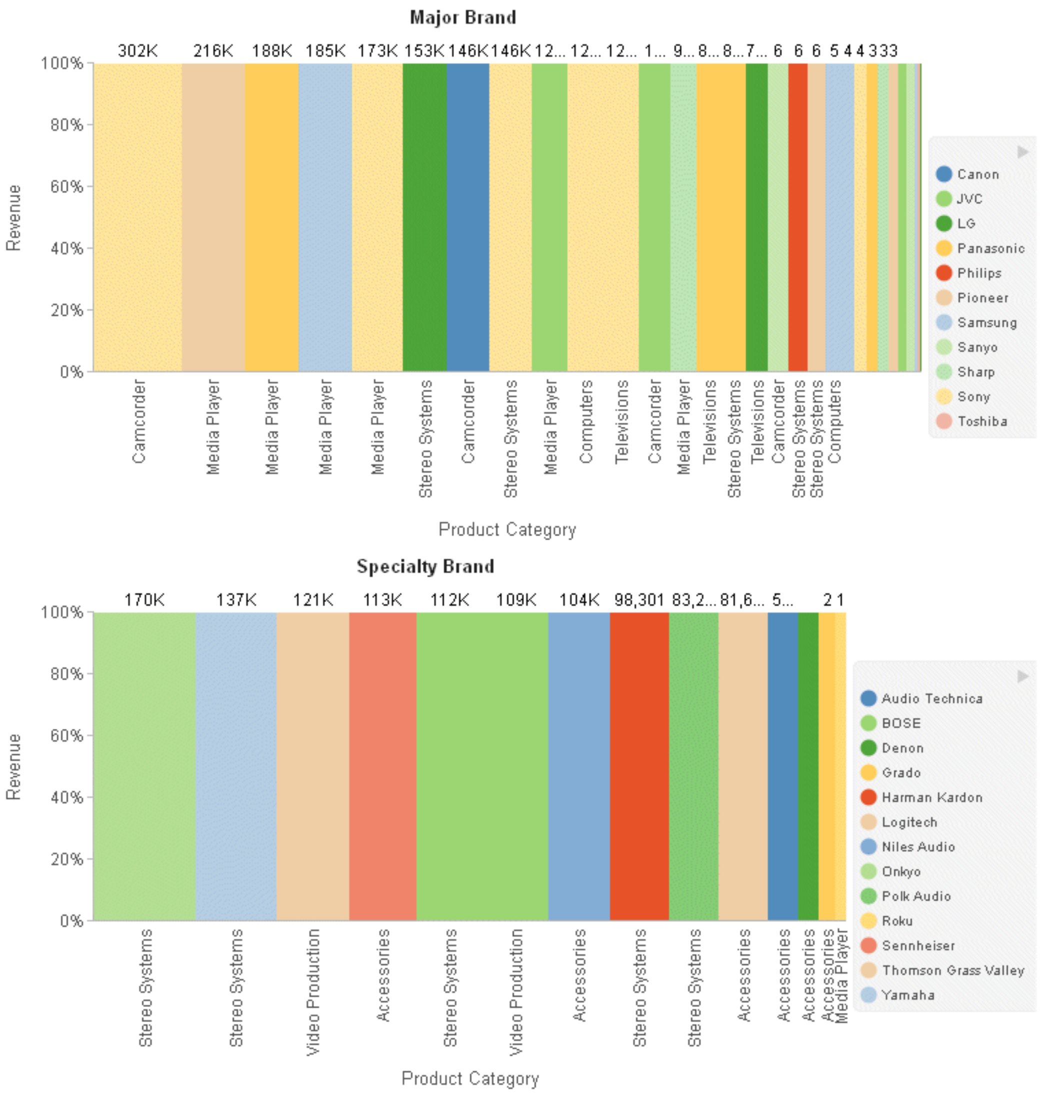
Type of Column or Parameter	**Attribute Category**
measure field	y-axis
GRXAXIS sort field	x-axis
GRLEGEND sort field	color
GRMULTIGRAPH sort field	page

Example: Converting a Mekko Chart Request to Chart Attribute Syntax

The following example generates a mekko chart that separates the outermost sort field (BRANDTYPE) onto separate charts, distinguishes the next sort field (BRAND) by placing it on the graph legend, and places the PRODUCT_CATEGORY sort field on the x-axis:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US
BY BRANDTYPE
BY BRAND
BY PRODUCT_CATEGORY
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH MEKKO
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

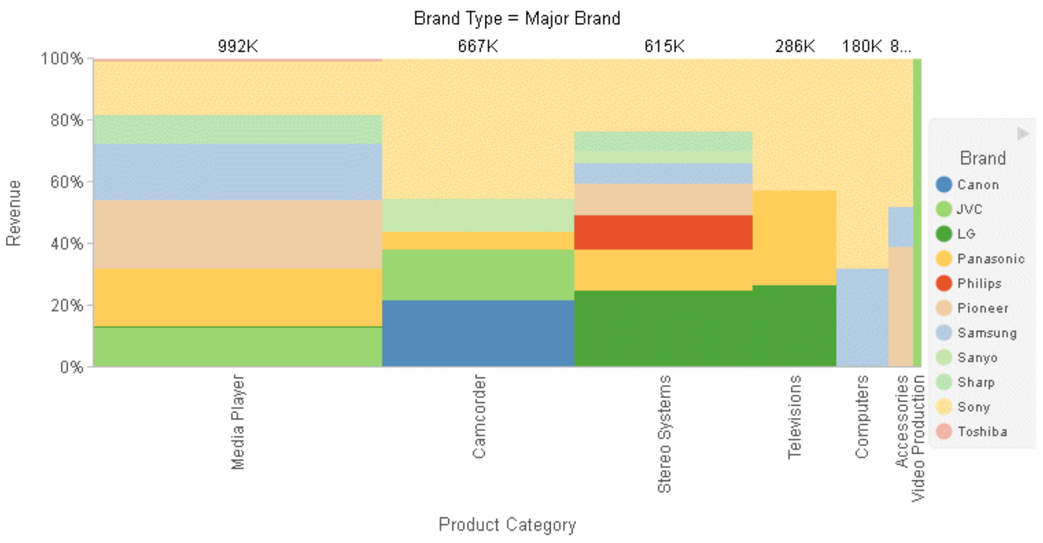
The output is shown in the following image:

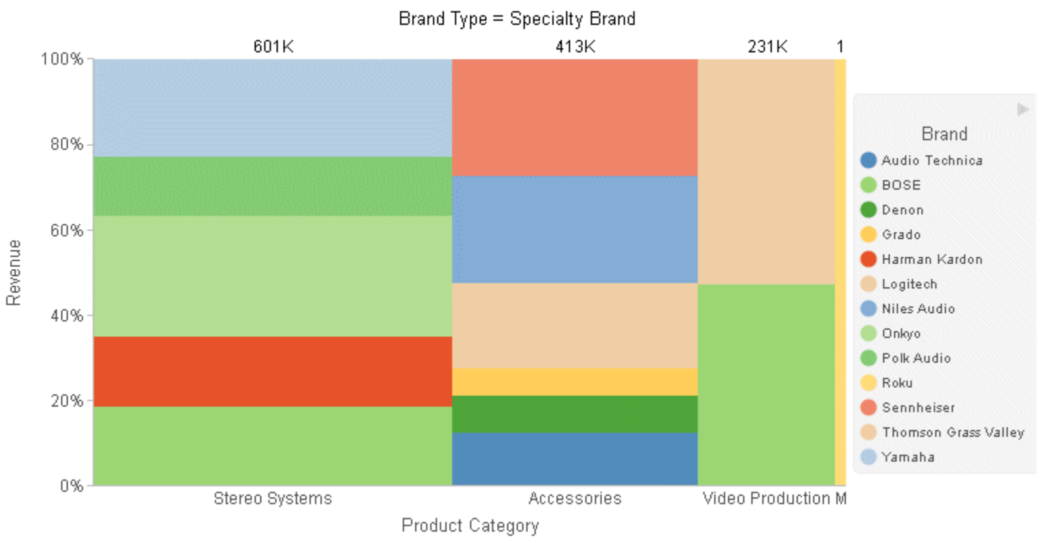


The following is the same request converted to chart attribute syntax. The BRANDTYPE sort field is assigned to the page category, the BRAND sort field is assigned to the color category, the PRODUCT_CATEGORY sort field is assigned to the x-axis category, and the REVENUE_US measure is assigned to the y-axis category. The chart type is MEKKO:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"Brand Type = <BRANDTYPE"
SUM REVENUE_US
BY BRANDTYPE
BY BRAND
BY PRODUCT_CATEGORY
ON GRAPH SET LOOKGRAPH MEKKO
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=page,$
TYPE=DATA, COLUMN=BRAND, BUCKET=COLOR,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=x-axis,$
TYPE=DATA, COLUMN=REVENUE_US, BUCKET=y-axis,$
END
```

The output shown in the following image looks different from the GRMERGE output, since chart attribute syntax clusters similar x-axis values together. For example, the media players from all of the brands are displayed as one stacked bar:





Converting Pie Chart Requests to Chart Attribute Syntax

Pie charts slice a measure based on a sort field. A pie chart can have a hole. The defaults for several properties vary depending on whether the request uses traditional WebFOCUS syntax, chart attribute syntax, or a StyleSheet.

Reference: LOOKGRAPH Conversions for Pie Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values along with additional properties that may be needed for the chart type.

LOOKGRAPH Parameter	
PIE	PIE
PIEMULTI	PIE

PIERING	PIE
	
	Add the following JSON property:
	
	`*GRAPH_JS`
	`"pieProperties": {`
	`    "holeSize": "10%"`
	`}`
	
	You can change the percentage used for the hole size.

PIEMULTR	PIE
	
	Add the following JSON property:
	
	`*GRAPH_JS`
	`"pieProperties": {`
	`    "holeSize": "10%"`
	`}`
	
	You can change the percentage used for the hole size.

Reference: Attribute Category Assignments for Pie Charts

The following table lists the attribute category conversions for pie charts.

Type of Column or Parameter	Attribute Category
measure field	measure
Either GRLEGEND or GRXAXIS sort field	color, column, or row
GRMULTIGRAPH sort field	page

For a pie chart, the WebFOCUS tools have allowed limited multiple sort fields, even with traditional syntax. The user dragged these fields to the Category and Slices components. However, the tools now convert these requests to chart attribute syntax before running them. Depending on the number and types of these additional sort fields, the tools assign them to the column and color attribute categories, as shown in the following table. The column category generates a matrix of pie charts, one in each column (or row, depending on your configuration options) of a single-row (column) matrix. For complete information about chart attribute categories, see *WebFOCUS Chart Attribute Syntax* on page 143.

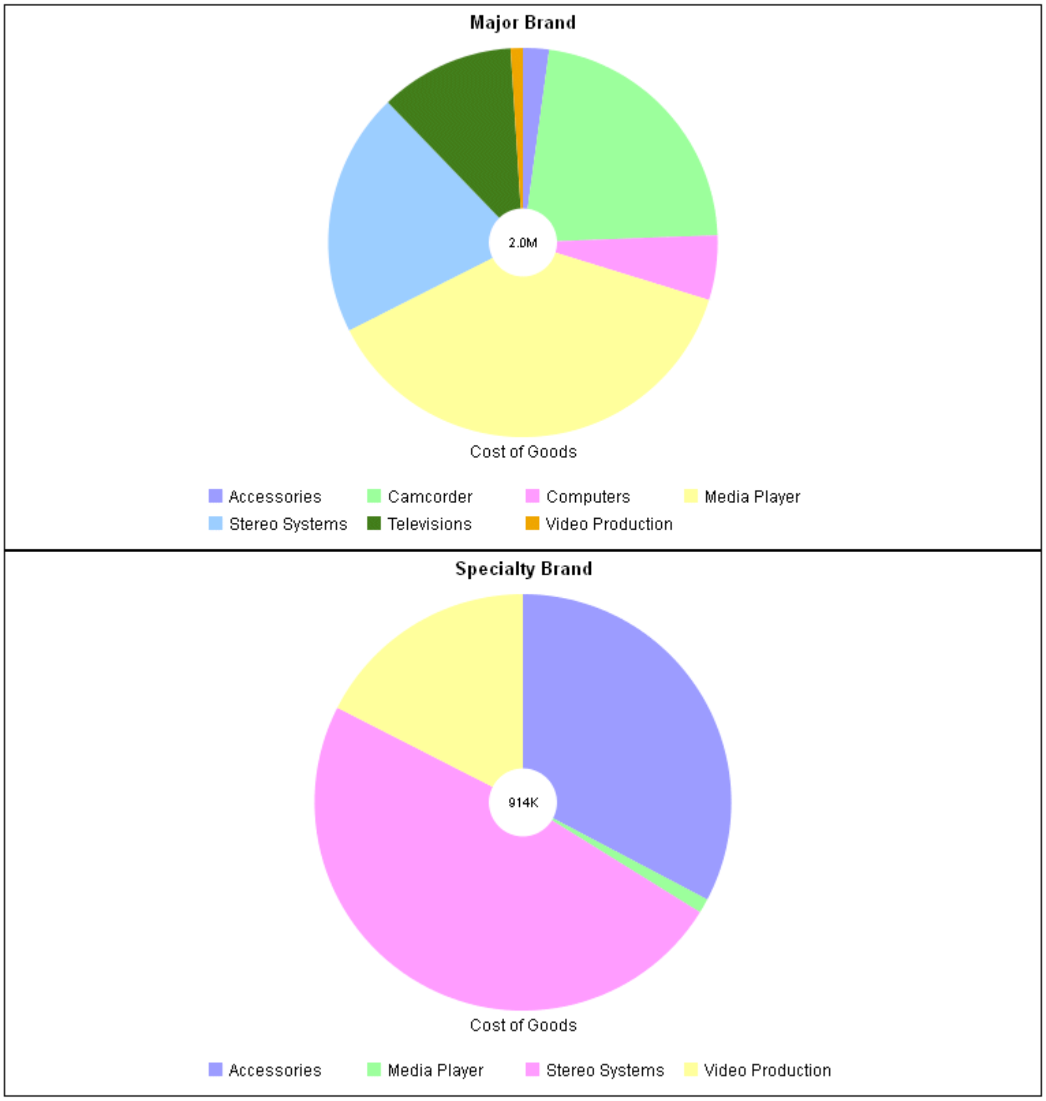
# of Category Fields	# of Slices Fields	Assigned to Column Attribute Category	Assigned to Color Attribute Category
1	0	0	Category field
0	1	0	Slices field
1	1	Category field	Slices field
2	0	First field	Second field
2	1	Slices field first, then first Category field	Second Category field

Example: Converting a Pie Chart Request to Chart Attribute Syntax

The following example generates multiple pie ring charts using traditional syntax (LOOKGRAPH value is PIEMULTR). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate pie charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY sort field is the GRLEGEND sort field:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET LOOKGRAPH PIEMULTR
END
```

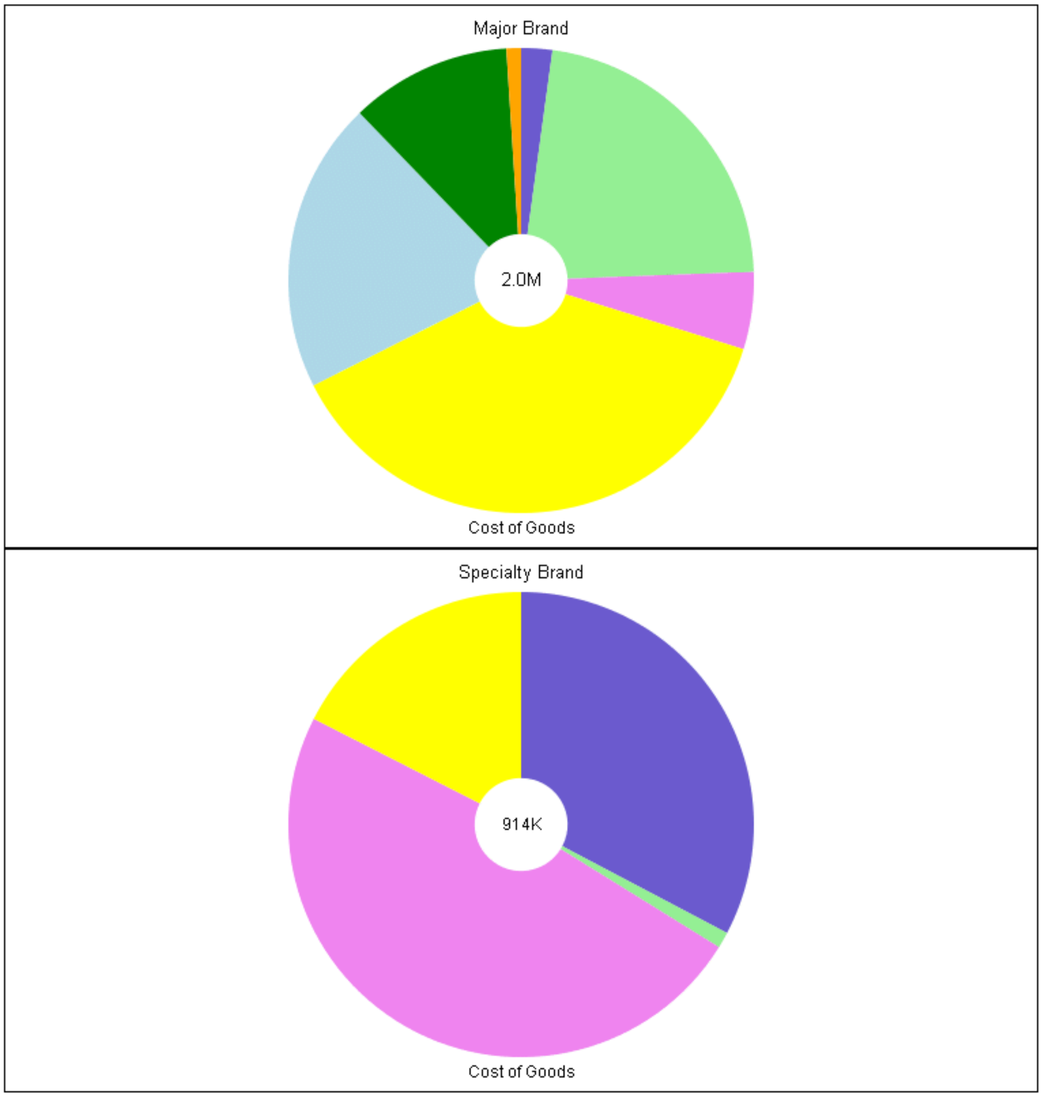
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is PIE. The measure (COGS_US) is assigned to the measure attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (PRODUCT_CATEGORY) is assigned to the color attribute category. The pieProperties:holeSize property is set to 20%. Some other JSON properties are set to assign colors to the slices and make the legend not visible:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color,$
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=page,$
*GRAPH_JS
"legend": {"visible": false},
"series": [
    {"series": 0, "color": "slateblue"},
    {"series": 1, "color": "lightgreen"},
    {"series": 2, "color": "violet"},
    {"series": 3, "color": "yellow"},
    {"series": 4, "color": "lightblue"},
    {"series": 5, "color": "green"},
    {"series": 6, "color": "orange"}],
"pieProperties": {
    "holeSize": "20%",
    "label": {"visible": true},
    "totalLabel": {"visible": true}}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Example: Converting a Pie Chart Request to a Columnar Chart Layout

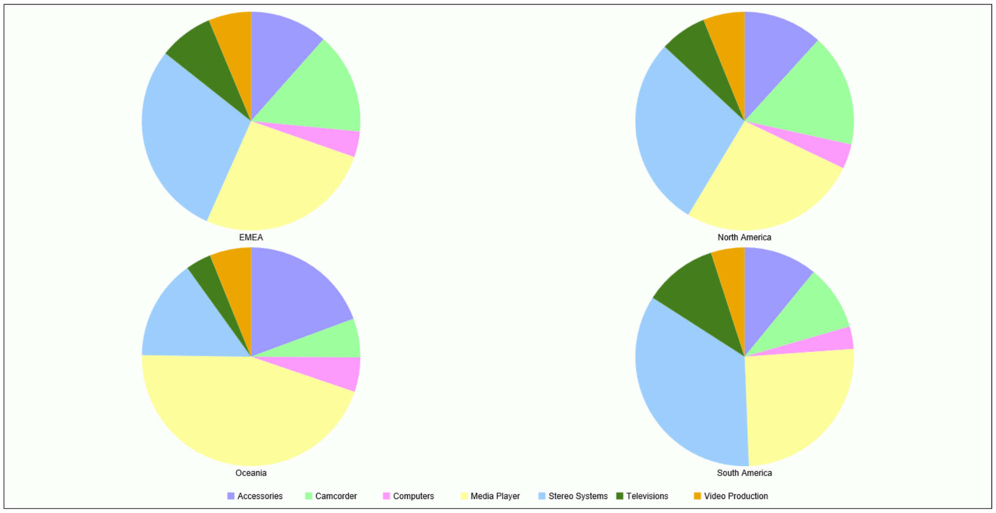
For a pie chart, the WebFOCUS tools have allowed limited multiple sort fields, even with traditional syntax. The user dragged these fields to the Category and Slices components. However, the tools now convert these requests to chart attribute syntax before running them.

Depending on the number and types of these additional sort fields, the tools assign them to the column and color attribute categories. The column category generates a matrix of pie charts, one in each column (or row, depending on your configuration options) of a single-row (column) matrix. This generates a layout for the multiple pie charts that looks different from the layout generated by the traditional syntax, where all of the pies were in the same frame.

The following example generates multiple pie charts using traditional WebFOCUS chart syntax and assigns the PRODUCT_CATEGORY sort field to the GRLEGEND parameter and the BUSINESS_REGION sort field to the GRXAXIS parameter.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
ACROSS BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH SET AUTOFIT ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH PIEMULTI
END
```

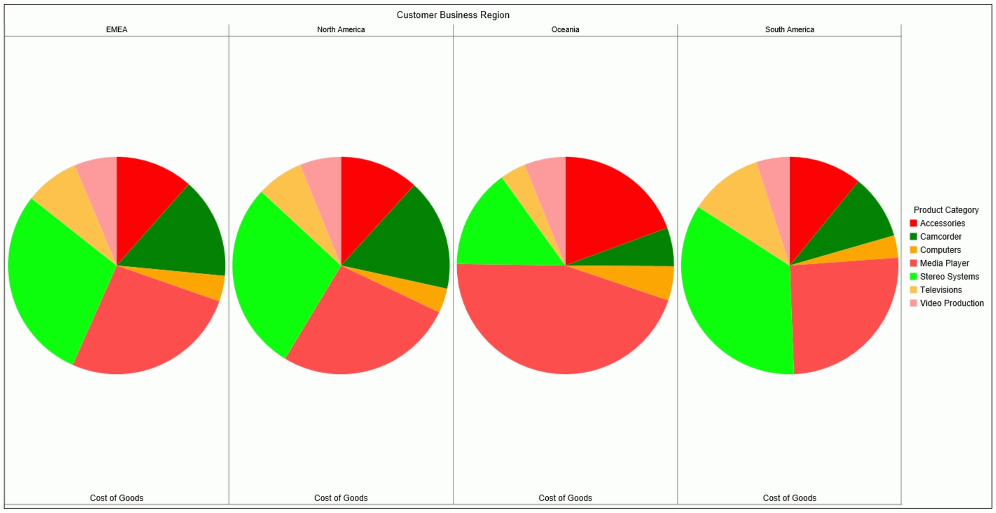
The output is shown in the following image.



The following version of the request uses chart attribute syntax. The PRODUCT_CATEGORY sort field is assigned to the *color* category, and the BUSINESS_REGION sort field is assigned to the *column* category.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BUSINESS_REGION
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET AUTOFIT ON
ON GRAPH SET LOOKGRAPH PIE
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=COGS_US, BUCKET=measure,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color,$
TYPE=DATA, COLUMN=BUSINESS_REGION, BUCKET=column,$
END
```

The output is shown in the following image. Note that each pie is in its own column.



Converting Scatter Chart Requests to Chart Attribute Syntax

Scatter charts generally plot one measure against another, although WebFOCUS provides LOOKGRAPH values that support dimensions as well.

Reference: LOOKGRAPH Conversions for Scatter Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values.

LOOKGRAPH Parameter	
SCATTER	SCATTER
SCATTERN	SCATTER
SCATTERS	SCATTER

Reference: Attribute Category Assignments for SCATTER Charts

The following table lists the attribute category conversions for SCATTER charts. Charts with LOOKGRAPH SCATTER have a numeric sort field, which is displayed on the x-axis, and numeric measures that are displayed on the y-axis.

Note: Merge parameters are not supported with LOOKGRAPH SCATTER.

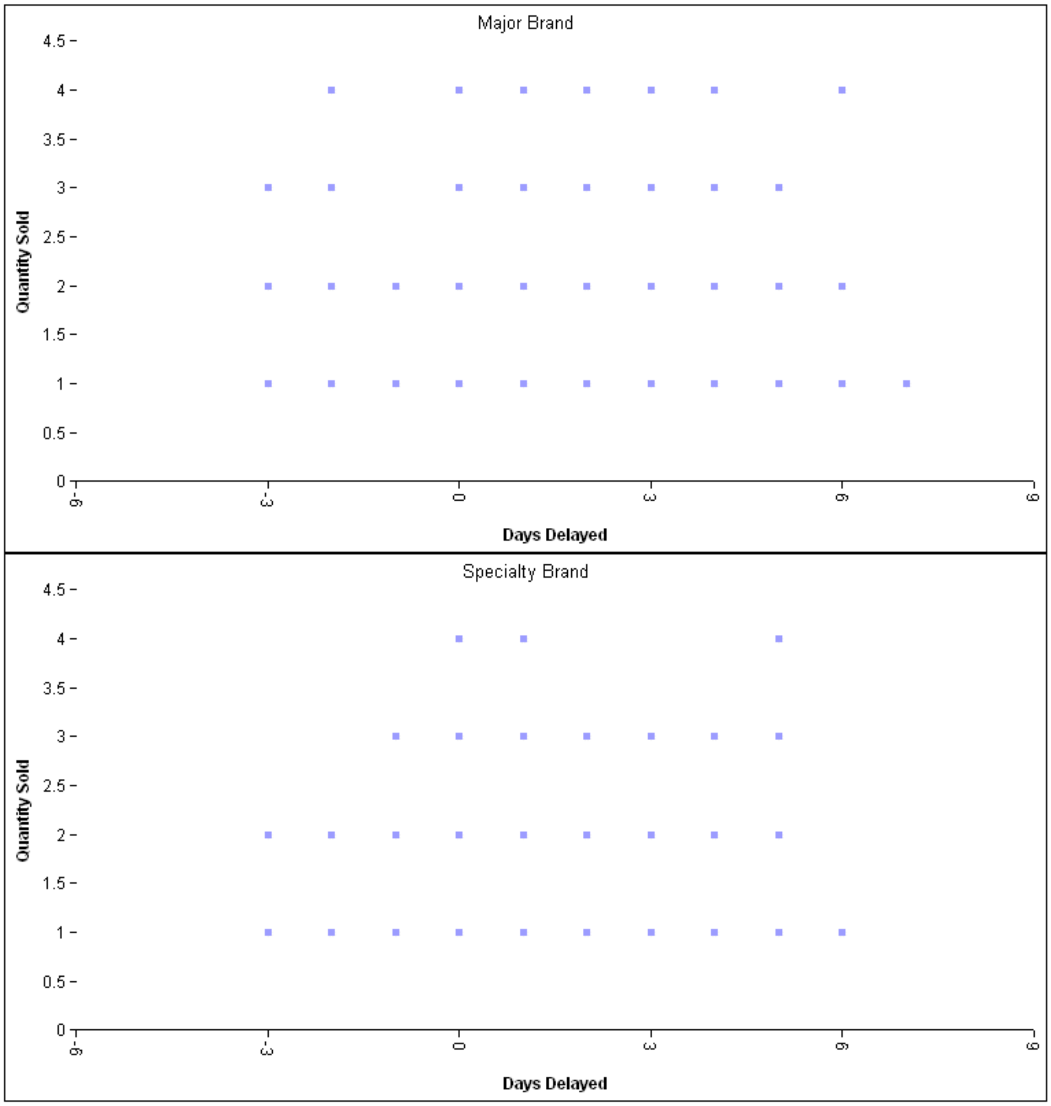
Type of Column or Parameter	Attribute Category
Measure field.	y-axis
Sort field. Must be numeric for SCATTER.	x-axis
High-level sort field	page

Example: Converting a SCATTER Chart Request to Chart Attribute Syntax

The following example generates multiple scatter charts using traditional syntax (LOOKGRAPH value is SCATTER). The high-level sort field (BRANDTYPE) is the sort field that generates multiple charts, the measure (QUANTITY_SOLD) is plotted on the y-axis, and the numeric sort field (DAYSDELAYED) is plotted on the x-axis:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
PRINT QUANTITY_SOLD
BY BRANDTYPE
BY DAYSDELAYED
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH PCHOLD FORMAT JSCHART
END
```

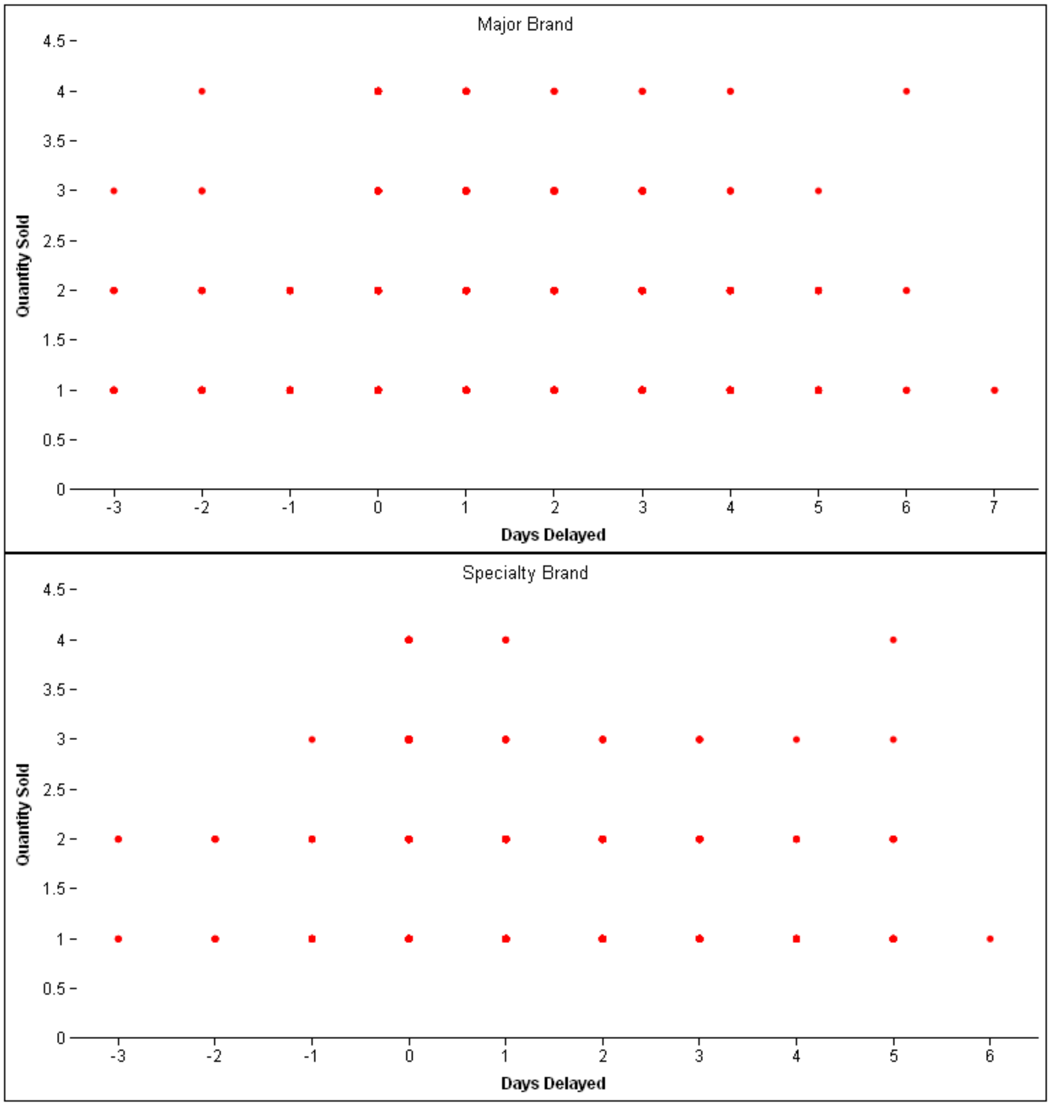
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is SCATTER. The measure (QUANTITY_SOLD) is assigned to the y-axis category, the numeric sort field (DAYSDELAYED) is assigned to the x-axis attribute category, and the high-level sort field (BRANDTYPE) is assigned to the page attribute category.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
PRINT QUANTITY_SOLD
BY BRANDTYPE
BY DAYSDELAYED
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
type=data, column=quantity_sold, bucket=y-axis,$
type=data, column=daysdelayed, bucket=x-axis,$
type=data, column=brandtype, bucket=page,$
*GRAPH_JS
"xaxis": {"majorGrid": {"visible": false}},
"yaxis": {"majorGrid": {"visible": false}}
*END
END
```

The output is shown in the following image:



Reference: Attribute Category Assignments for SCATTERN Charts

Charts with LOOKGRAPH SCATTERN must have two measures for the axes, and the sort field represents the plotted points.

The following table lists the attribute category conversions for SCATTERN charts.

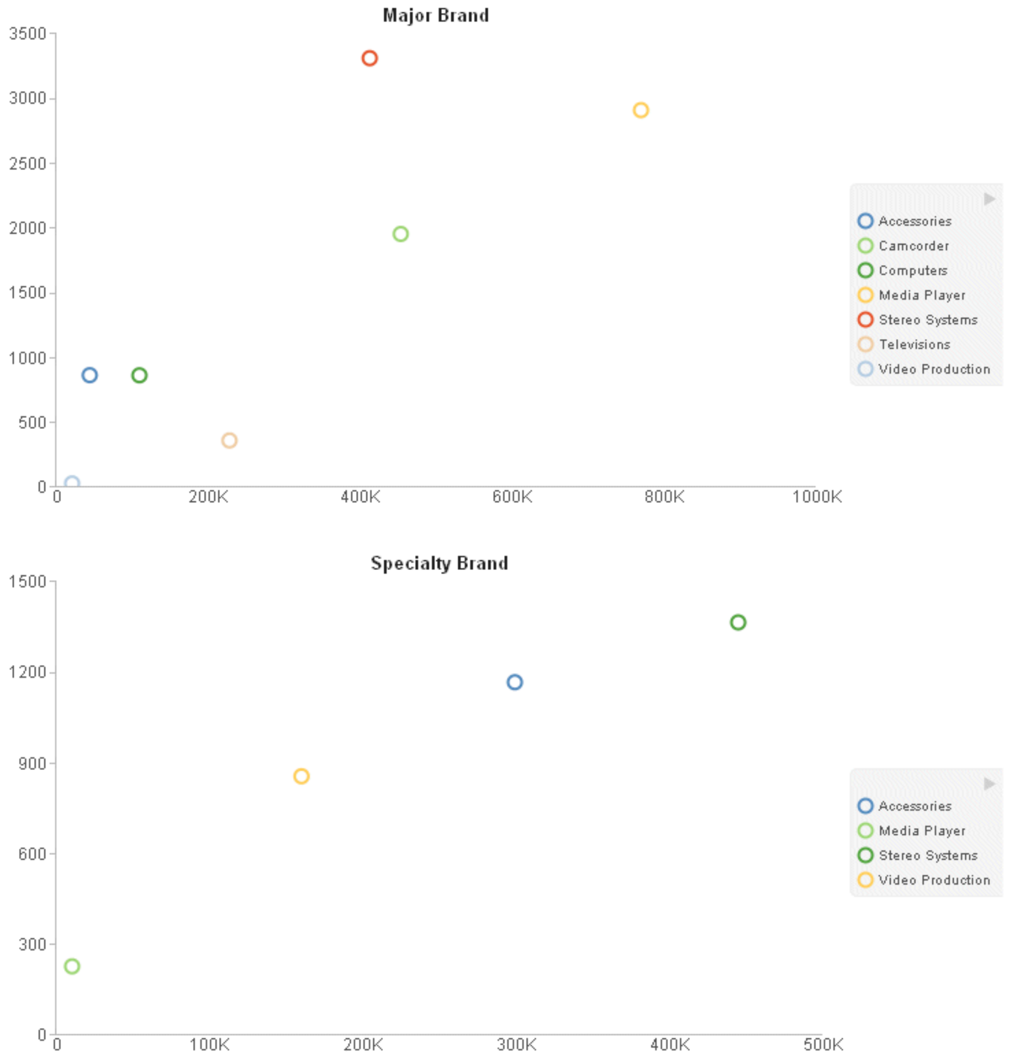
Type of Column or Parameter	Attribute Category
Measure field.	x-axis
Measure field.	y-axis
GRLEGEND or GRXAXIS sort field (dimension or measure). With SCATTERN, there must be two measures for the axes, and the sort field represents the points plotted.	color
GRMULTIGRAPH sort field	page

Example: Converting a SCATTERN Chart Request to Chart Attribute Syntax

The following example generates multiple scatter charts using traditional syntax (LOOKGRAPH value is SCATTERN). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate scatter charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY sort field is the GRLEGEND sort field:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM COGS_US
QUANTITY_SOLD
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERN
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 0
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

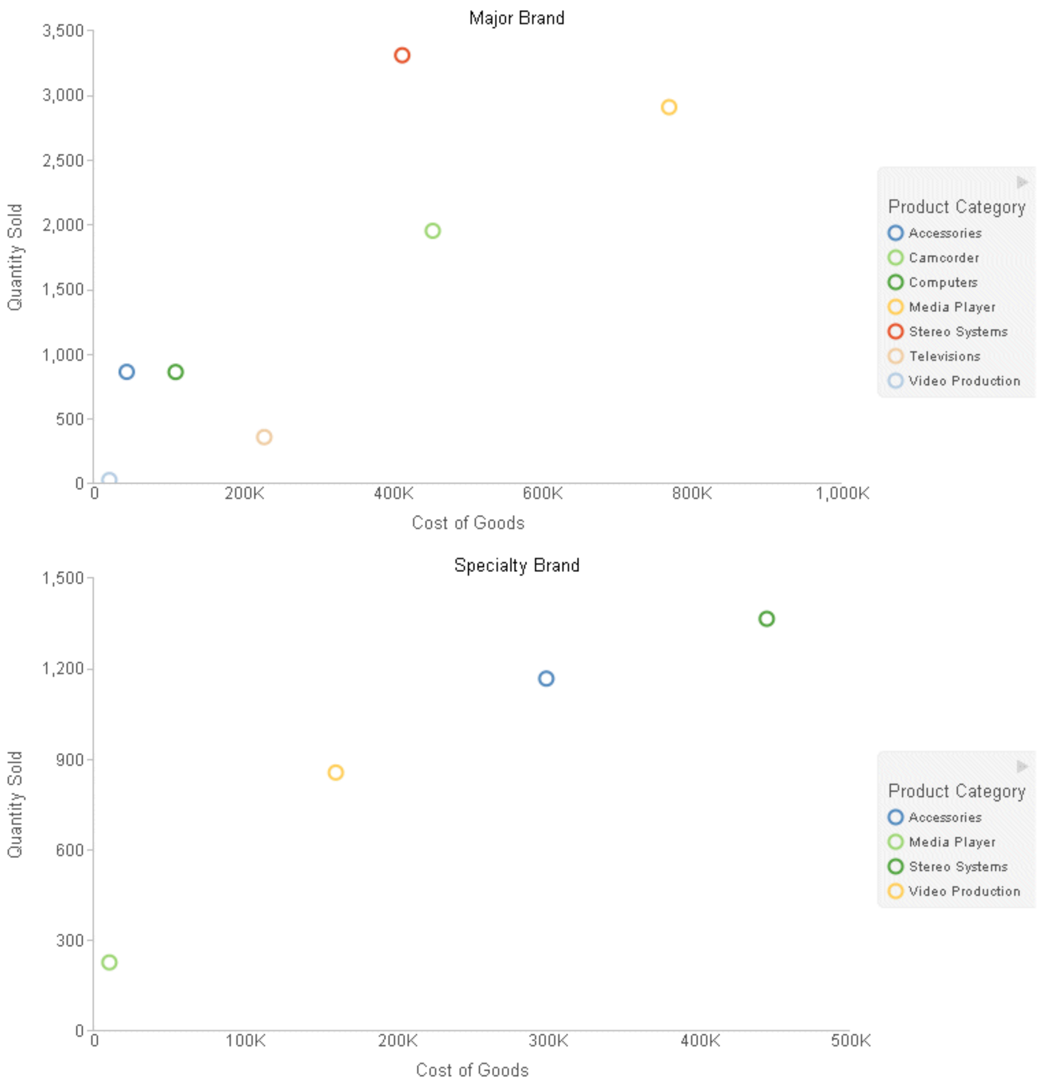
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is SCATTER. The measures (COGS_US and QUANTITY_SOLD) are assigned to the x-axis and y-axis attribute categories, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (PRODUCT_CATEGORY) is assigned to the color attribute category.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
SUM COGS_US QUANTITY_SOLD
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=QUANTITY_SOLD, BUCKET=y-axis,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=x-axis,$
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=page,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=color,$
END
```

The output is shown in the following image:



Reference: Attribute Category Assignments for SCATTERS Charts

Charts with LOOKGRAPH SCATTERS can have any number of measures to be plotted on the y-axis, and the sort field becomes the x-axis. However, chart attribute syntax supports only one y-axis measure.

Note: Merge parameters are not supported with LOOKGRAPH SCATTERS.

The following table lists the attribute category conversions for SCATTERS charts.

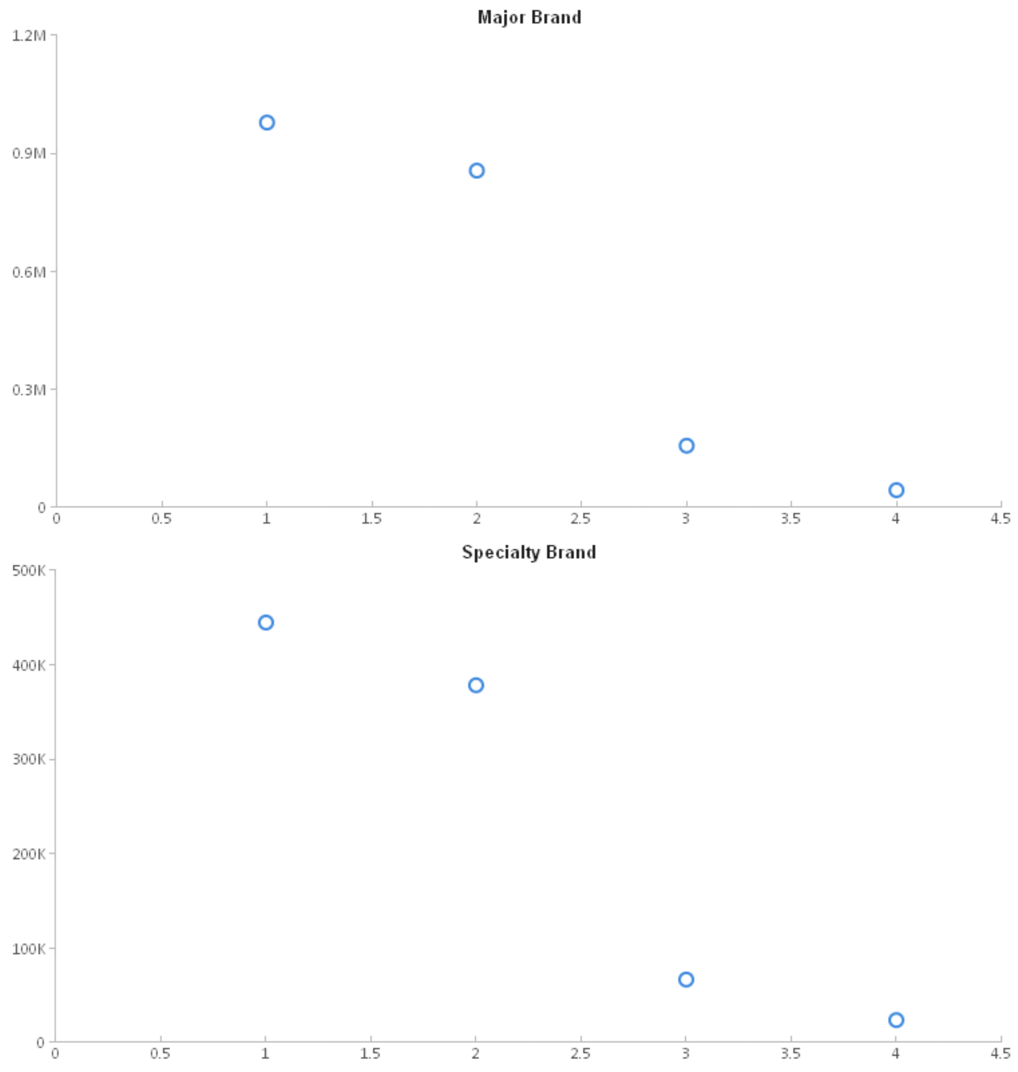
Type of Column or Parameter	Attribute Category
Measure field (any number).	y-axis Only one measure supported on the y-axis.
Sort field (measure or dimension). With SCATTERS, the sort field becomes the x-axis.	x-axis
High-level sort field.	page

Example: Converting a SCATTERS Chart Request to Chart Attribute Syntax

The following example generates multiple scatter charts using traditional syntax (LOOKGRAPH value is SCATTERS). The high-level sort field (BRANDTYPE) is the sort field that generates multiple charts. The QUANTITY_SOLD sort field represents the x-axis. The measure COGS_US is plotted on the y-axis:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM COGS_US
BY BRANDTYPE
BY QUANTITY_SOLD
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTERS
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

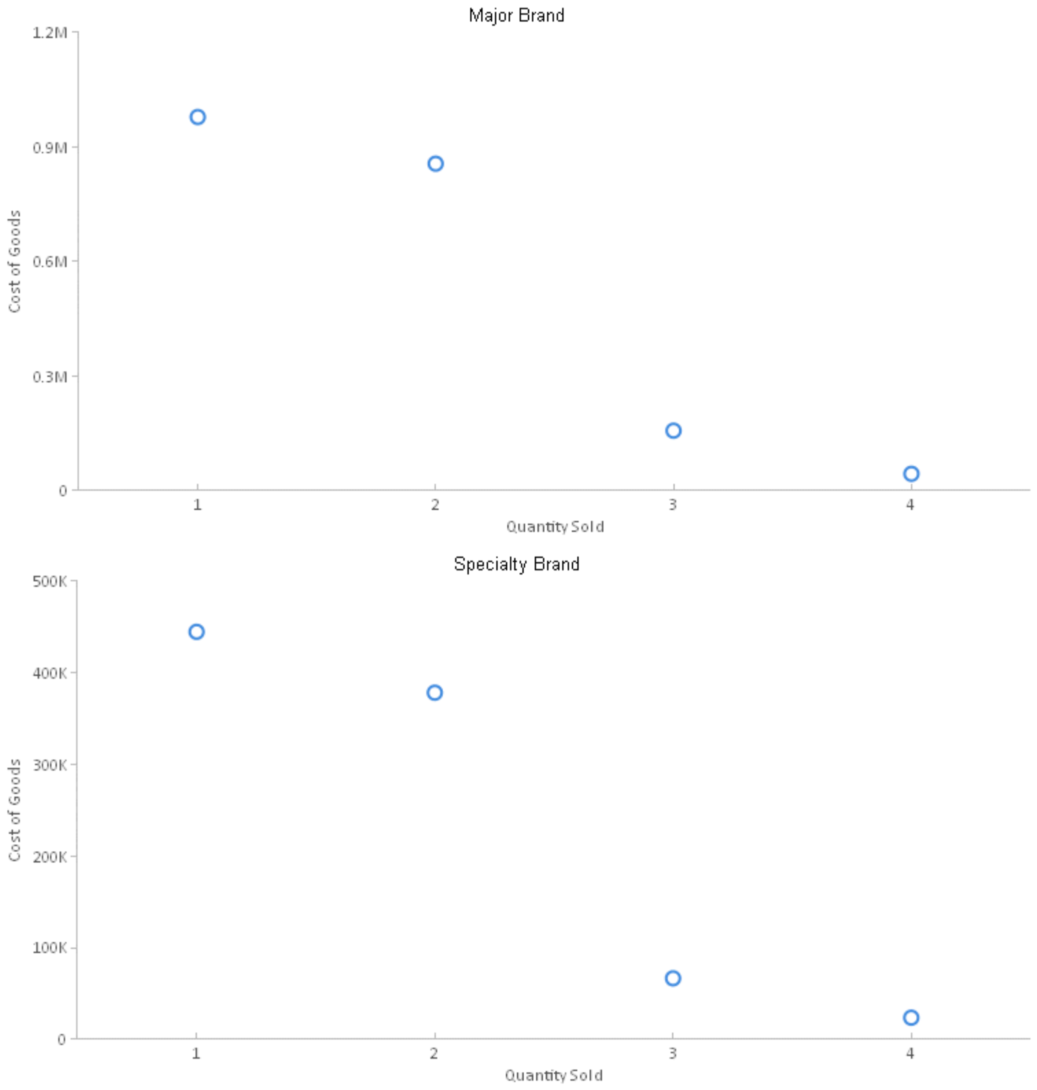
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is SCATTER. The measure (COGS_US) is assigned to the y-axis attribute category, the low-level sort field QUANTITY_SOLD is assigned to the x-axis attribute category, and the high-level sort field (BRANDTYPE) is assigned to the page attribute category:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM COGS_US
BY BRANDTYPE
BY QUANTITY_SOLD
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SCATTER
ON GRAPH SET STYLE *
type=data, column=cogs_us, bucket=y-axis,$
type=data, column=quantity_sold, bucket=x-axis,$
type=data, column=brandtype, bucket=page,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

The output is shown in the following image:



Converting Spectral Chart Requests to Heatmap Chart Requests

A heatmap chart is a matrix in which each cell is filled with a color that represents the value of the measure for that cell.

Traditional WebFOCUS chart syntax does not have a heatmap chart type. It has a spectral chart type, which is similar. However, spectral charts support multiple unrelated measures, while heatmaps support only a single measure. To convert a spectral chart with multiple measures, you have to issue separate heatmap chart requests for each measure.

Reference: LOOKGRAPH Conversions for Heatmap Charts

The following table lists the traditional LOOKGRAPH value and the new LOOKGRAPH value.

LOOKGRAPH Parameter	
SPECTRAL	HEATMAP

Reference: Attribute Category Assignments for Heatmap Charts

The following table lists the attribute category conversions for heatmap charts. The GRLEGEND and GRXAXIS parameters can be used interchangeably.

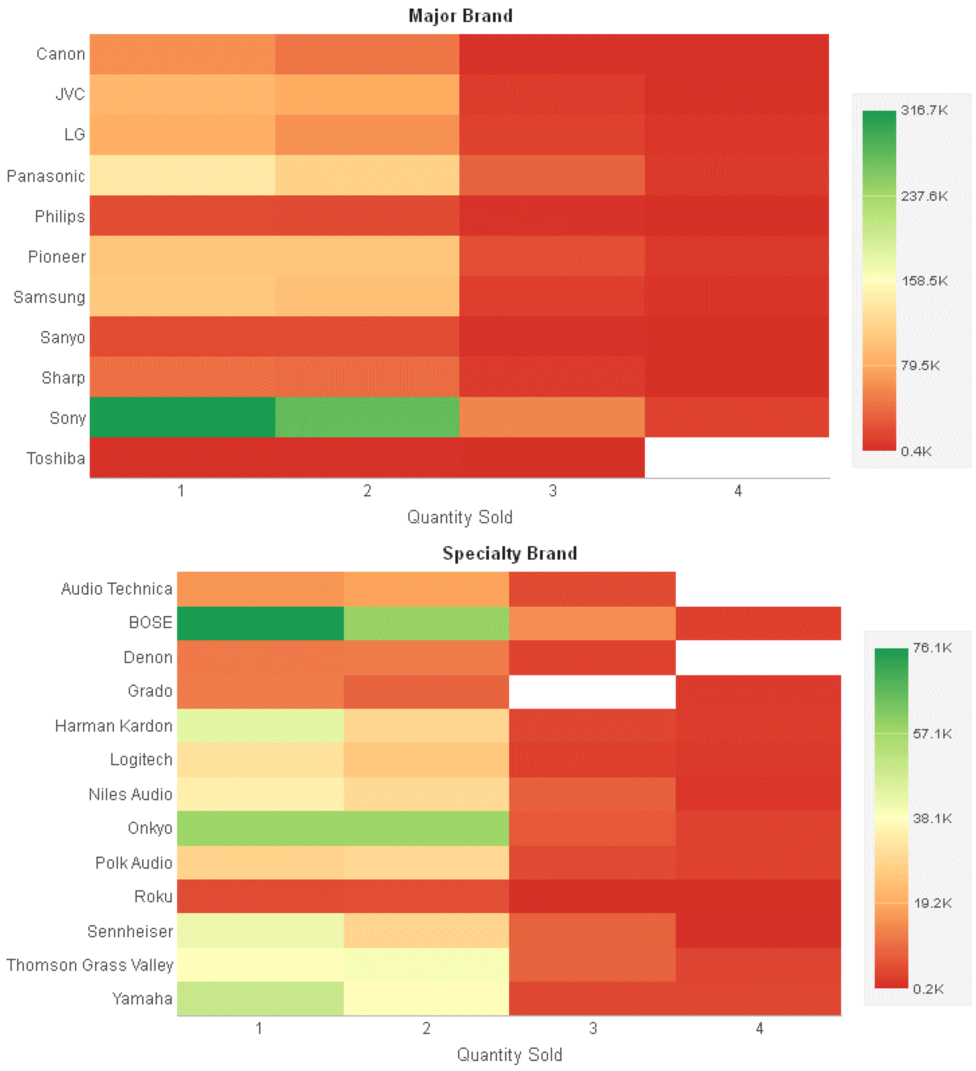
Type of Column or Parameter	**Attribute Category**
measure field	color
GRLEGEND sort field	y-axis
GRXAXIS (ACROSS) sort field	x-axis
GRMULTIGRAPH sort field	page

Example: Converting a Spectral Chart Request to a Heatmap Chart Request

The following example generates multiple spectral charts using traditional syntax (LOOKGRAPH value is SPECTRAL). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate spectral charts are generated for each value of BRANDTYPE. The BRAND sort field generates the y-axis, and the QUANTITY_SOLD sort field generates the x-axis. The measure (COGS_US) is used for the color of the rectangles. The VZERO parameter is set to match the default value for chart attribute syntax requests.

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
BY BRANDTYPE
BY BRAND
ACROSS QUANTITY_SOLD
ON GRAPH SET VZERO OFF
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH SPECTRAL
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

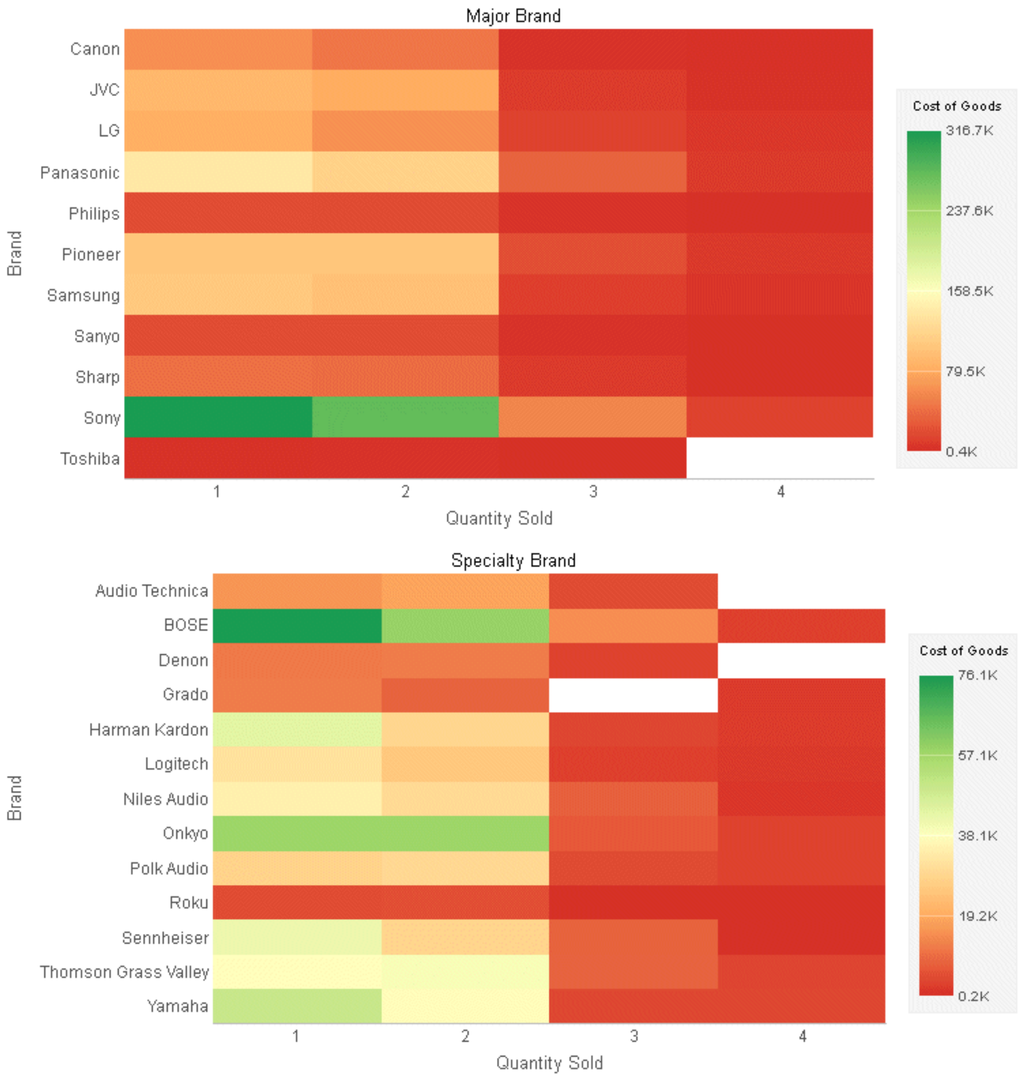
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is HEATMAP. The measure (COGS_US) is assigned to the color attribute category, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, the BRAND sort field is assigned to the y-axis category, and the QUANTITY_SOLD sort field is assigned to the x-axis attribute category. The ACROSS phrase is changed to BY:

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE "
SUM COGS_US
BY BRANDTYPE
BY BRAND
BY QUANTITY_SOLD
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET EMBEDHEADING ON
ON GRAPH SET LOOKGRAPH HEATMAP
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
type=data, column=brandtype, bucket=page,$
type=data, column=cogs_us, bucket=color,$
type=data, column=brand, bucket=y-axis,$
type=data, column=quantity_sold, bucket=x-axis,$
ENDSTYLE
END
```

The output is shown in the following image:



Converting Streamgraph Chart Requests to Chart Attribute Syntax

A streamgraph is a simplified version of a stacked area chart.

Reference: LOOKGRAPH Conversions for Streamgraph Charts

The following table lists the traditional LOOKGRAPH values and the new LOOKGRAPH values.

LOOKGRAPH Parameter	
STREAM	STREAM

Reference: Attribute Category Assignments Streamgraph Charts

The following table lists the attribute category conversions for streamgraph charts.

Type of Column or Parameter	**Attribute Category**
measure field	y-axis
sort field	x-axis

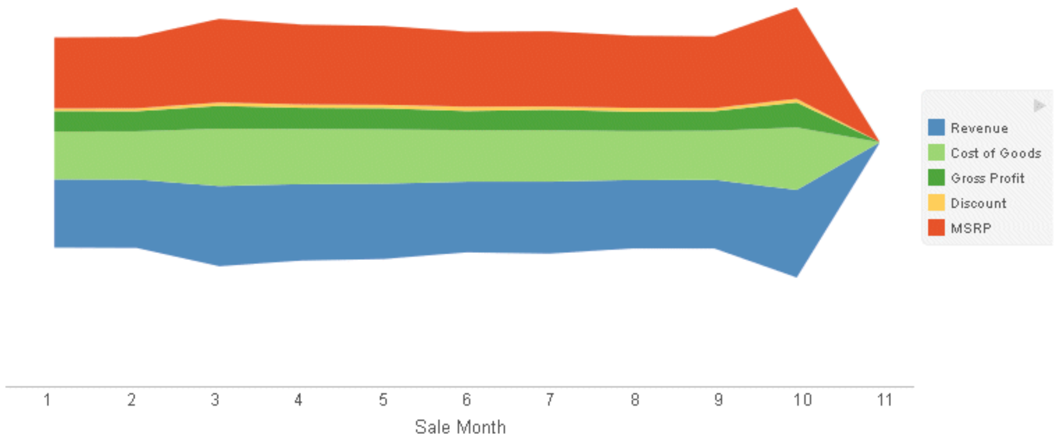
Note: Merge parameters are not supported with streamgraphs.

Example: Converting a Streamgraph Chart Request to Chart Attribute Syntax

The following example generates a streamgraph chart. The measures are REVENUE_US, COGS_US, GROSS_PROFIT_US, DISCOUNT_US, and MSRP_US. The sort field is TIME_MTH:

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US COGS_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH STREAM
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

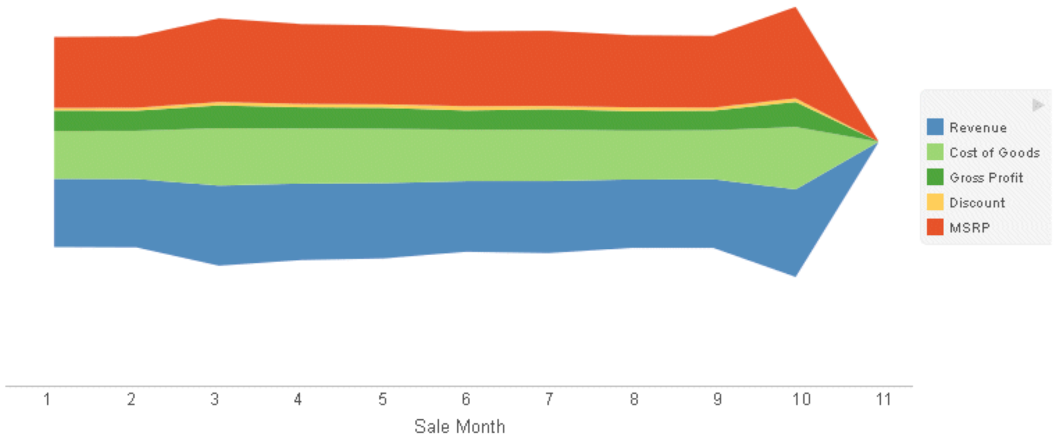
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The TIME_MTH sort field is assigned to the x-axis category, and the measures are assigned to the y-axis category.

```
GRAPH FILE WF_RETAIL_LITE
SUM REVENUE_US COGS_US GROSS_PROFIT_US DISCOUNT_US MSRP_US
BY TIME_MTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH STREAM
ON GRAPH SET STYLE *
type=data, column=revenue_us, bucket=y-axis,$
type=data, column=COGS_US, bucket=y-axis,$
type=data, column=GROSS_PROFIT_US, bucket=y-axis,$
type=data, column=DISCOUNT_US, bucket=y-axis,$
type=data, column=MSRP_US, bucket=y-axis,$
type=data, column=time_mth, bucket=x-axis,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

The output is shown in the following image.



Converting a Tagcloud Chart to Chart Attribute Syntax

A tagcloud chart is a visual representation of group labels. The size of each label is proportional to its data value. A second measure can be used to color the labels.

Reference: LOOKGRAPH Conversions for Tagcloud Charts

The following table lists the traditional LOOKGRAPH value and the new LOOKGRAPH value.

LOOKGRAPH Parameter	
TAGCLOUD	TAGCLOUD

Reference: Attribute Category Assignments for Tagcloud Charts

The following table lists the attribute category conversions for tagcloud charts. The GRLEGEND and GRXAXIS parameters can be used interchangeably.

Type of Column or Parameter	**Attribute Category**
first measure field	size

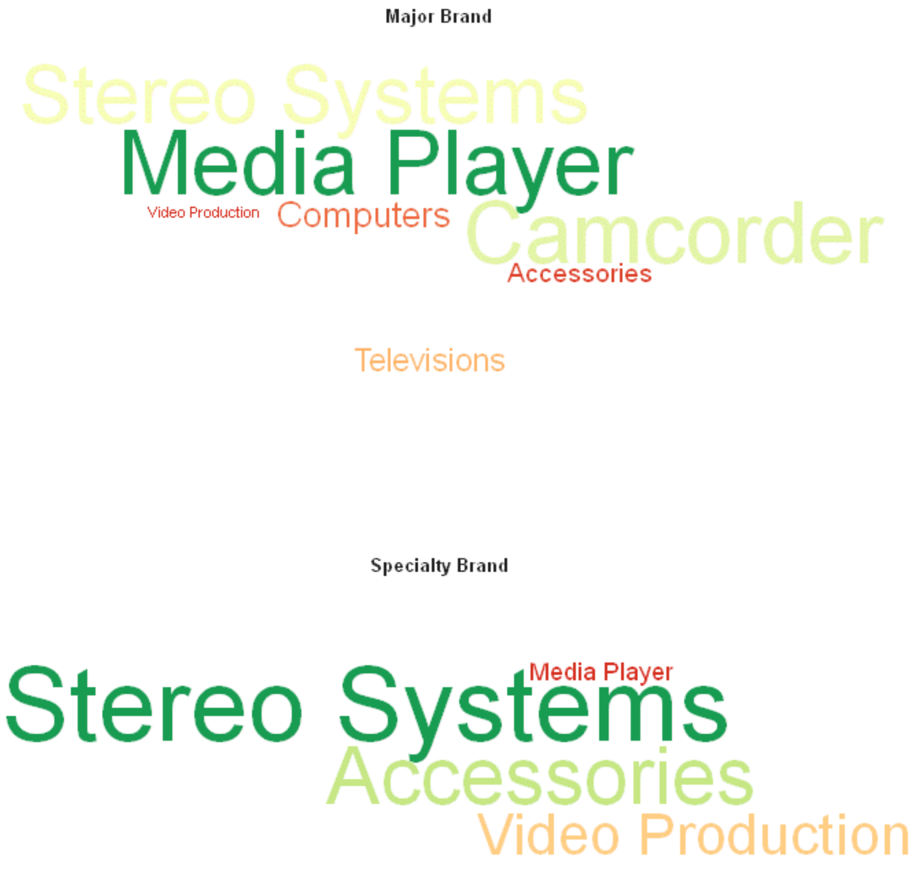
Type of Column or Parameter	Attribute Category
second measure field, if any	color
GRLEGEND or GRXAXIS sort field	detail
GRMULTIGRAPH sort field	page

Example: Converting a Tagcloud Chart Request to Chart Attribute Syntax

The following example generates multiple tagcloud charts using traditional syntax (LOOKGRAPH value is TAGCLOUD). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate tagcloud charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY sort field generates the labels. The first measure (GROSS_PROFIT_US) is used for the size of the labels, and the second measure (COGS_US) is used for the color of the labels:

```
GRAPH FILE WF_RETAIL_LITE
SUM GROSS_PROFIT_US COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

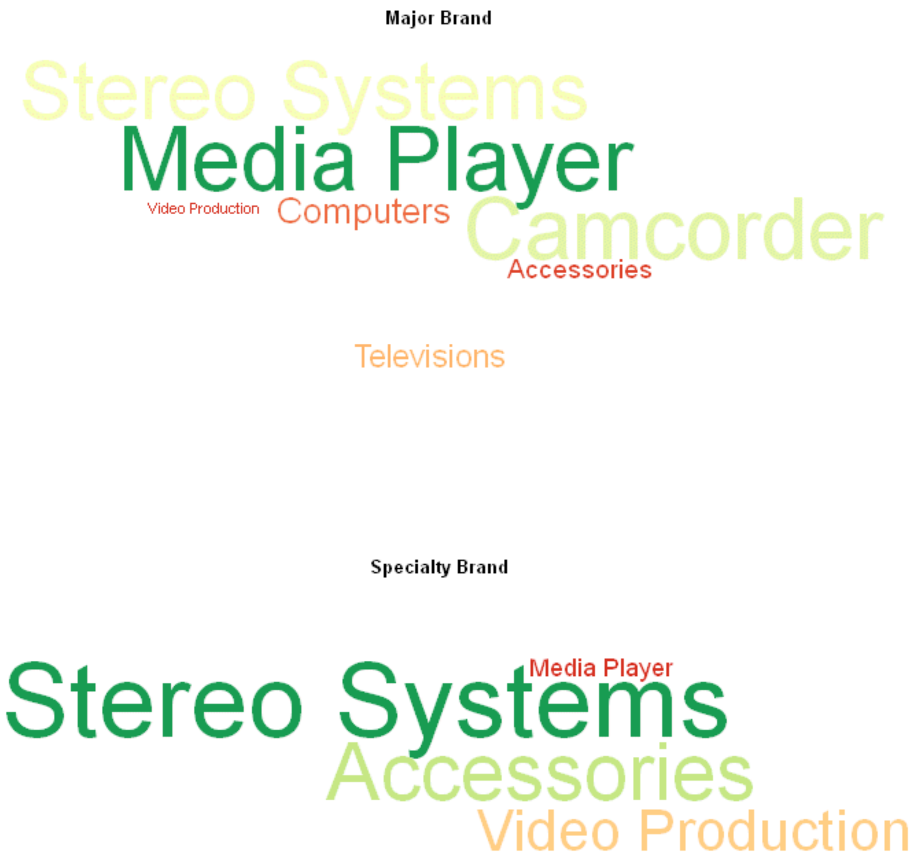
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is TAGCLOUD. The measures (GROSS_PROFIT_US and COGS_US) are assigned to the size and color attribute categories, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort field (PRODUCT_CATEGORY) is assigned to the detail attribute category.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
" <BRANDTYPE </1"
SUM GROSS_PROFIT_US COGS_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
ON GRAPH SET EMBEDHEADING ON
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TAGCLOUD
ON GRAPH SET STYLE *
type=data, column=brandtype, bucket=page,$
type=data, column=product_category, bucket=detail,$
type=data, column=gross_profit_us, bucket=size,$
type=data, column=cogs_us, bucket=color,$
type=heading, font=arial, style=bold,$
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

The output is shown in the following image:



Converting Treemap Chart Requests to Chart Attribute Syntax

Treemap charts show hierarchical data as a set of nested colored rectangles.

Reference: LOOKGRAPH Conversions for Treemap Charts

The following table lists the traditional LOOKGRAPH value and the new LOOKGRAPH value.

LOOKGRAPH Parameter	
TREEMAP	TREEMAP

Reference: Attribute Category Assignments for Treemap Charts

The following table lists the attribute category conversions for treemap charts. The GRLEGEND and GRXAXIS parameters can be used interchangeably.

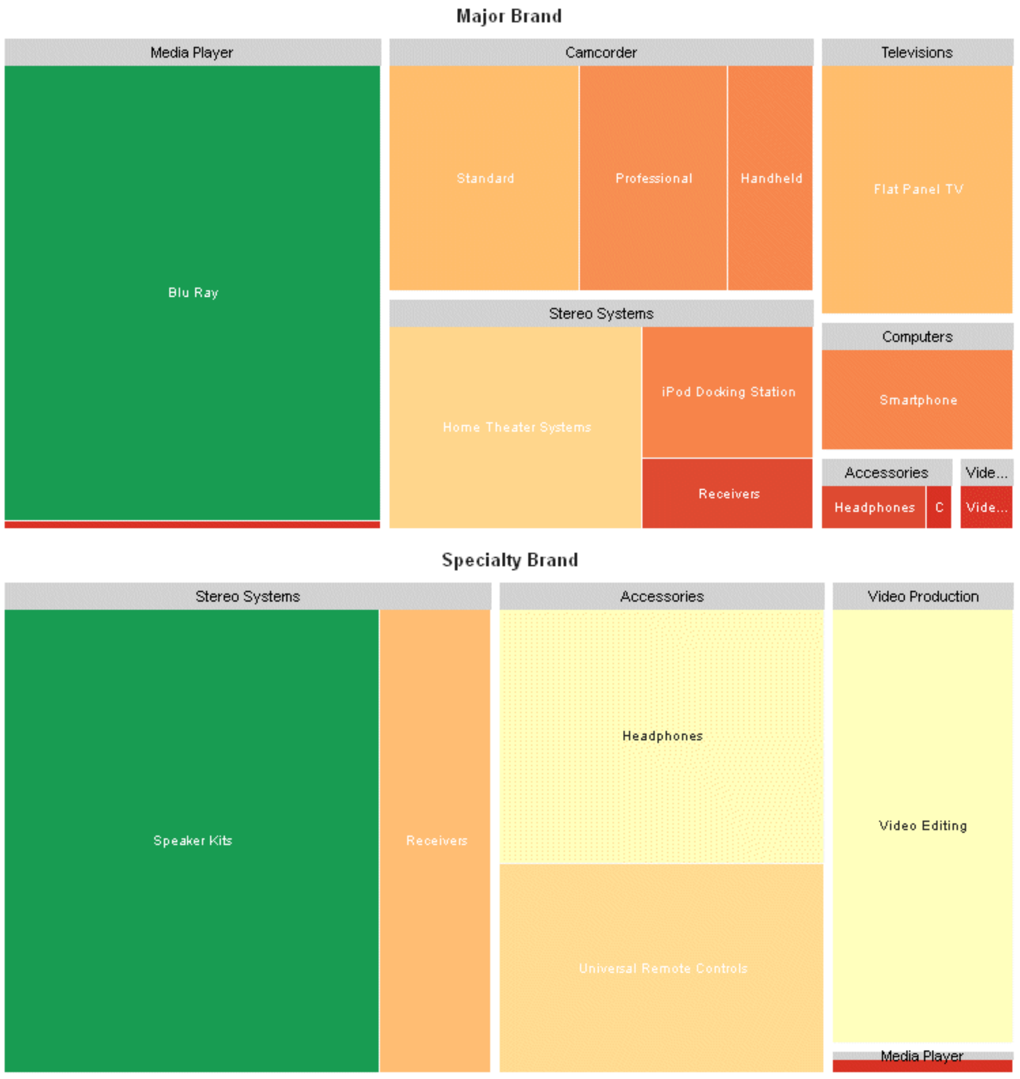
Type of Column or Parameter	Attribute Category
first measure field	size
second measure field, if any	color
GRLEGEND sort field	detail
GRXAXIS sort field	detail
GRMULTIGRAPH sort field	page

Example: Converting a Treemap Chart Request to Chart Attribute Syntax

The following example generates multiple treemap charts using traditional syntax (LOOKGRAPH value is TREEMAP). The high-level sort field (BRANDTYPE) is the GRMULTIGRAPH sort field. Separate treemap charts are generated for each value of BRANDTYPE. The PRODUCT_CATEGORY and PRODUCT_SUBCATEG sort fields generate the nested rectangles. The first measure (COGS_US) is used for the size of the rectangles, and the second measure (REVENUE_US) is used for the color of the rectangles:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US
REVENUE_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 1
ON GRAPH SET GRLEGEND 1
ON GRAPH SET GRXAXIS 1
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
ENDSTYLE
END
```

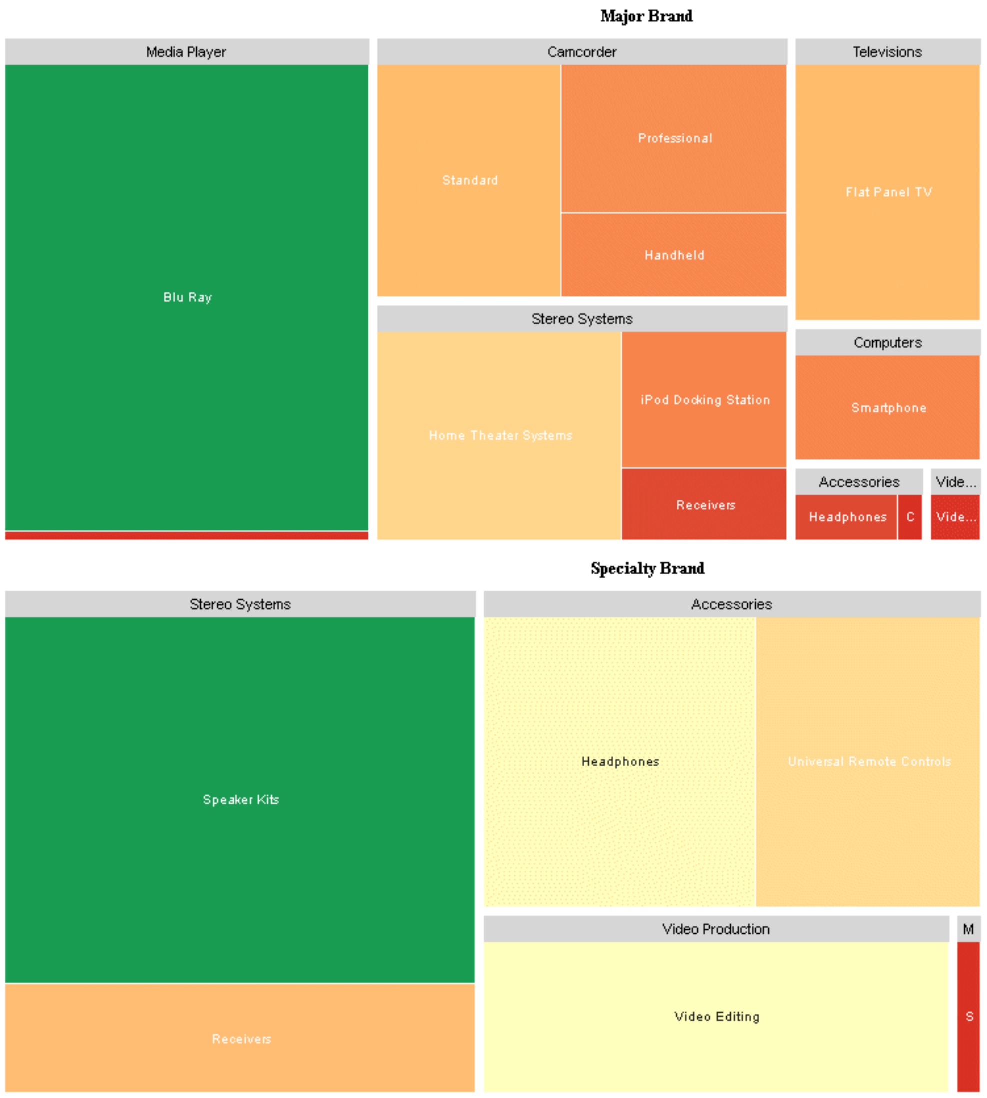
The output is shown in the following image:



The following is the same request converted to chart attribute syntax. The LOOKGRAPH value is TREEMAP. The measures (COGS_US and REVENUE_US) are assigned to the size and color attribute categories, the high-level sort field (BRANDTYPE) is assigned to the page attribute category, and the low-level sort fields (PRODUCT_CATEGORY and PRODUCT_SUBCATEG) are assigned to the detail attribute category.

```
GRAPH FILE WF_RETAIL_LITE
HEADING CENTER
"<BRANDTYPE"
SUM COGS_US
REVENUE_US
BY BRANDTYPE
BY PRODUCT_CATEGORY
BY PRODUCT_SUBCATEG
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH TREEMAP
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
TYPE=DATA, COLUMN=BRANDTYPE, BUCKET=PAGE,$
TYPE=DATA, COLUMN=COGS_US, BUCKET=size,$
TYPE=DATA, COLUMN=REVENUE_US, BUCKET = color,$
TYPE=DATA, COLUMN=PRODUCT_CATEGORY, BUCKET=DETAIL,$
TYPE=DATA, COLUMN=PRODUCT_SUBCATEG, BUCKET=DETAIL,$
TYPE=HEADING, STYLE=BOLD,$
*GRAPH_JS
"legend": {"visible": false}
*END
ENDSTYLE
END
```

The output is shown in the following image:



Converting Requests With Nested X-Axes to Chart Attribute Syntax

In the traditional WebFOCUS chart syntax, nested x-axes are generated when there are multiple sort fields and the GRXAXIS parameter is set to a number greater than 1. With chart attribute syntax, nested x-axes are generated when multiple hierarchical sort fields are assigned to the x-axis attribute category.

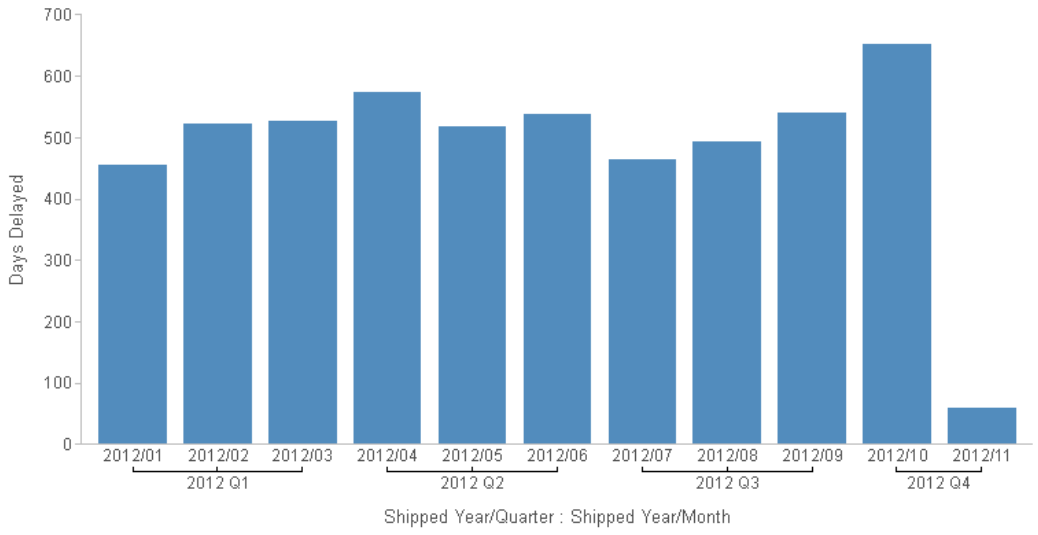
You can display the nested x-axes as concatenated field values by setting a concatenation string for the x-axis labels, as described in *Axis Properties* on page 325.

Example: Converting a Request With Nested X-Axes to Chart Attribute Syntax

The following request generates nested x-axes using the traditional WebFOCUS chart syntax.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY TIME_DATE_QTR
BY TIME_DATE_MONTH
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 2
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"xaxis": {"labels": {"nestingConcatSymbol": null}}
*END
ENDSTYLE
END
```

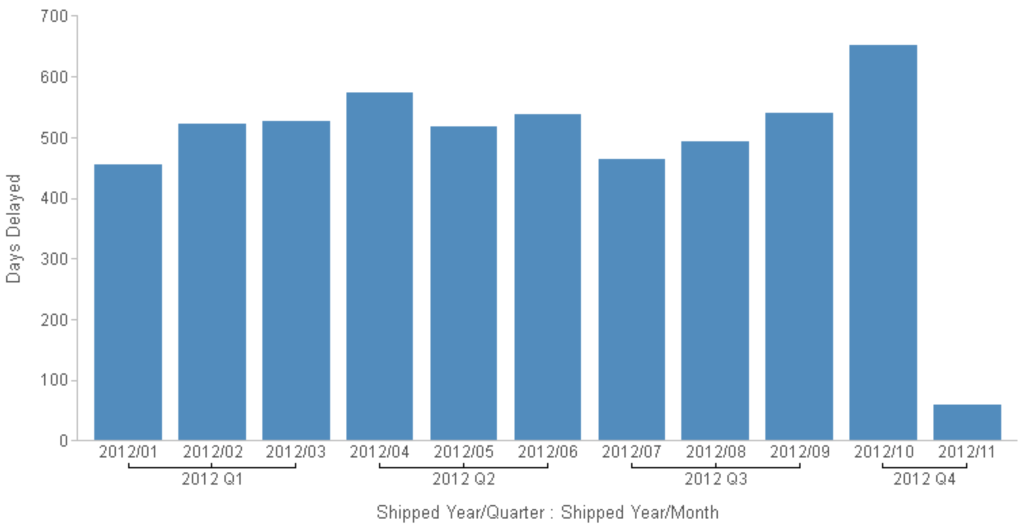
The output is shown in the following image.



The following version of the request assigns both sort fields to the x-axis attribute category.

```
GRAPH FILE WF_RETAIL_LITE
SUM DAYSDELAYED
BY TIME_DATE_QTR
BY TIME_DATE_MONTH
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET LOOKGRAPH BAR
ON GRAPH SET STYLE *
TYPE=DATA, COLUMN=DAYSDELAYED, BUCKET=y-axis, $
TYPE=DATA, COLUMN=TIME_DATE_QTR, BUCKET=x-axis, $
TYPE=DATA, COLUMN=TIME_DATE_MONTH, BUCKET=x-axis, $
INCLUDE=IBFS:/FILE/IBI_HTML_DIR/javaassist/intl/EN/combine_templates/ENWarm.sty,$
*GRAPH_JS
"xaxis": {"labels": {"nestingConcatSymbol": null}}
*END
ENDSTYLE
END
```

The output is shown in the following image.



Appendix C

Cumulative List of Changes to JSON Properties

This topic lists the properties that have been changed or deprecated for all releases up to and including the current release.

In this appendix:

Changes to JSON Properties in Release 8.2

The following changes to JSON properties took place in WebFOCUS Release 8.2.

- The data text labels properties (dataLabels) have been changed from chart-wide properties to series-specific properties. The dataLabels:displayMode and dataLabels:callback properties have been replaced by the content property. For information, see *Series-Specific Properties* on page 413.
- The series-specific showDataValues property has been deprecated.
- The chart-wide property axisAutoLayout has been deprecated.
- In Release 8.2 Version 02, the line property for global trendlines has been replaced by the lineStyle property.
- In Release 8.2 Version 02, the legend position properties *free* and *xy* have been deprecated. The legend position can now be a JSON object. For information, see *Controlling the Position of the Legend* on page 299.
- In Release 8.2 Version 05, the colorScale:colorBands properties have been deprecated and replaced by adding start and stop values to colorScale:colors for all color modes.

Changes to JSON Properties in Release 8.1

The following changes to JSON properties took place in WebFOCUS Release 8.1.

- xaxisNumeric and xaxisOrdinal have been replaced with xaxis.
- zaxisOrdinal has been replaced with zaxis.
- fillMode has been replaced with fillEffect.

- pieProperties:feelerLine has been replaced with dataLabels:feelerLine.
- yaxis:colorScale and heatmapProperties:dataColors have been replaced with colorScale.

Index

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at *Sarah_Buccellato@ibi.com.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

WebFOCUS

Creating HTML5 Charts With WebFOCUS Language

Release 8205

DN4501532.0219

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898