

WebFOCUS

WebFOCUS Graphics
Release 8.1 Version 03

Active Technologies, EDA, EDA/SQL, FIDEL, FOCUS, Information Builders, the Information Builders logo, iWay, iWay Software, Parlay, PC/FOCUS, RStat, Table Talk, Web390, WebFOCUS, WebFOCUS Active Technologies, and WebFOCUS Magnify are registered trademarks, and DataMigrator and Hyperstage are trademarks of Information Builders, Inc.

Adobe, the Adobe logo, Acrobat, Adobe Reader, Flash, Adobe Flash Builder, Flex, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2015 , by Information Builders, Inc. and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Contents

Preface	41
Documentation Conventions	41
Related Publications	42
Customer Support	42
Information You Should Have	43
User Feedback	44
Information Builders Consulting and Training	44
1. Graphing Basics	45
What Is Scalable Vector Graphics?	45
Viewing Scalable Vector Graphics	46
Portable Network Graphics	47
Virtual Coordinate System	47
A Look at Coordinate Systems: Cartesian and Virtual.	47
How Does WebFOCUS Graphics Use the Virtual Coordinate System?.....	53
Implementing API Calls	60
Syntax: How to Implement an API Call.	60
2. Using WebFOCUS Graphics	63
How Properties and Methods Work	64
The GraphType Property	64
GraphType Values.....	64
GraphType Methods.....	66
How Graphs Are Imaged	66
Graph Sizing.....	67
3D Graphs.....	67
Single Axis/Vertical Graphs.....	68
Single Axis/Horizontal Graphs.....	68
Dual Axes and Bi-Polar Vertical Graphs.....	69
Dual Axes and Bi-Polar Horizontal Graphs.....	69
Pie Graphs.....	70
Scatter Graphs.....	71
Stock Graphs.....	71
Histograms.....	72
Spectral Maps.....	73

Bubble Graphs	73
Chart Layout Manager	74
Setting the Color of the Page Surrounding the Graph	76
Syntax: How to Set the Color of the Page Surrounding the Graph.	76
Graph-Wide Parameters	77
Graph Frame Attributes	78
Axis Attributes	79
Axis Properties.	79
Ordinal Axis Properties.	82
Ordinal Axis Methods.	82
Numeric Axis Properties.	82
Numeric Axis Methods.	83
Axis Independent Methods.	84
Tick Mark Properties and Methods.	85
Properties for Dual Axes Graphs.	86
Autofitting Axis Text.	87
Grid Lines	87
Ordinal Axis Grid Lines.	87
Numeric Axis Grid Lines.	88
3D Grid Lines.	89
Axis Independent Methods for Grids.	90
Curve Fit Lines	90
Java 2D Line Objects	91
Risers and Markers	92
Riser and Marker Properties.	92
Riser and Marker Methods.	93
Exceptional Risers.	96
Riser Automatic Color Rendering/Styling	96
Syntax: How to Implement Automatic Repeated Riser Shading.	96
Syntax: How to Implement Automatic Gradient Riser Coloring.	99
Legends	101
Legend Properties.	101
Legend Methods.	102
Setting the Legend Position.	103
Syntax: How to Set the Legend Position.	103

Labels and Titles	117
Graph-Wide Titles.....	117
Ordinal Axis Labels and Titles.....	118
Numeric Axis Labels and Titles.....	119
Label and Title Methods.....	119
Formatting Date and Time Values for Y-Axis Labels	120
Using Nested/Multi-Dimensional Labels.....	124
Data Text	125
Customizing Data Text	128
get/setCustomDataText().....	129
Formatting Numbers and Numeric Labels	130
Preset Number Formats.....	130
Number Format Patterns.....	133
Special Formatting Using Macros.....	136
Formatting Text Objects	136
Java 2D Text Objects	137
Autofitting	138
Graph-Specific Properties and Methods	139
3D Cube Characteristics.....	139
Pie Graph Properties and Methods.....	141
Stock Graph Properties and Methods.....	142
Bubble Graph Properties and Methods.....	143
Funnel Graph Properties and Methods.....	144
Gauge Graph Properties & Methods.....	146
Syntax: How to Set the Title Position in a Gauge Chart.....	147
Syntax: How to Control the Display of the Value in the Gauge Chart.....	148
Syntax: How to Control the Display of the Minimum Value on the Gauge Scale....	149
Syntax: How to Control the Display of the Maximum Value on the Gauge Scale....	150
Multi-Y Axes Graph Properties and Methods.....	152
Syntax: How to Set the Axis Offset From the Base of the Frame	153
Pareto Graph Properties and Methods.....	154
Waterfall Graph Properties and Methods.....	154
Colors and Shading	155
Defining Colors.....	158
Syntax: How to Define a Color for a Graph Object.....	158

Reference: Usage Notes for Color Transparency.	160
Textures and Gradients	164
Panning, Rotating, and Scaling	164
Graph Editing Characteristics	165
Annotations	166
Object IDs.	166
Enable Annotation.	167
Define Annotation Label/String.	167
Set Annotation Text Color.	167
Set Annotation Font Size.	167
Set Annotation Location.	167
Enable Annotation Box.	168
Set Annotation Box Color.	168
Enable/Disable Annotation Box Border Color.	168
Initial/Default Annotation Box Settings.	169
Maximum Number of Annotations.	170
Drill-Down and URLs	170
URL Methods.	170
Example Drill-Down.	171
Multiple Drill-Down From Graph	171
Syntax: How to Create Multiple Drill-Down Links.	171
Alternate Formatting	174
Alternate Formatting Objects.	174
Alternate Formatting Methods.	176
get/setAltFmtFrameNumColors().	176
get/setAltFmtFrameSeparator().	176
Applying Colors and Labels.	177
Beveled Borders	178
Object IDs.	178
Enable/Disable Beveled Borders.	178
get/setBorderType().	179
get/set...Insets().	180
get/setCornerArcWidth/Height().	181
Annotation Example.	183
Reference Lines	183

Reference: APIs Related to Reference Lines.....	184
Reference: Properties Supported by Reference Lines.....	184
APIs Related to the Reference Line Legend.....	186
3. Using Data in a Graph	187
Graph Titles and Labels	187
Data Scrolling and Zooming	188
Data Scrolling.....	189
Data Zooming.....	191
How Missing Data Is Handled	193
Data Requirements for Different Graph Types	193
Data in a 3D Graph.....	196
Data in a Bar/Line/Area Graph.....	197
Data in a Pie Graph.....	198
Data in an XY Scatter Graph.....	199
Data in an XY Scatter Graph With Labels.....	200
Data in a Polar Graph.....	201
Data in a Radar Graph.....	202
Data in a Spectral Map.....	203
Data in a Bubble Graph.....	204
Data in a Funnel Graph.....	205
Data in a Multi-Y Graph.....	205
Data in a Pareto Graph.....	208
Data in a Waterfall Graph.....	208
Data in a Stock Graph.....	211
GraphType 70.....	211
GraphType 72.....	212
GraphTypes 73...75.....	213
GraphTypes 76...78.....	214
GraphTypes 79...81.....	215
GraphType 82.....	218
GraphType 84.....	220
Time Scale Axis Graphs.....	221
4. Properties	229
Properties for Graphs	229

AdjustAutoScaleForRefLines	239
AlignMultiYBars	241
AltFmtFrameSeparator	243
AltFmtFrameNumColors	245
AltFmtMatchAxisLabelColor	246
AutoHideDataTextOnSmallRisers	248
Autoshading	250
AutoshadingWalls	253
BiDirectional	254
BoxPlotType	256
Cascade	258
ColorMode	260
ConnectLineMarkers	262
ConnectParetoBars	263
ConnectScatterMarkers	265
CubeFocusFactor	267
CubeIsometricProjection	269
CubeLightSourceX	271
CubeLightSourceY	272
CubeLightSourceZ	274
CubePanX	275
CubePanY	277
CubeProportional	278
CubeRiserInterpolation	280
CubeSizeX	280
CubeSizeY	281
CubeSizeZ	283
CubeSquareRisers	284
CubeTranslationX	286
CubeTranslationY	288
CubeTranslationZ	290
CubeViewerX	292
CubeViewerY	293
CubeViewerZ	294
CubeWallThickX	296

CubeWallThickY	298
CubeWallThickZ	300
CubeZoomFactor	302
CurveFitEquationDisplay	303
CurveFitHighOrderFirst	305
CurveFitNumSegments	306
DataItemsAlongSeries	306
DataLineThickness	308
DataScrollerPresenceGroups	310
DataScrollerPresenceSeries	311
DataSortType	312
DataTextAngleDefault	314
DataTextDisplay	316
DataTextFormat	318
DataTextFormatPattern	320
DataTextPosition	322
DataTextRadiusDefault	324
DataTextStackedTotalOnTop()	325
Format Option for the Stack Total Value	327
Syntax: How to Format the Stack Total Value.	327
DataTextTemplateBubble	329
DataTextTemplateHiLo	330
DataTextTemplateScatter	332
DepthAngle	333
DepthRadius	334
Display3DFloor	336
Display3DLeftWall	338
Display3DRightWall	340
DisplayBarAsPictograph	342
DisplayFunnelGroup	343
DisplayParetoSeries	344
DocumentHeightInTwips	346
DocumentWidthInTwips	346
DrawAreaMarkers	347
DrawCompleteAreaRiser	348

DualAxisLineDisplay	350
DualAxisSplitPosition	351
ExactColorByHeight	352
ExtendToFrameEdge	353
FillMissingData	354
FootnoteDisplay	356
FootnoteString	357
ForceSeriesAbsolute	359
FrameAutoShade	360
FrameDisplay	362
FunnelLabelValueLocation	363
GaugeBackgroundStyle	366
GaugeColorCenterByQuality	367
GaugeDescending	369
GaugeIgnoreBackgroundStyle	370
GaugeMajorAutoStep	370
GaugeMajorTickRelativeTickLength	371
GaugeMinorAutoStep	373
GaugeMinorTickRelativeTickLength	373
GaugeNeedleStyle	374
GaugeRelativeInnerRadius	375
GaugeRelativeThickness	377
GaugeSizeMode	377
GaugeStartAngle	379
GaugeStopAngle	380
GraphType	380
Grid3DFloorDisplayX	381
Grid3DFloorDisplayZ	382
Grid3DLeftWallDisplayY	384
Grid3DLeftWallDisplayZ	385
Grid3DRightWallDisplayX	387
Grid3DRightWallDisplayY	388
Grid3DRiserDisplayX	390
Grid3DRiserDisplayY	391
Grid3DRiserDisplayZ	393

HiliteBarResize	394
IgnoreTickSkip	395
InterpretAsHLOC	396
LegendAutomatic	397
LegendDisplay	398
LegendMarkerPosition	400
LegendMarkersPerRow	402
LegendOrient	403
LegendReverse	404
LegendScroll	405
LegendSeriesCount	406
LegendSeriesStart	407
MajorGridDrawEveryCount	408
MajorGridTickInterval	409
MinorGridTickInterval	410
MarkerDisplay	411
MarkerSizeDefault	412
MoveAxisTitles	414
NestedLabels	415
NullLegendMarkerShapesAsSquares	416
NumMissingDataSegments	418
NumPieArcSegmentsPerPixel	419
OtherPercentage	420
OtherSeries	421
OutputSVG	422
OverDrawSkip	423
O1AxisLineDisplay	423
O1AxisSide	425
O1ExcludeMaxLabel	428
O1ExcludeMinLabel	429
O1LabelAutoAngle	430
O1LabelAutoSkip	432
O1LabelDisplay	433
O1LabelMargin	434
O1LabelRotate	435

O1LabelSkipBegin	436
O1LabelSkipCount	438
O1LabelStagger	439
O1LabelWrap	441
O1MajorGridDisplay	443
O1MajorGridStyle	444
O1MajorTickDisplay	445
O1MajorTickStyle	446
O1MinorGridCount	446
O1MinorGridDisplay	447
O1MinorGridStyle	449
O1MinorTickCount	450
O1MinorTickDisplay	451
O1MinorTickStyle	452
O1TitleDisplay	452
O1TitleString	454
O2ExcludeMaxLabel	454
O2ExcludeMinLabel	456
O2LabelAutoSkip	457
O2LabelDisplay	458
O2LabelRotate	459
O2LabelSkipBegin	461
O2LabelSkipCount	463
O2LabelWrap	463
O2TitleDisplay	464
O2TitleString	466
ParetoDisplayThreshold	466
ParetoY2MajorGridStep	469
ParetoY2MajorTickStep	471
PieBarLabelDisplay	471
PieBarLabelString	473
PieBarSeries	473
PieDepth	474
PieFeelerTextDisplay	477
PieFeelerTextFormat	478

PieFeelerTextFormatPattern	479
PieLabelDisplay	481
PieRingSize	483
PieRingTotalDisplay	484
PieRingTotalFormat	486
PieRingTotalFormatPattern	487
PieRotate	488
PieSorting	491
PiesPerRow	492
PieTilt	493
Place	496
PrimitiveAntialiasing	497
QuadrantLineCountX	499
QuadrantLineCountY	500
ReferencePercentageMode	501
ReferenceTextPosition	502
RemoveDuplicateDataText	502
RepositionDataTextCollision	504
RepositionStackedDataTextCollision	505
ReshapeEnable	507
ResizeBarMode	508
ReverseGroups	509
ReverseSeries	510
Riser3DThicknessY	512
RiserBarGroupSpacing	513
RiserBorderMode	514
RiserWidth	516
ScaleFromZero	517
ScrollOffsetGroup	519
ScrollOffsetSeries	520
SelectionEnable	522
SelectionEnableMove	523
SeriesAreRows	523
SeriesDefaultBorderColor	524
SeriesDefaultTransparentBorderColor	526

SeriesLineWidthDefault	527
SeriesLooping	528
ShadowXOffsetDefault	530
ShadowYOffsetDefault	530
SmoothLines	530
SquareMarkers	532
StackedDataValueSum	533
Stock52WeekHighDisplay	535
Stock52WeekHighValue	536
Stock52WeekLowDisplay	538
Stock52WeekLowValue	539
StockCandleForVolumeColor	540
StockCloseSplitDisplay	543
StockCloseTicksDisplay	544
StockMovingAverageDisplay	545
StockOpenTicksDisplay	546
StockTickLength	547
SubtitleDisplay	549
SubtitleString	550
TextAntialiasing	551
TitleDisplay	553
TitleString	555
ToolTipDelay	556
ToolTipDisplay	556
ToolTipMode	556
ToolTipTimeoutEnabled	558
TransparentMarkers	558
TransparentMarkerSize	559
UseDefaultBubbleMarker	560
UseOffScreen	561
UseSeriesBorderDefaults	562
UseSeriesShapes	562
UseTimeScaleAxis	564
ViewableGroups	566
ViewableGroupsStock	567

ViewableSeries	568
ViewableSeriesStock	569
Viewing3DAnglePreset	571
WaterfallLastGroupTotal	572
WaterfallStackColorMode	574
X1AxisDescending	575
X1AxisLineDisplay	576
X1AxisSide	578
X1ExcludeMaxLabel	579
X1ExcludeMinLabel	581
X1LabelDisplay	582
X1LabelFormat	584
X1LabelFormatPattern	585
X1LabelRotate	586
X1LabelStagger	588
X1LogScale	589
X1MajorGridDisplay	591
X1MajorGridStep	592
X1MajorGridStepAuto	594
X1MajorGridStyle	596
X1MajorTickDisplay	597
X1MajorTickStyle	598
X1MinorGridDisplay	598
X1MinorGridStep	599
X1MinorGridStepAuto	601
X1MinorGridStyle	603
X1MinorTickDisplay	604
X1MinorTickStyle	605
X1MustIncludeZero	605
X1OffScaleDisplay	606
X1ScaleMax	608
X1ScaleMaxAuto	610
X1ScaleMin	611
X1ScaleMinAuto	613
X1TitleDisplay	614

X1TitleString	615
X1ZeroLineDisplay	616
Y1AxisDescending	617
Y1AxisLineDisplay	618
Y1AxisSide	619
Y1ExcludeMaxLabel	621
Y1ExcludeMinLabel	622
Y1LabelDisplay	624
Y1LabelFormat	625
Y1LabelFormatPattern	626
Y1LabelRotate	627
Y1LabelStagger	628
Y1LogScale	629
Y1MajorGridDisplay	630
Y1MajorGridStep	632
Y1MajorGridStepAuto	633
Y1MajorGridStyle	634
Y1MajorTickDisplay	636
Y1MajorTickStyle	636
Y1MinorGridDisplay	636
Y1MinorGridStepAuto	638
Y1MinorGridStyle	639
Y1MinorTickDisplay	640
Y1MinorTickStepAuto	640
Y1MinorTickStyle	640
Y1MustIncludeZero	641
Y1OffScaleDisplay	642
Y1ScaleMax	643
Y1ScaleMaxAuto	645
Y1ScaleMin	646
Y1ScaleMinAuto	648
Y1TitleDisplay	649
Y1TitleString	650
Y1ZeroLineDisplay	651
Y2AxisDescending	653

Y2AxisLineDisplay	654
Y2AxisSide	655
Y2ExcludeMaxLabel	657
Y2ExcludeMinLabel	658
Y2LabelDisplay	660
Y2LabelFormat	661
Y2LabelFormatPattern	662
Y2LabelRotate	663
Y2LabelStagger	664
Y2LogScale	666
Y2MajorGridDisplay	667
Y2MajorGridStep	668
Y2MajorGridStepAuto	670
Y2MajorGridStyle	671
Y2MinorGridDisplay	673
Y2MinorGridStep	675
Y2MinorGridStepAuto	676
Y2MinorGridStyle	677
Y2MustIncludeZero	679
Y2OffScaleDisplay	680
Y2ScaleMax	682
Y2ScaleMaxAuto	683
Y2ScaleMin	685
Y2ScaleMinAuto	686
Y2TitleDisplay	688
Y2TitleString	689
Y2ZeroLineDisplay	690
Y3AxisDescending	691
Y3LogScale	693
Y3ZeroLineDisplay	695
Y4AxisDescending	696
Y4LogScale	698
Y4ZeroLineDisplay	700
Y5AxisDescending	702
Y5LogScale	704

Y5ZeroLineDisplay	706
ZeroValueDataTextDisplay	707
ZeroValueDataTextStackedTotalOnTopDisplay	709
ZoomDirection	711
ZoomPercentage	713
5. Methods	715
get Methods	715
getAutoSkip()	715
getAxisAssignment()	716
getAxisDescending()	717
getAxisOffset()	718
getAxisSide()	720
getAxisSize()	721
getBorder()	722
getBorderColor()	723
getBorderType()	724
getBorderWidth()	724
getChartFromFTP()	724
getCornerArcWidth/Height()	725
getCurveFitPolynomialOrder()	726
getCurveFitType()	726
getCustomDataText()	727
getCustomDataTextFormat()	728
getCustomDataTextFormatPattern()	729
getDataAsDouble()	731
getDataAsString()	731
getDataCoord()	732
getDataLabel()	733
getDataTextAngle()	733
getDataTextRadius()	734
getDataValue()	735
getDisplay()	736
getDisplayOffScale()	737
getDocumentHeightInPoints()	738

getDocumentWidthInPoints().	739
getDrawTicksPerpendicular().	739
getEffectiveColorMode().	739
getEffectiveGroupOffset().	740
getEffectiveSeriesOffset().	741
getEffectiveViewableGroups().	742
getEffectiveViewableSeries().	743
getExcludeMaxLabel().	744
getExcludeMinLabel().	745
getFillColor().	746
getFillType().	747
getFontName().	748
getFontSize().	749
getFontSizeAbsolute().	750
getFontSizeInPoints().	751
getFontSizeVC().	751
getFontStyle().	752
getFrameDepthOffset().	753
getFrameRect().	754
getGradientDirection().	755
getGradientNumPins().	755
getGradientPinLeftColor().	756
getGradientPinPosition().	756
getGradientPinRightColor().	756
getGridCount().	757
getGridStep().	758
getGridStepAuto().	759
getGridStyle().	760
getGroupLabel().	761
getHasGraphics2D().	762
getIgnoreSeries().	762
getInsets().	763
get...inset().	763
getLabelStagger().	764
getLegendMinWidth().	765

getLegendRect().	765
getLevelDateFormat().	766
getLevelFillColor().	767
getLevelHeight().	767
getLevelInterval().	767
getLevelTextFormat().	768
getLineBasicStrokeType().	768
getLineCap().	769
getLineDash().	770
getLineDashFloat().	770
getLineDashPhase().	770
getLineJoin().	771
getLineMiterLimit().	771
getLineWidth().	771
getLogScale().	772
getLogScaleBase().	773
getMarkerShape().	774
getMarkerSize().	776
getMarkerTemplate().	777
getMaxNumAnnotations().	778
getMillisFromDate().	778
getMinimumSize().	779
getMinNumGroups().	779
getMinNumSeries().	780
getNonZeroBaseline().	781
getNonzeroBaselineValue().	782
getNumNonIgnoredSeries().	782
getNumColumnsInGroup().	783
getNumGroups().	784
getNumSeries().	785
getNumTotalGroups().	786
getNumTotalSeries().	787
getO1MajorGridColorBand1().	787
getO1MajorGridColorBand2().	788
getPieSliceDelete().	789

getPieSliceDetach().	790
getPlaceAlign().	791
getPlaceMargin().	791
getPlaceMaxNumWordWrapLines().	792
getPlaceMinFontSize().	792
getPlaceRect().	793
getPlaceResize().	793
getPlaceSkip().	794
getPlaceTruncateCount().	794
getPlaceWordWrap().	795
getQuadrantLineValueX().	795
getQuadrantLineValueY().	796
getRect().	797
getReferenceLineValue().	798
getReferenceLineXY().	799
getReferenceMarker().	800
getReferenceRect()	801
getScaleMax().	803
getScaleMaxAuto().	804
getScaleMin().	805
getScaleMinAuto().	806
getScaleMustIncludeZero().	807
getSeriesBorderColor().	807
getSeriesFillColor().	808
getSeriesLabel().	809
getSeriesType().	810
getShadowColor().	811
getShadowDisplay().	812
getShadowFillType().	812
getShadowGradientEndColor().	812
getShadowGradientStartColor().	813
getShadowTextureURL().	813
getShadowXOffset().	814
getShadowYOffset().	814
getSkipBegin().	814

getSkipCount().	815
getTextBackgroundColor().	815
getTextFormatPattern().	815
getTextFormatPreset().	816
getTextJustHoriz().	817
getTextJustVert().	818
getTextRotation().	818
getTextStrikethrough().	818
getTextString().	819
getTexture().	819
getTextureBGColor().	820
getTextureDisplayMode().	821
getTextureFlipMode().	821
getTextWrap().	822
getTickCount().	822
getTickLength().	822
getTickStep().	823
getTickStepAuto().	823
getTickStyle().	823
getTickWidth().	824
getTransparentBorderColor().	824
getTransparentFillColor().	825
getUseTextBackgroundColor().	825
getWaterfallGroupMode().	825
getX1MajorGridStepAutoValue().	826
getX1ScaleMaxAutoValue().	827
getX1ScaleMinAutoValue().	827
getY1MajorGridColorBand1().	828
getY1MajorGridColorBand2().	829
getY1MajorGridStepAutoValue().	830
getY1ScaleMaxAutoValue().	831
getY1ScaleMinAutoValue().	832
getY2MajorGridColorBand1().	833
getY2MajorGridColorBand2().	834
getY2MajorGridStepAutoValue().	835

getY2ScaleMaxAutoValue().....	836
getY2ScaleMinAutoValue().....	837
restore Methods	838
restoreAllSlices().....	838
restoreDefaults().....	839
save()	840
send Methods	840
sendChartToFTP().....	840
sendGIFToFile().....	841
sendGIFToFileExc().....	841
sendGIFToFTP().....	842
sendGIFToStream().....	843
sendImageMapToStream().....	843
sendJPEGToStream().....	844
sendPNGToFile().....	845
sendPNGToFileExc().....	845
sendPNGToStream().....	845
sendPNGToStreamExc().....	846
set Methods	846
setAltFmtFrameNumColors().....	846
setAltFmtFrameSeparator().....	848
setAutoColorLerpFactor().....	848
setAutoSkip().....	849
setAxisAssignment().....	850
setAxisDescending().....	853
setAxisOffset().....	854
setAxisSide().....	856
setAxisSize().....	857
setBorder().....	859
setBorderColor().....	860
setBorderType().....	861
setBorderWidth().....	862
setCornerArc().....	863
setCornerArcWidth/Height().....	864
setCubeRotationMatrix().....	865

setCurveFitPolynomialOrder().	866
setCurveFitType().	868
setCustomDataText().	869
setCustomDataTextFormat().	870
setCustomDataTextFormatPattern().	872
setData().	875
setDataLabel().	876
setDataRange().	877
setDataRangeToExtent().	878
setDataTextAngle().	879
setDataTextRadius().	881
setDataValue().	882
setDisplay().	883
setDisplayOffScale().	884
setDrawTicksPerpendicular().	885
setExceptionalRiser().	887
setExcludeMaxLabel().	888
setExcludeMinLabel().	889
setFillColor().	890
setFillType().	892
setFontName().	893
setFontSize().	894
setFontSizeAbsolute().	896
setFontSizeInPoints().	897
setFontSizeVC().	898
setFontStyle().	899
setGradientAutoDirectionMode().	900
setGradientDirection().	902
setGradientNumPins().	903
setGradientPinLeftColor().	904
setGradientPinPosition().	906
setGradientPinRightColor().	908
setGridCount().	909
setGridStep().	910
setGridStepAuto().	911

setGridStyle().	913
setGroupLabel().	914
setIgnoreSeries().	915
setInset().	916
set...Insets.	917
setLabelStagger().	918
setLegendPosition().	919
setLegendRect().	921
setLevelBorderColor().	921
setLevelDateFormat().	923
setLevelFillColor().	924
setLevelHeight().	925
setLevelInterval().	925
setLevelTextColor().	926
setLevelTextFormat().	927
setLineBasicStrokeType().	928
setLineCap().	929
setLineDash().	930
setLineDashPhase().	931
setLineJoin().	931
setLineMiterLimit().	932
setLineStyleLegendMarkers().	933
setLineWidth().	937
setLineWidthAllSeries().	938
setLogScale().	938
setLogScaleBase().	940
setMarkerShape().	940
setMarkerSize().	942
setMaxNumAnnotations().	943
setNoExceptionalRiser().	944
setNonZeroBaseline().	944
setNonZeroBaselineValue().	945
setNoSelection().	946
setPieBreakoutMode().	946
setPieBreakoutPercentage().	947

setPieSliceDelete()	948
setPieSliceDetach()	949
setPlaceAlign()	950
setPlaceMargin()	951
setPlaceMaxNumWordWrapLines()	951
setPlaceMinFontSize()	952
setPlaceRect()	954
setPlaceResize()	954
setPlaceSkip()	957
setPlaceTruncateCount()	957
setPlaceWordWrap()	959
setQuadrantLineValueX()	961
setQuadrantLineValueY()	962
setRect()	963
setReferenceLine()	964
setReferenceLineValue()	965
setReferenceLineXYRatio()	966
setScaleMax()	968
setScaleMaxAuto()	969
setScaleMin()	970
setScaleMinAuto()	971
setScaleMustIncludeZero()	972
setSeriesAreRows()	974
setSeriesBorderColor()	976
setSeriesFillColor()	977
setSeriesLabel()	979
setSeriesLooping()	980
setSeriesType()	981
setShadowColor()	984
setShadowDisplay()	985
setShadowFillType()	986
setShadowGradientEndColor()	986
setShadowGradientStartColor()	987
setShadowTextureURL()	988
setShadowXOffset()	989

setShadowYOffset().	990
setSize().	990
setSkipBegin().	991
setSkipCount().	992
setTextBackgroundColor().	993
setTextFormatPattern().	994
setTextFormatPreset().	995
setTextJustHoriz().	996
setTextJustVert().	998
setTextRotation().	1000
setTextStrikethrough().	1001
setTextString().	1002
setTextureBGColor().	1003
setTextureDisplayMode().	1004
setTextureFlipMode().	1005
setTextureResource().	1006
setTextureURL().	1007
setTextWrap().	1008
setTickCount().	1009
setTickLength().	1011
setTickStep().	1011
setTickStepAuto().	1012
setTickStyle().	1013
setTickWidth().	1013
setToolTipBackdropColor().	1014
setToolTipBackdropSize().	1015
setToolTipBackgroundColor().	1016
setToolTipColor().	1016
setToolTipFontName().	1017
setToolTipFontSize().	1019
setToolTipFontStyle().	1020
setToolTipOff().	1021
setToolTipOn().	1022
setTransparentBorderColor().	1022
setTransparentFillColor().	1022

setUseReferenceLineXYRatio().....	1023
setUserToolTip().....	1023
setUserToolTipDefault().....	1025
setUseTextBackgroundColor().....	1025
setWaterfallGroupMode().....	1027
setX1().....	1029
setX2().....	1030
setY1().....	1030
setY2().....	1031
setZoomIndex().....	1031
6. Methods That Return an Object ID	1033
Object ID Methods	1033
getAllSeries()	1037
getAnnotation()	1038
getAnnotationBox()	1038
getAnnotationLabel()	1039
getAnnotationLine()	1040
getAreaRiser()	1040
getAreaRiserBottom()	1041
getAreaRiserTop()	1041
getBarRiser()	1042
getBarRiserBottom()	1042
getBarRiserSide()	1043
getChartBackground()	1043
getColorByHeight()	1043
getCubeFloor()	1044
getCubeFloorGridX()	1045
getCubeFloorGridZ()	1046
getCubeLeftWall()	1047
getCubeLeftWallGridY()	1047
getCubeLeftWallGridZ()	1048
getCubeRightWall()	1049
getCubeRightWallGridX()	1049
getCubeRightWallGridZ()	1050

getCubeRiserFace()	1051
getCubeRiserGridX()	1051
getCubeRiserGridY()	1052
getCubeRiserGridZ()	1053
getCurveFitCorrelationText()	1053
getCurveFitCorrelationTextbox()	1053
getCurveFitEquationText()	1054
getCurveFitEquationTextbox()	1054
getCurveFitLineRiser()	1055
getDataText()	1056
getExceptionalRiser()	1056
getFootnote()	1057
getFootnoteBox()	1058
getFrame()	1059
getFrameBottom()	1059
getFrameSeparator()	1059
getFrameSide()	1060
getFunnelCrust()	1060
getFunnelFeeler()	1061
getFunnelLabel()	1062
getFunnelValueLabel()	1063
getGaugeArea()	1064
getGaugeBackground()	1064
getGaugeBand1...5()	1065
getGaugeBand1...5Label()	1066
getGaugeLabel()	1066
getGaugeMajorTick()	1067
getGaugeMinorTick()	1068
getGaugeNeedle()	1068
getGaugeScaleArea()	1069
getGaugeTitle()	1069
getGlobal()	1070
getGroup()	1070
getGroupScroller()	1071
getGroupScrollerEditBox()	1072

getGroupScrollerEditLeft()	1073
getGroupScrollerEditRight()	1073
getGroupScrollerEditThumb()	1074
getGroupScrollerThumb()	1074
getHistogramRiser()	1075
getLegendArea()	1075
getLegendMarker()	1076
getLegendScroller()	1076
getLegendScrollerThumb()	1077
getLegendText()	1078
getLegendTextBox()	1079
getLineMarker()	1079
getLineRiser()	1080
getLineRiserSide()	1081
getLineRiserTop()	1081
getNestedO1Label0...8()	1082
getNestedO1Label0...8Box()	1082
getNestedO1LabelLine()	1083
getNonZeroBaseLineZeroLine()	1084
getO1Axis()	1084
getO1AxisLine()	1085
getO1Label()	1086
getO1MajorGrid()	1087
getO1MajorTick()	1088
getO1MinorGrid()	1089
getO1MinorTick()	1090
getO1Title()	1091
getO1TitleBox()	1092
getO2Axis()	1093
getO2Label()	1095
getO2MajorGrid()	1095
getO2MinorGrid()	1096
getO2Title()	1097
getO2TitleBox()	1098
getParetoLineMarker()	1099

getParetoLineRiser()	1100
getParetoLineRiserSide()	1101
getParetoLineRiserSide1()	1101
getParetoLineRiserSide2()	1102
getParetoLineRiserTop()	1102
getParetoLineRiserTop1()	1102
getParetoLineRiserTop2()	1102
getPieBarConnectLineBottom()	1103
getPieBarConnectLines()	1103
getPieBarLabel()	1104
getPieBarRiserTop()	1104
getPieBarSlice()	1104
getPieBarSliceCrust()	1104
getPieFeelerLine()	1105
getPieFrame()	1105
getPieLabel()	1106
getPieRingLabel()	1107
getPieSliceFeelerPercent()	1108
getPieSliceFeelerValue()	1109
getPieSliceLabel()	1110
getPlaceHorzO1LabelsTitle()	1111
getPlaceVertO1LabelsTitle()	1113
getQuadrantLine()	1115
getReferenceLine()	1115
getReferenceLineLegendText()	1116
getReferenceLineO1()	1117
getReferenceLineO2()	1118
getReferenceLineX1()	1118
getReferenceLineY1()	1118
getReferenceLineY2()	1119
getReferenceLineTextO1()	1120
getReferenceLineTextO2()	1120
getReferenceLineTextX1()	1120
getReferenceLineTextY1()	1121
getReferenceLineTextY2()	1121

getScatterQuadrantLine()	1121
getSeries()	1122
getSeriesGroup()	1123
getSeriesScroller()	1124
getSeriesScrollerEditBox()	1126
getSeriesScrollerEditLeft()	1126
getSeriesScrollerEditRight()	1126
getSeriesScrollerEditThumb()	1127
getSeriesScrollerThumb()	1127
getSlice()	1127
getSliceCrust()	1127
getSliceRing()	1128
getSpectralLegendMarker()	1128
getSpectralMarker()	1129
getStackConnectLine()	1129
getStock52WeekHighLine()	1130
getStock52WeekLowLine()	1131
getStockCloseTick()	1132
getStockFallingRiser()	1133
getStockHighLine()	1134
getStockLowerRiser()	1134
getStockLowLine()	1134
getStockMALine()	1134
getStockOpenTick()	1135
getStockRiser()	1136
getStockRisingRiser()	1136
getStockVolumeRiser()	1137
getSubTitle()	1138
getSubTitleBox()	1139
getThermGaugeArea()	1140
getThermGaugeBackground()	1141
getThermGaugeNeedle()	1142
getTimeScaleLevel()	1143
getTitle()	1144
getTitleBox()	1145

getWaterFallLine()	1146
getX1Axis()	1147
getX1AxisLine()	1148
getX1Label()	1149
getX1MajorGrid()	1150
getX1MajorTick()	1151
getX1MinorGrid()	1152
getX1MinorTick()	1153
getX1Title()	1153
getX1TitleBox()	1154
getX1ZeroLine()	1155
getY1Axis()	1156
getY1AxisLine()	1157
getY1FrontZeroLine()	1158
getY1Label()	1159
getY1MajorGrid()	1160
getY1MajorTick()	1161
getY1MinorGrid()	1162
getY1MinorTick()	1163
getY1Title()	1164
getY1TitleBox()	1165
getY1ZeroLine()	1166
getY2Axis()	1167
getY2AxisLine()	1168
getY2FrontZeroLine()	1169
getY2Label()	1170
getY2MajorGrid()	1171
getY2MajorTick()	1172
getY2MinorGrid()	1173
getY2MinorTick()	1174
getY2Title()	1175
getY2TitleBox()	1176
getY2ZeroLine()	1177
getY3Axis()	1178
getY3AxisLine()	1179

getY3FrontZeroLine()	1180
getY3Label()	1181
getY3MajorGrid()	1183
getY3MajorTick()	1184
getY3MinorGrid()	1184
getY3MinorTick()	1186
getY3Title()	1186
getY3TitleBox()	1188
getY3ZeroLine()	1189
getY4Axis()	1190
getY4AxisLine()	1191
getY4Label()	1192
getY4MajorGrid()	1193
getY4MajorTick()	1194
getY4MinorGrid()	1195
getY4MinorTick()	1196
getY4Title()	1197
getY4TitleBox()	1198
getY4ZeroLine()	1200
getY5Axis()	1201
getY5AxisLine()	1202
getY5Label()	1203
getY5MajorGrid()	1205
getY5MajorTick()	1206
getY5MinorGrid()	1206
getY5MinorTick()	1208
getY5Title()	1208
getY5TitleBox()	1210
getY5ZeroLine()	1211
A. Graph Types and 3D Preset Viewing Angles	1213
Graph Types	1213
3D Graphs	1217
3D Bars (GraphType=0)	1218
3D Pyramid (GraphType=1)	1218

3D Octagon (GraphType=2).....	1219
3D Cylinder (GraphType=3).....	1219
3D Floating Cubes (GraphType=4).....	1220
3D Floating Pyramids (GraphType=5).....	1220
3D Connected Series Area (GraphType=6).....	1221
3D Connected Series Ribbon (GraphType=7).....	1221
3D Cone (Graph Type=8).....	1222
3D Connected Group Area (GraphType=9).....	1222
3D Connected Group Ribbon (GraphType=10).....	1223
3D Sphere (GraphType=11).....	1223
3D Surface (GraphType=12).....	1224
3D Surface with Sides (GraphType=13).....	1224
3D Honeycomb Surface (GraphType=14).....	1225
3D Smooth Surface (GraphType=15).....	1225
3D Smooth Surface with Sides (GraphType=16).....	1226
Vertical Bar Graphs.....	1226
Vertical Clustered Bars (GraphType=17).....	1226
Vertical Stacked Bars (GraphType=18).....	1227
Vertical Dual-Axis Clustered Bars (GraphType=19).....	1227
Vertical Dual-Axis Stacked Bars (GraphType=20).....	1228
Vertical Bi-Polar Clustered Bars (GraphType=21).....	1228
Vertical Bi-Polar Stacked Bars (GraphType=22).....	1229
Vertical Percent Bars (GraphType=23).....	1229
Horizontal Bar Graphs.....	1229
Horizontal Clustered Bars (GraphType=24).....	1230
Horizontal Stacked Bars (GraphType=25).....	1230
Horizontal Dual-Axis Clustered Bars (GraphType=26).....	1231
Horizontal Dual-Axis Stacked Bars (GraphType=27).....	1231
Horizontal Bi-Polar Clustered Bars (GraphType=28).....	1232
Horizontal Bi-Polar Stacked Bars (GraphType=29).....	1232
Horizontal Percent Bars (GraphType=30).....	1233
Vertical Area Graphs.....	1233
Vertical Absolute Area (GraphType=31).....	1233
Vertical Stacked Area (GraphType=32).....	1234
Vertical Bi-Polar Absolute Area (GraphType=33).....	1234

Vertical Bi-Polar Stacked Area (GraphType=34).....	1235
Vertical Percent Area (GraphType=35).....	1235
Horizontal Area Graphs.....	1235
Horizontal Absolute Area (GraphType=36).....	1236
Horizontal Stacked Area (GraphType=37).....	1236
Horizontal Bi-Polar Absolute Area (GraphType=38).....	1237
Horizontal Bi-Polar Stacked Area (GraphType=39).....	1237
Horizontal Percent Area (GraphType=40).....	1238
Vertical Line Graphs.....	1238
Vertical Absolute Line (GraphType=41).....	1238
Vertical Stacked Line (GraphType=42).....	1239
Vertical Dual-Axis Absolute Line (GraphType=43).....	1239
Vertical Dual-Axis Stacked Line (GraphType=44).....	1240
Vertical Bi-Polar Absolute Line (GraphType=45).....	1240
Vertical Bi-Polar Stacked Line (GraphType=46).....	1241
Vertical Percent Line (GraphType=47).....	1241
Horizontal Line Graphs.....	1242
Horizontal Absolute Line (GraphType=48).....	1242
Horizontal Stacked Line (GraphType=49).....	1242
Horizontal Dual-Axis Absolute Line (GraphType=50).....	1243
Horizontal Dual-Axis Stacked Line (GraphType=51).....	1243
Horizontal Bi-Polar Absolute Line (GraphType=52).....	1244
Horizontal Bi-Polar Stacked Line (GraphType=53).....	1244
Horizontal Percent Line (GraphType=54).....	1245
Pie Graphs.....	1245
Pie (GraphType=55).....	1245
Ring Pie (GraphType=56).....	1246
Multi Pie (GraphType=57).....	1246
Multi Ring Pie (GraphType=58).....	1247
Multi Proportional Pie (GraphType=59).....	1247
Multi Proportional Ring Pie (GraphType=60).....	1248
Scatter Graphs.....	1248
XY Scatter (GraphType=61).....	1248
XY Scatter Dual-Axis (GraphType=62).....	1249
XY Scatter with Labels (GraphType=63).....	1249

XY Scatter with Labels Dual-Axis (GraphType=64).....	1250
Polar Graphs.....	1250
XY Polar (GraphType=65).....	1250
XY Polar Dual-Axis (GraphType=66).....	1251
Radar Graphs.....	1251
Radar Line (GraphType=67).....	1251
Radar Area (GraphType=68).....	1252
Radar Line Dual-Axis (GraphType=69).....	1252
Stock Graphs.....	1252
Open-Hi-Lo-Close Candle Stock Graph (GraphType=70).....	1252
Open-Hi-Lo-Close Candle Stock Graph with Volume (GraphType=71).....	1253
Candle Stock Open-Close (GraphType=72).....	1254
Stock Hi-Lo (GraphType=73).....	1254
Stock Hi-Lo Dual-Axis (GraphType=74).....	1255
Stock Hi-Lo Bi-Polar (GraphType=75).....	1255
Stock Hi-Lo Close (GraphType=76).....	1256
Stock Hi-Lo Close Dual-Axis (GraphType=77).....	1256
Stock Hi-Lo Close Bi-Polar (GraphType=78).....	1257
Stock Hi-Lo Open-Close (GraphType=79).....	1257
Stock Hi-Lo Open-Close Dual-Axis (GraphType=80).....	1258
Stock Hi-Lo Open-Close Bi-Polar (GraphType=81).....	1258
Stock Hi-Lo with Volume (GraphType=82).....	1259
Stock Hi-Lo Open-Close with Volume (GraphType=83).....	1259
Candle Stock Open-Close with Volume (GraphType=84).....	1260
Stock Hi-Lo Close with Volume (GraphType=88).....	1260
Histogram Graphs.....	1261
Vertical Histogram (GraphType=85).....	1261
Horizontal Histogram (GraphType=86).....	1261
Spectral Map.....	1262
Spectral Map (GraphType=87).....	1262
Bubble Graphs.....	1262
Bubble Graph (GraphType=89).....	1262
Bubble Graph with Labels (GraphType=90).....	1263
Bubble Graph Dual-Axis (GraphType=91).....	1263
Bubble Graph with Labels Dual-Axis (GraphType=92).....	1264

Pie-Bar Graphs.....	1264
Pie-Bar Graph (GraphType=93).....	1264
Ring Pie-Bar Graph (GraphType=94).....	1265
Special Graphs.....	1265
Vertical Waterfall Graph (GraphType=100).....	1265
Horizontal Waterfall Graph (GraphType=101).....	1266
Pareto Graph (GraphType=102).....	1266
Multi-Y Y1/Y2/Y3-Axes Graph (GraphType=103).....	1267
Multi-Y Y1/Y2/Y3/Y4-Axes Graph (GraphType=104).....	1267
Multi-Y Y1/Y2/Y3/Y4/Y5-Axes Graph (GraphType=105).....	1268
Funnel Graph (GraphType=106).....	1268
Pyramid Graph (GraphType=111).....	1269
Thermometer Gauge Graph (GraphType=138).....	1269
Box Plot Graphs.....	1269
Box Plot Graph (GraphType=124).....	1270
Horizontal Box Plot Graph (GraphType=130).....	1270
3D Preset Viewing Angles	1270
Standard.....	1271
Tall and Skinny.....	1272
From the Top.....	1272
Distorted.....	1273
Short and Fat.....	1273
Group's Eye.....	1274
Group Emphasis.....	1274
Few Series.....	1275
Few Groups.....	1275
Distorted Standard.....	1276
Shorter and Fatter.....	1276
Thick Wall for Series.....	1277
Thick Wall Standard.....	1277
California Special.....	1278
Blast-O-Vision.....	1278
Special.....	1279

B. Graph Properties Index	1281
Current Graph Properties	1281
Graph Properties Reserved For Future Use	1306
Graph Properties Reserved for Internal Use	1307
C. Graph Methods Index	1309
Current Graph Methods	1309
Reserved for Future Use	1329
Reserved for Internal Use	1331
D. IdentObj Methods Index	1339
Current IdentObj Methods	1339
IdentObj Methods Reserved for Future Use	1350
E. Static Variables	1353
Current Static Variables	1353
Customer Connections	1371

Preface

This documentation describes how to use WebFOCUS Graphics API. It is intended for developers of WebFOCUS applications.

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Graphing Basics	Provides an introduction to Scalable Vector and Portable Network Graphics.
2	Using WebFOCUS Graphics	Describes concepts of methods and properties.
3	Using Data in a Graph	Describes the use of data in a graph, including managing data overflow and accounting for missing data.
4	Properties	Describes the properties you can set for each graph type.
5	Methods	Describes methods that you can use to set graph properties.
6	Methods That Return an Object ID	Describes methods that return an object ID.
A	Graph Types and 3D Preset Viewing Angles	Provides a reference for graph types and preset viewing angles.
B	Graph Properties Index	Provides an index of properties for graph types.
C	Graph Methods Index	Provides an index of methods for graph types.
D	IdentObj Methods Index	Describes IdentObj methods used for graph types.
E	Static Variables	Lists static variables for graph types.

Documentation Conventions

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

Convention	Description
THIS TYPEFACE or this typeface	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	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.
[]	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 any 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

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 <http://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.

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.

Graphing Basics

You can include API calls in WebFOCUS procedures to create dynamic, sophisticated graphs that showcase your business data.

If you save a graph generated by a WebFOCUS procedure in Scalable Vector Graphics (SVG) or Portable Network Graphics (PNG) format, you can produce a high quality image for display or print, with advanced features provided by the API calls.

WebFOCUS graphs are plotted on a Cartesian grid using the Virtual Coordinate System, introduced in this section. These topics describe all of the features for Scalable Vector and Portable Network Graphics.

In this chapter:

- ❑ [What Is Scalable Vector Graphics?](#)
- ❑ [Viewing Scalable Vector Graphics](#)
- ❑ [Portable Network Graphics](#)
- ❑ [Virtual Coordinate System](#)
- ❑ [Implementing API Calls](#)

What Is Scalable Vector Graphics?

Scalable Vector Graphics (SVG) is a file format, based on Extensible Markup Language (XML), that enables web application developers to design powerful, interactive graphical images.

SVG was introduced, and continues to be developed, by the World Wide Web Consortium (W3C), a non-profit group of corporations that work together to create new, open standard products for the web. Open standard products are not proprietary—they do not belong to any company or individual. Other open standard products created by the W3C are HyperText Markup Language (HTML) and XML.

The following table compares vector graphs with bitmapped graphs, which have been used traditionally in web applications until now.

Vector Graphs	Bitmapped Graphs
SVG	GIF, JPEG, PNG
Describe shapes and paths	Provide information about every pixel
Are compact and generate high quality images on any media	Are bulky and sometimes generate poor quality images
Are scalable and produce high quality images regardless of resolution	Show progressive, jagged edges, known as the staircase effect, when displayed at lower resolutions or stretched

SVG has several other advantages over bitmapped graphs:

- ❑ You can zoom in on any part of an SVG graph, always seeing a clear image.
- ❑ You can perform searches and make edits on any part of an SVG graph because it is actually composed of text.
- ❑ You can interactively respond to a variety of user actions with highlighting, pop-ups, special effects, audio, and animation.
- ❑ SVG files are usually much smaller and easier to compress than bitmapped files.
- ❑ SVG supports 16 million colors rather than 256 colors. It also supports industry-wide initiatives for achieving web-based color consistency across systems.

An SVG graph is created with underlying XML code, which also has benefits for development on the web. SVG:

- ❑ Works well across platforms, output resolutions, and methods of representing colors.
- ❑ Is supported in virtually any language for global implementation in many locales.
- ❑ Is compatible with any scripting language (for example, JavaScript) or procedural language (for example, WebFOCUS) and uses data from any type of data source, from ASCII to relational.

Viewing Scalable Vector Graphics

To view SVG graphs in a browser, all you need is a free plug-in that you can download from Adobe®, called the SVG Viewer. The SVG Viewer is available for Netscape® and Microsoft® Internet Explorer.

For more information about downloading and using this tool, go to the Adobe website and search for information on the SVG Viewer.

You can also embed SVG graphs in PDF documents. Viewing these documents requires only the Adobe PDF Viewer—no browser plug-in is necessary.

Portable Network Graphics

In addition to SVG, WebFOCUS gives you the option of using the Portable Network Graphics (PNG) file format. Since a plug-in viewer is required for browser display of the SVG format, you may decide to use PNG format.

PNG graphs are bitmapped and can support 16 million colors. In addition, they have lossless compression (data is decompressed 100% back to the original). Therefore, saving, altering, and resaving a PNG does not degrade its overall quality.

For these reasons, PNG graphs are superior to GIF and JPEG graphs, and you can display them in virtually any modern browser. The major difference between PNG and SVG is that SVG is vector based and uses XML as the file format.

Virtual Coordinate System

Whether you save your graphs in SVG or PNG format, you will plot them on a Cartesian grid using the Virtual Coordinate System. The Virtual Coordinate System provides a way to position and size graph elements relative to the area in which the graph is displayed. Changing the aspect ratio (ratio of width to height) of the area results in the necessary adjustments to the graph elements, creating a graph that fits properly in the space allotted for it.

This feature has many benefits, including the ability to set font size with virtual coordinates, ensuring that text appears proportionate regardless of the size of a graph.

The Virtual Coordinate System provides data visualization in WebFOCUS applications widely distributed on the web to users with different display technologies and output devices.

A Look at Coordinate Systems: Cartesian and Virtual

This topic takes a look at the basics of coordinate systems to help you understand how you will create WebFOCUS graphs.

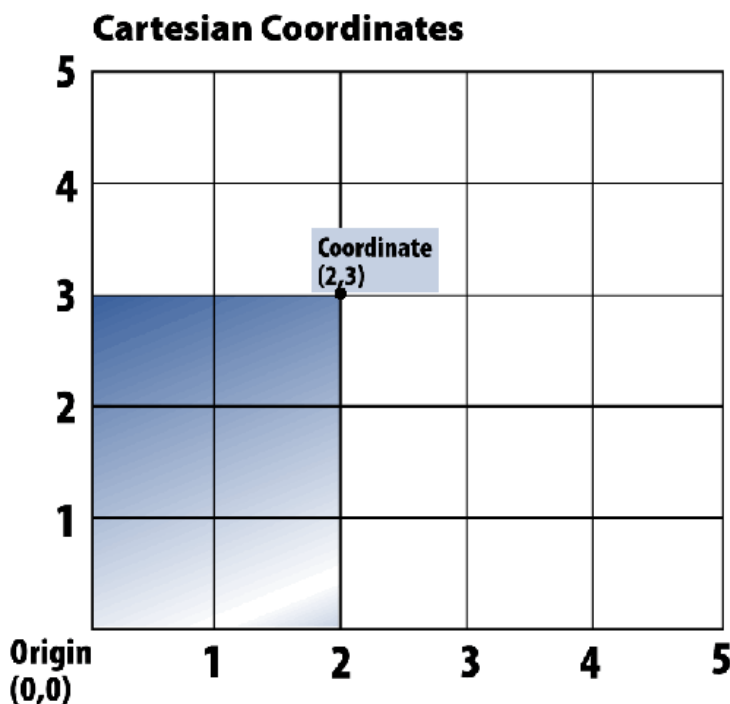
The Cartesian Coordinate System consists of two number lines perpendicular to each other. Called axes, the horizontal number line is the X-axis and the vertical number line is the Y-axis.

The axes form a plane (typically, a flat surface). With the Cartesian Coordinate System, you assign a pair of numbers, designated as (X,Y) , to identify a point on the plane. The first number is the X-coordinate and the second number is the Y-coordinate; together, the numbers are the coordinates of the point.

The coordinate point at which the two axes intersect is the origin $(0,0)$.

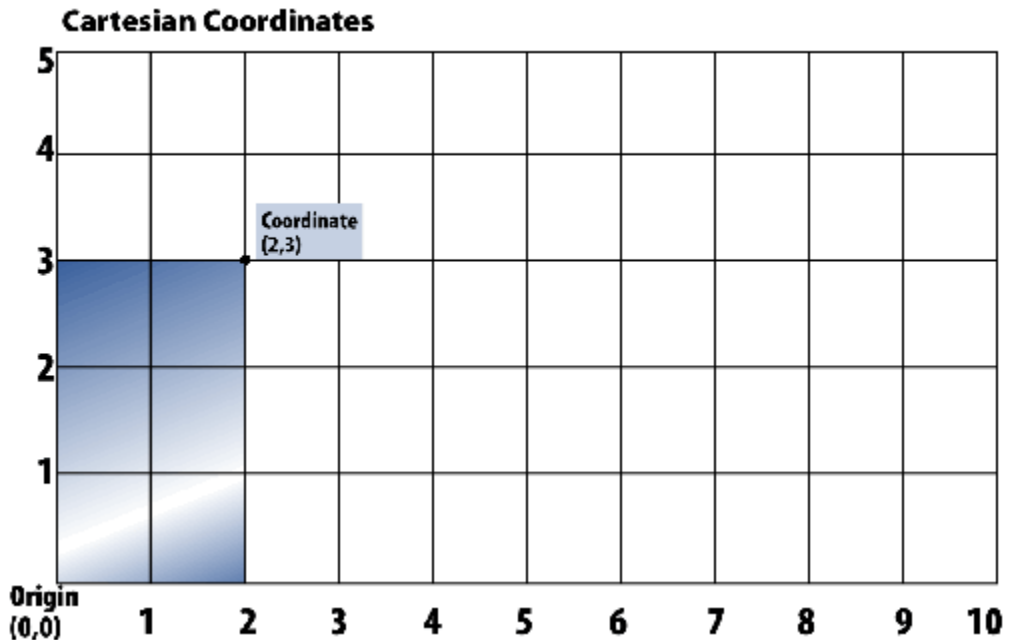
The following diagram plots a point on a plane. The Cartesian coordinates of the point are $(2,3)$. Starting at the origin $(0,0)$, the first number represents 2 units to the right of the origin on the X-axis, and the second number represents 3 units up on the Y-axis.

Tip: The two Cartesian axes divide a plane into four quarters, called quadrants. For simplicity, the diagrams show only one quadrant.



If we double the width of the grid—that is, change its aspect ratio (the ratio of width to height)—the number of coordinate points on the X-axis increases to 10. The space between coordinate points remains the same; the space between coordinate point 0 and coordinate point 1 (or between 1 and 2, 2 and 3, and so on) does not change between grids.

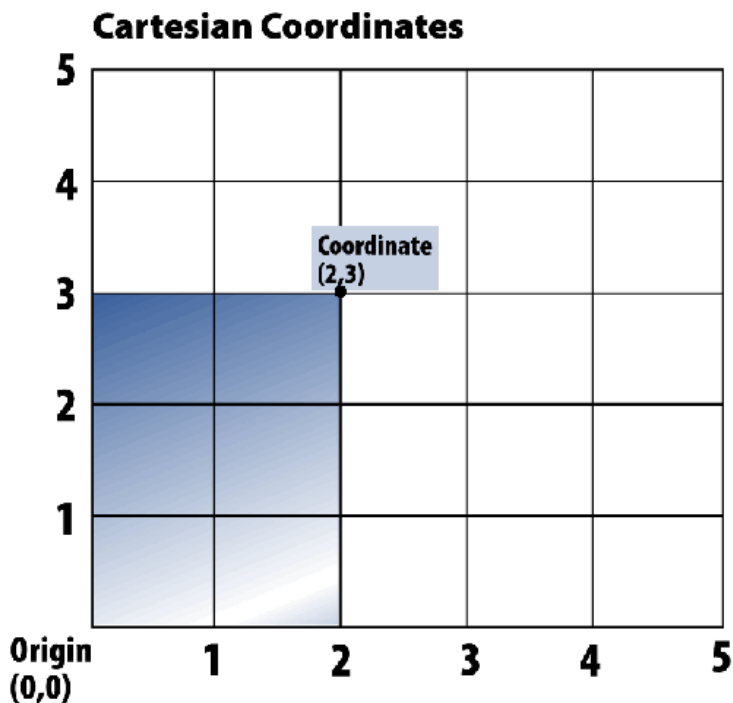
As a result, when we plot (2,3) on the wider grid, the 2 units to the right of the origin are the same width as before, and the shaded area is unchanged. In fact, it will always be rendered exactly the same regardless of the size and shape of the grid.

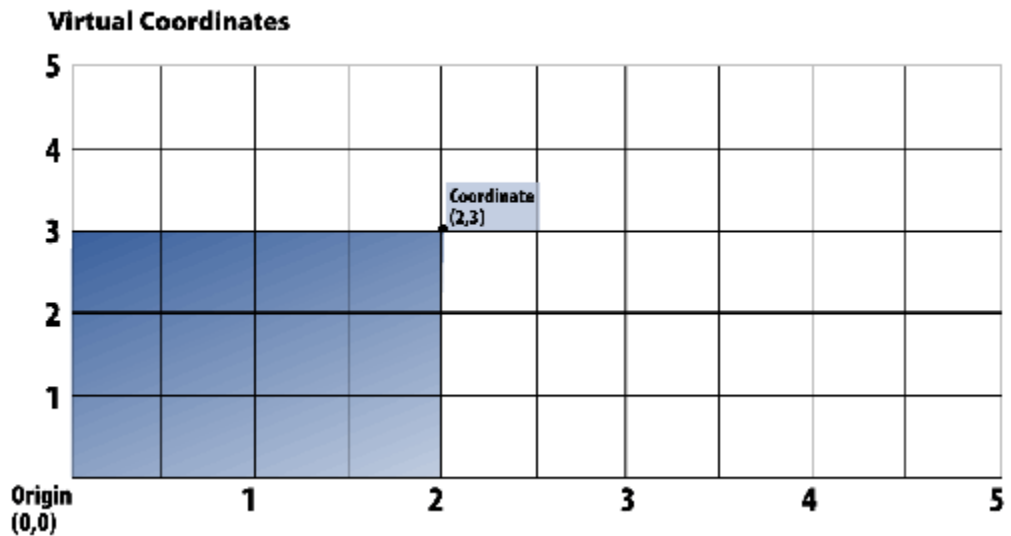


The **Virtual Coordinate System** used by WebFOCUS Graphics is based on a Cartesian grid. As shown before, the first diagram plots a point with Cartesian coordinates (2,3) on a Cartesian grid. This time, however, when we double the width of the grid (second diagram), the number of coordinate points on the X-axis remains 5, and, consequently, the space between coordinate points increases.

In the Virtual Coordinate System, the number of coordinate points on both axes is constant regardless of the size of the grid.

As a result, when we plot (2,3) on the wider "virtual" grid, the 2 units to the right of the origin are doubled in width from before. The Virtual Coordinate System, illustrated in the second diagram, results in the proportionate display of the shaded area in the increased width of the grid; it is stretched horizontally. The width of the shaded area will always change relative to the width of the grid.





Example: Using Virtual Coordinates to Control Font Size

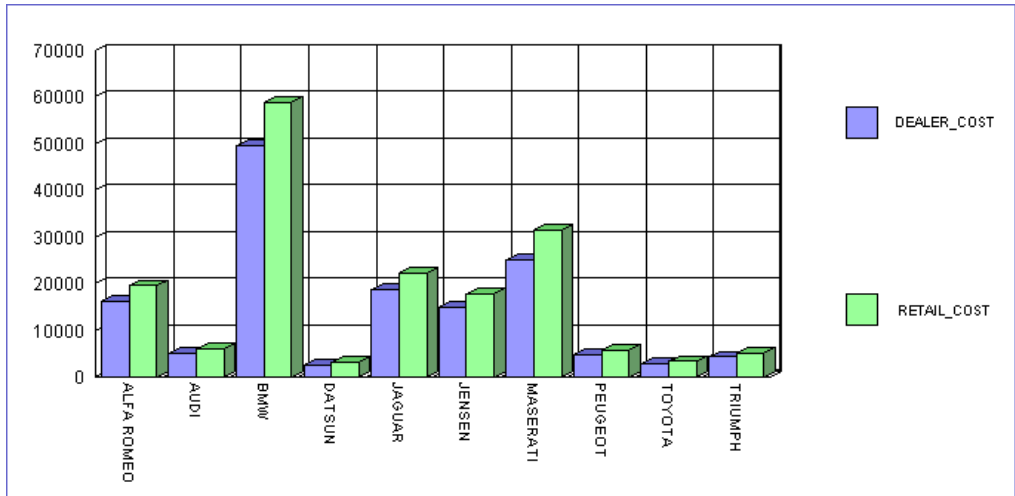
The following graphs are generated by a WebFOCUS procedure. The specific API calls that created these graphs are described in later sections of this manual.

This example illustrates the results you can achieve using virtual coordinates to control font size. The font size changes relative to the designated size of the area in which the graph is displayed (the graph output area).

In the graph request:

- ☐ Absolute coordinates control the font size for the X-axis and Y-axis labels.
- ☐ Virtual coordinates control the font size for the legend.
- ☐ The width and height of the graph output area are controlled by two WebFOCUS graph commands.

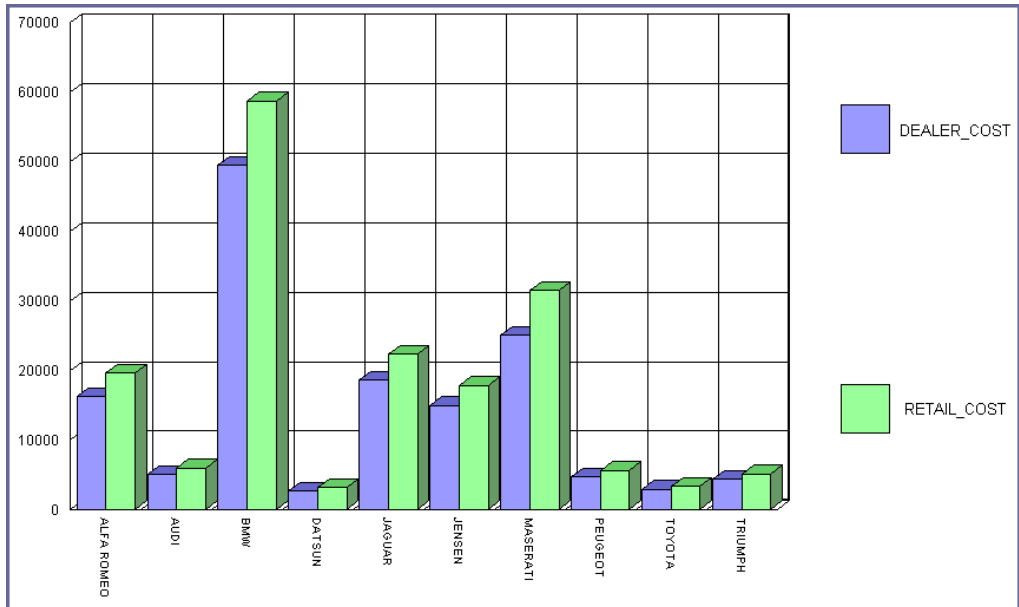
In the first graph, the font size for the axis labels and the legend is proportionate to the size of the graph. All the text on the graph is easy to read regardless of the way font size is specified in the procedure—with absolute coordinates or with virtual coordinates:



Next we modify the procedure to increase the size of the graph output area. The font size for the axis labels—specified with absolute coordinates—remains the same, and the text is now proportionately too small and harder to read than before. An absolute size, whether it is the size of a font or another graph element, always renders exactly the same whether the graph is large or small.

The font size for the legend—specified with virtual coordinates—increases proportionately with the size of the graph and the text is still easy to read.

In this example, the size of the vertical bars (risers) is also controlled with virtual coordinates. Like the size of the legend text, they increase proportionately when the graph output area is increased.



How Does WebFOCUS Graphics Use the Virtual Coordinate System?

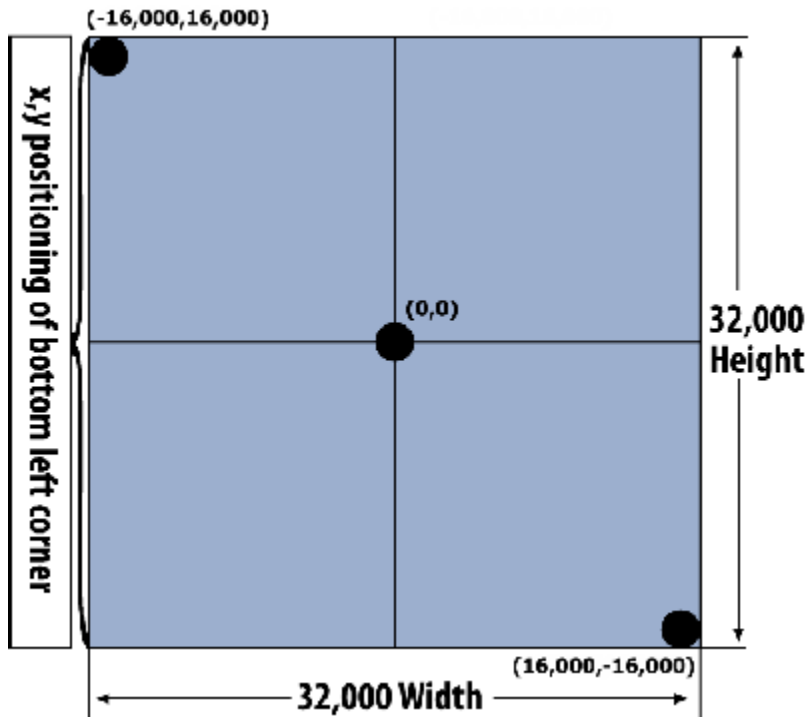
WebFOCUS Graphics uses the Virtual Coordinate System to position and size the elements of a graph. You can control several rectangular elements in WebFOCUS, but you typically create graphs using the following:

- ❑ **Applet Rectangle**—Defines the graph output area, which is the entire area allowed for the presentation of a graph. The Applet Rectangle sets the size and shape of the virtual coordinate grid used for plotting a WebFOCUS graph.
- ❑ **Frame Rectangle**—Is the area within which the graph coordinates are plotted.
- ❑ **Legend Rectangle**—Explains how to interpret the data plotted on a graph.

To plot a point in the Cartesian Coordinate System, you must provide at least two coordinates (X,Y). In WebFOCUS Graphics, four coordinates are required to plot a rectangle such as the Frame Rectangle or Legend Rectangle—X, Y, Width, and Height. The first two coordinates (X,Y) control the point location of the bottom left corner of the rectangle, while the last two coordinates control the width and height of the rectangle.

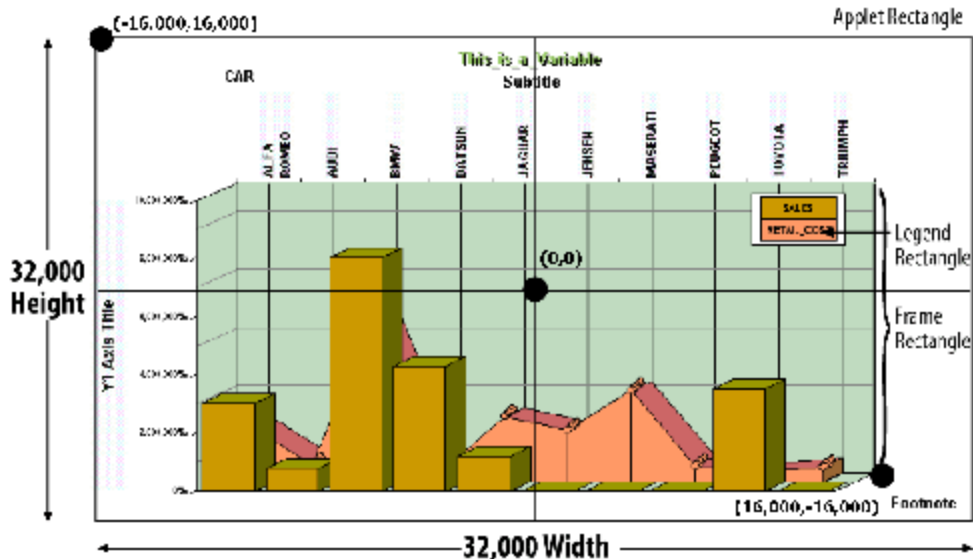
Two absolute values set the size and shape of the Applet Rectangle.

The following is a Cartesian grid showing the total number of X-axis and Y-axis coordinate points used to plot a WebFOCUS graph. It is represented as a square, with points plotted in the Cartesian Coordinate System. Coordinate point (0,0) is the center of the grid.



If we superimpose this grid over a WebFOCUS graph (next diagram), its shape changes from a square to a rectangle (the Applet Rectangle), a shape typically used in the presentation of a graph. However, the number of points on both the X-axis and Y-axis remains 32,000. The fact that the number of points is constant regardless of the size or shape of the grid is the reason why the coordinate system used by WebFOCUS Graphics is called the Virtual Coordinate System.

The values passed on a WebFOCUS Graphics API call to position and size graph elements do not apply to pixels; rather, they are conceptual points on a conceptual grid.



The examples that follow always show the position of coordinate point (0,0) to mark the center of the grid used in the Virtual Coordinate System. It will help you see the effect of the changes made to the sample graphs. The position of coordinate point (0,0) on the grid is constant, and graph elements such as rectangles move relative to it through the arguments passed on a WebFOCUS API call.

Example: Plotting a Frame Rectangle in the Default Applet Rectangle

In these examples, WebFOCUS Graphics uses the default values for the Applet Rectangle. You plot an Applet Rectangle with the WebFOCUS HAXIS setting (width of graph output area) and VAXIS setting (height of graph output area) in a procedure. In general, the optimal display format for an Applet Rectangle is 1.618 units in width for every 1 unit in height.

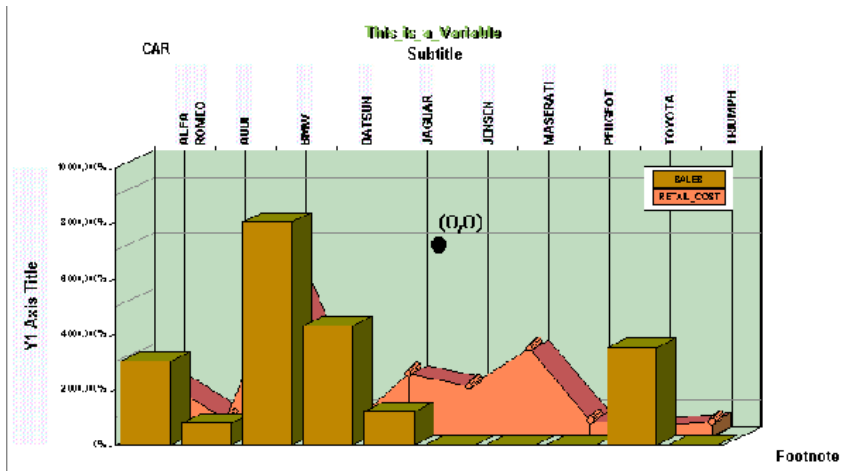
This aspect ratio is essentially a golden rectangle, a form used since ancient Greece in art and architecture. It is naturally pleasing to the human eye.

The WebFOCUS Graphics default values are:

```
SET HAXIS=770
SET VAXIS=400
```

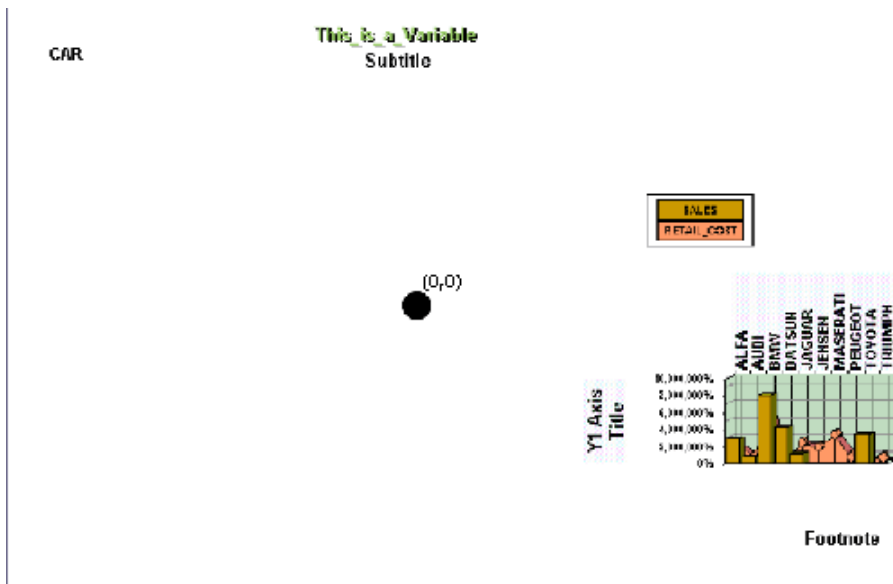
In the first example, the Frame Rectangle is plotted with the following WebFOCUS Graphics API code:

```
setRect(getFrame(),new Rectangle(-10753,-12619,22378,19359));
```



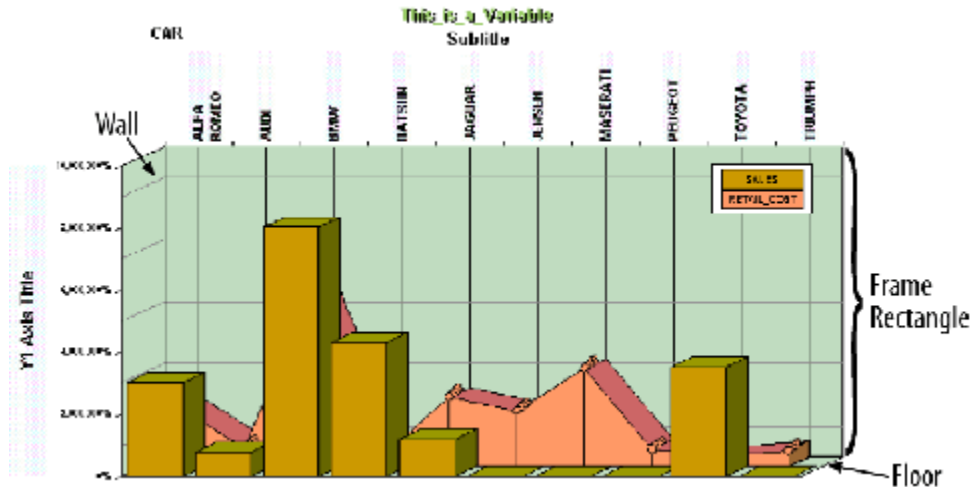
In the next example, the API code repositions the bottom left corner of the Frame Rectangle within the Applet Rectangle and decreases its size. The position of coordinate point (0,0) remains constant, and the Frame Rectangle moves relative to it. The API code is:

```
setRect(getFrame(),new Rectangle(10000,-10000,5000,5000));
```

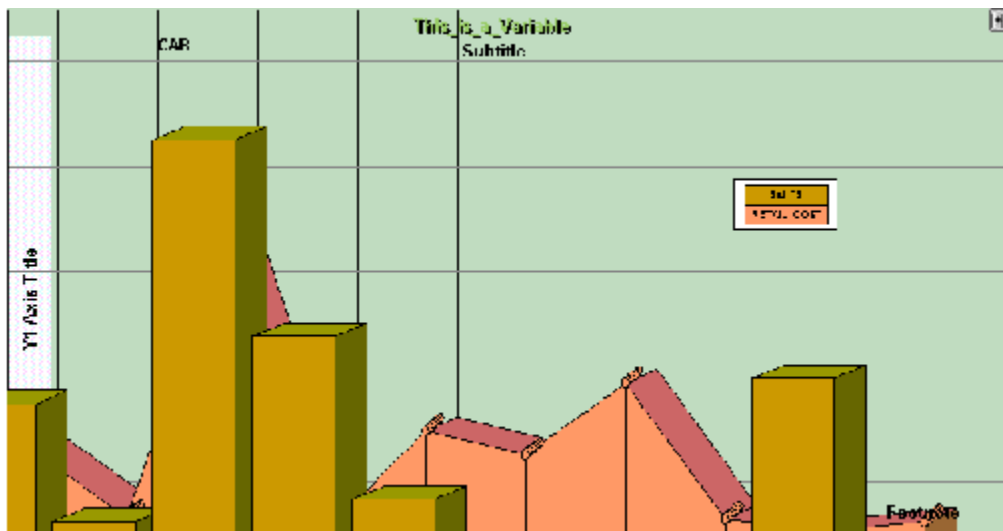


Example: Plotting a Graph With a Three-dimensional Effect

The Frame Rectangle does not include the wall or floor of a graph rendered with a three-dimensional effect for depth, as shown in the following diagram. This is important to keep in mind when you position and size the Frame Rectangle.



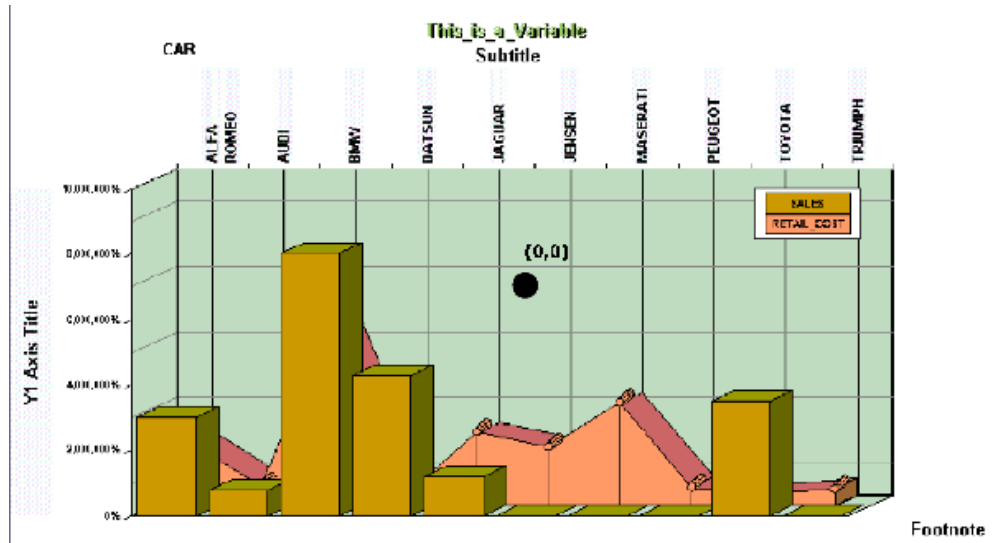
For example, you cannot set the coordinates to the maximum number allowed, in an attempt to make the Frame Rectangle fill the entire Applet Rectangle. To do that results in the loss of data, as illustrated in the following unintentional presentation:



Example: Plotting a Legend Rectangle

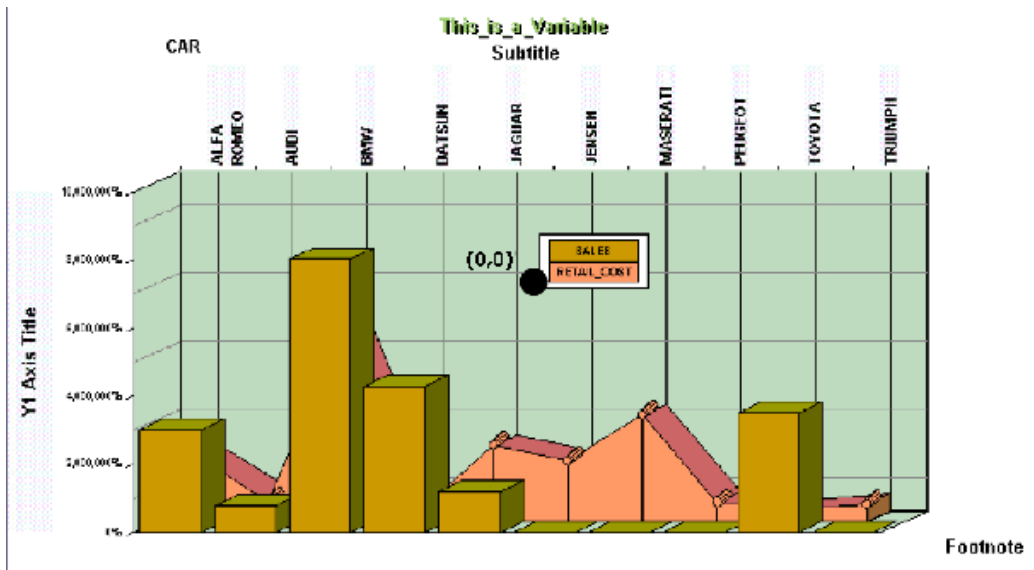
The virtual coordinates for a Legend Rectangle work just like those for a Frame Rectangle. In the first example, the Legend Rectangle is plotted with the following WebFOCUS Graphics API code:

```
setRect(getLegendArea(),new Rectangle(7300,2488,3300,3100)):
```



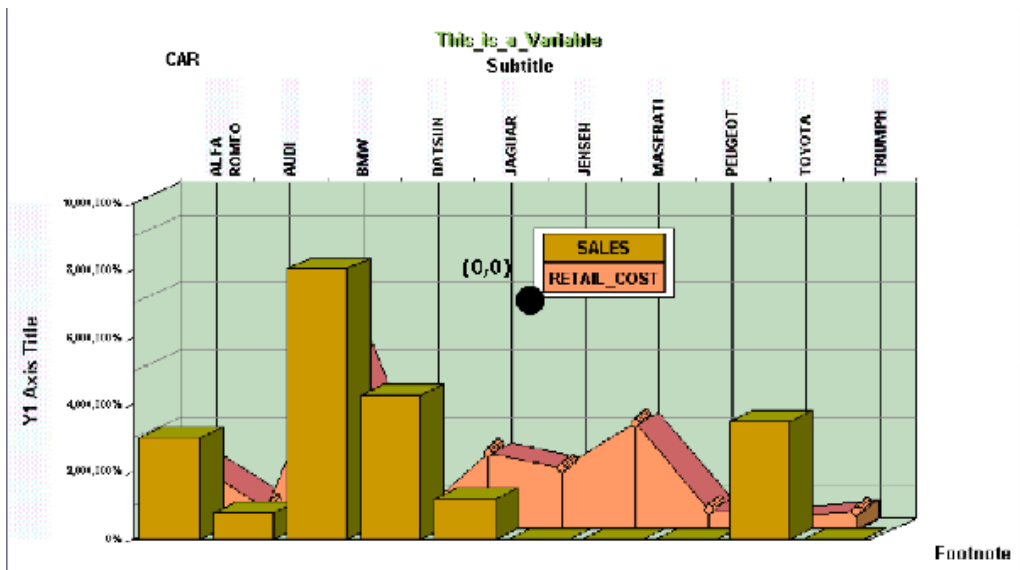
You can change the first two coordinates to reposition the legend toward the center of the graph:

```
setRect(getLegendArea(),new Rectangle(0,0,3300,3100)):
```



Or you can change the last two coordinates to enlarge the legend:

```
setRect(getLegendArea(),new Rectangle(0,0,4300,4100)):
```



Implementing API Calls

You can enhance graph output by adding a variety of API calls within *GRAPH_SCRIPT and *END commands in the StyleSheet section of the request (between ON GRAPH SET STYLE * and ENDSTYLE commands).

Syntax: **How to Implement an API Call**

```
GRAPH FILE filename
graph commands
.
.
.
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
API call
.
.
.
*END
WebFOCUS StyleSheet command
ENDSTYLE
END
```

where:

filename

Specifies a data source.

graph commands

Are commands allowed in a procedure. For a description of valid commands, with syntax and examples, see *Creating Reports With WebFOCUS Language* in the *Creating Reports With WebFOCUS Language* manual.

API call

Is a property setting, method call, or multiple property settings or method calls. For a description of property settings and method calls, with syntax and examples, see [Properties](#) on page 229, [Methods](#) on page 715, and [Methods That Return an Object ID](#) on page 1033.

WebFOCUS StyleSheet command

Is an optional WebFOCUS StyleSheet command or multiple StyleSheet commands. For a description of StyleSheet commands, with syntax and examples, see the *Formatting Reports: An Overview* chapter through the *Formatting Report Data* chapter in the *Creating Reports With WebFOCUS Language* manual.

Example: Customizing a Graph Using API Calls

The following annotated example illustrates how to customize a graph using four API calls. The API calls are highlighted in the request.

The calls illustrate the API syntax rules. They:

- ❑ Are case sensitive.
- ❑ End with a semicolon (;).
- ❑ Comply with explicitly defined spacing conventions.

The procedure is

```

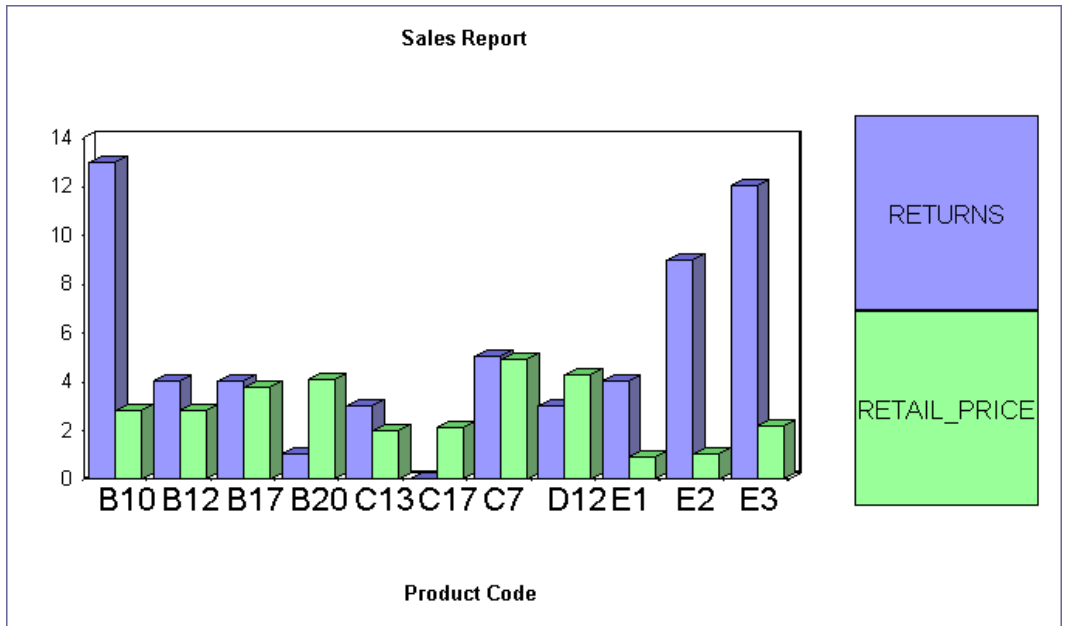
GRAPH FILE SALES
SUM RETURNS
RETAIL_PRICE
ACROSS PROD_CODE AS 'Product Code'
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
1. setLegendMarkerPosition(4);
2. set01LabelRotate(0);
3. setTitleString("Sales Report");
4. setTextJustHoriz(getTitle(),1);
*END
ENDSTYLE
END

```

where:

1. Displays legend text inside the legend marker.
2. Displays the X-axis labels horizontally.
3. Displays the title Sales Report.
4. Centers the title.

The output is:



Using WebFOCUS Graphics

This topic describes the concepts of methods and properties.

In this chapter:

- ☐ [How Properties and Methods Work](#)
- ☐ [The GraphType Property](#)
- ☐ [How Graphs Are Imaged](#)
- ☐ [Chart Layout Manager](#)
- ☐ [Setting the Color of the Page Surrounding the Graph](#)
- ☐ [Graph-Wide Parameters](#)
- ☐ [Graph Frame Attributes](#)
- ☐ [Axis Attributes](#)
- ☐ [Grid Lines](#)
- ☐ [Curve Fit Lines](#)
- ☐ [Java 2D Line Objects](#)
- ☐ [Risers and Markers](#)
- ☐ [Legends](#)
- ☐ [Labels and Titles](#)
- ☐ [Data Text](#)
- ☐ [Customizing Data Text](#)
- ☐ [Formatting Numbers and Numeric Labels](#)
- ☐ [Formatting Text Objects](#)
- ☐ [Java 2D Text Objects](#)
- ☐ [Autofitting](#)
- ☐ [Graph-Specific Properties and Methods](#)
- ☐ [Colors and Shading](#)
- ☐ [Textures and Gradients](#)
- ☐ [Panning, Rotating, and Scaling](#)
- ☐ [Graph Editing Characteristics](#)
- ☐ [Annotations](#)
- ☐ [Drill-Down and URLs](#)
- ☐ [Multiple Drill-Down From Graph](#)
- ☐ [Alternate Formatting](#)
- ☐ [Beveled Borders](#)
- ☐ [Reference Lines](#)

How Properties and Methods Work

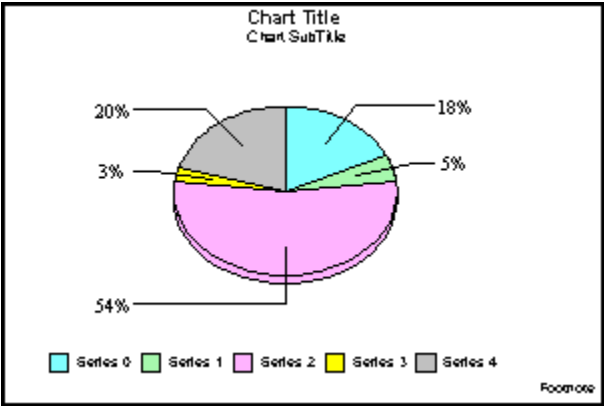
You can use properties to assign a single value to one or more objects in a graph. You can use all properties as methods by assigning a set or get prefix to the property name. Most methods are overloaded and allow you to select an object in a graph by its object ID, sequentially with an integer value.

The GraphType Property

Use the GraphType property to select and display a particular graph type.

Example:

```
/* Set the graph type to Pie */
perspective1.setGraphType(55);
```



GraphType Values

The following graph types can be selected with the GraphType property.

GraphType Values	Graph Category
0...16	3D (three-dimensional)
17...23	vertical bar
24...30	horizontal bar
31...35	vertical area
36...40	horizontal area

GraphType Values	Graph Category
55...60	pie
61...64	scatter
65...66	polar
67...69	radar
70...84 & 88	stock
85...82	histogram
87	spectral map
90...92	bubble graph
93...94	pie-bar graph
99	unreserved
100...101	waterfall graph
102	Pareto
103...105	multi-Y axes graph
106	funnel graph
110	gauge graph
111	pyramid graph
124	boxplot
130	hboxplot

See [Graph Types and 3D Preset Viewing Angles](#) on page 1213 for a complete list of graph types and an illustration of each graph.

GraphType Methods

You can use these methods to determine the graph type that is currently selected:

- ❑ `isChart3DType()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a value in the range: 0...7, 9, 10, or 12...14 (a 3D graph).
- ❑ `isChartBiPolar()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a bi-polar graph type. For example, `GraphType` 21 (vertical bi-polar clustered bars), 22 (vertical bi-polar stacked bars), and so on.
- ❑ `isChartBLAType()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a bar, line, or area (BLA) graph type (`GraphTypes` 17...54).
- ❑ `isChartDualY()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a dual Y-axes graph type. For example, `GraphType` 19 (vertical dual-axis clustered bars), 20 (vertical dual-axis stacked bars), and so on.
- ❑ `isChartOrientHorz()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a horizontally-oriented graph type. For example, `GraphType` 24 (horizontal clustered bars), 25 (horizontal stacked bars), and so on.
- ❑ `isChartPieType()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a pie graph type (`GraphTypes` 55...60).
- ❑ `isChartScatter()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a scatter graph type (`GraphTypes` 61...64).
- ❑ `isChartStockType()`; Returns true or false indicating whether or not the `GraphType` property is currently set to a stock graph type (`GraphTypes` 70...84 or 88).

How Graphs Are Imaged

Unless otherwise specified, an internal set of default properties determines how graphs are initially imaged. This section describes the default graph size and how the basic graph types are drawn using the default properties.

For more information see:

- ❑ [Graph-Specific Properties and Methods](#) on page 139 for special graph types such as funnel, multi-Y, Pareto, and waterfall.
- ❑ [Data Requirements for Different Graph Types](#) on page 193 for graphs that require special data and settings.

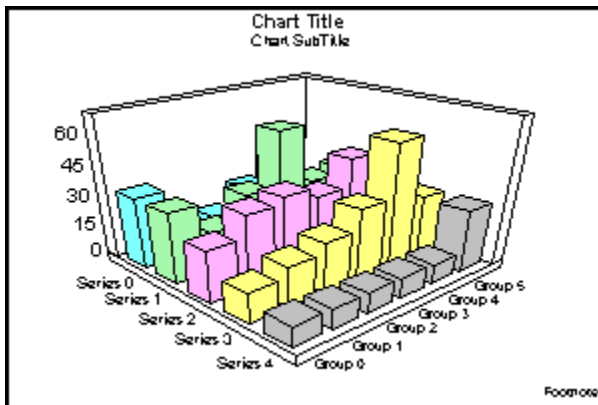
Graph Sizing

When a graph is initially imaged, it is drawn using the WebFOCUS default size (760, 400).



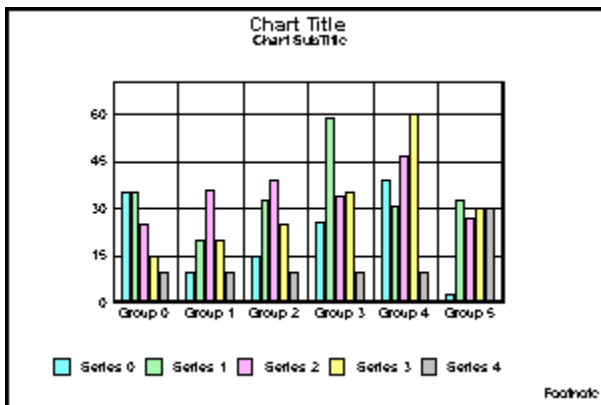
3D Graphs

A 3D graph, is imaged using the standard Viewing3DAnglePreset(1). This images the graph with the labels for the ordinal axes (O1 (groups) and O2 (series)) on the bottom of the graph and labels for the numeric (Y1) axis on the left side.



Single Axis/Vertical Graphs

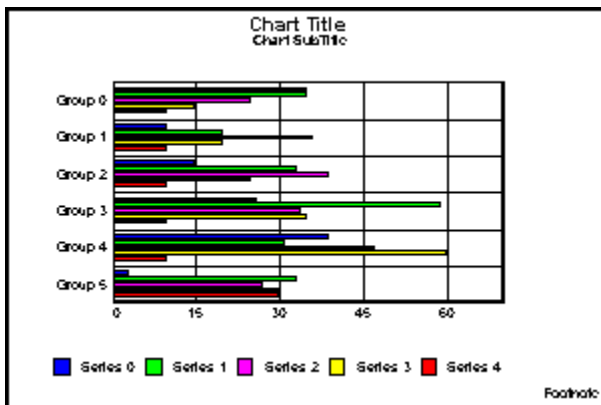
A single axis, vertical graph, is imaged with the ordinal (O1) axis on the bottom of the graph and numeric (Y1) axis on the left side of the graph. The series labels are drawn in the legend area at the bottom of the graph.



For all vertical and horizontal graphs, the default is simple 2-D rendering, which produces a flat image.

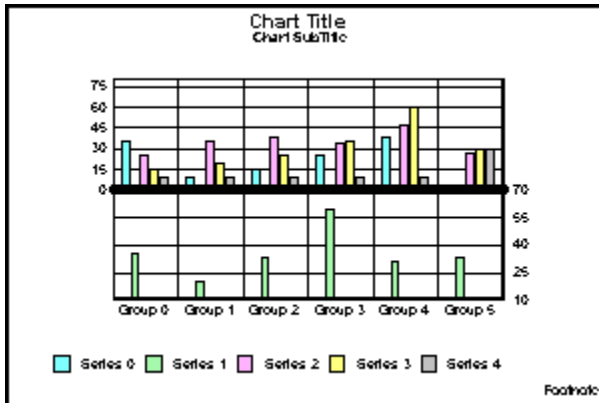
Single Axis/Horizontal Graphs

A single axis, horizontal graph is imaged with the ordinal (O1) axis on the left side of the graph and numeric (Y1) axis on the bottom side of the graph.



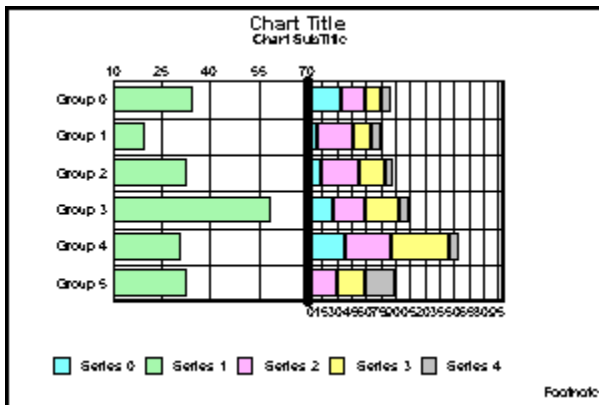
Dual Axes and Bi-Polar Vertical Graphs

In dual axes and bi-polar vertical graphs, the Y1 axis is imaged on the left side of the graph and the Y2 axis is drawn on the right side.



Dual Axes and Bi-Polar Horizontal Graphs

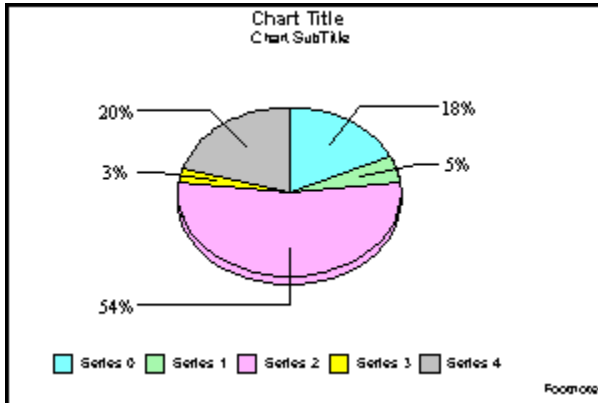
In dual axes and bi-polar horizontal graphs, the Y1 axis is drawn on the bottom of the graph and the Y2 axis is imaged on the top graph.



A dual-axes line is drawn dividing the two halves of the graph. The default dual-axes split position is 50.

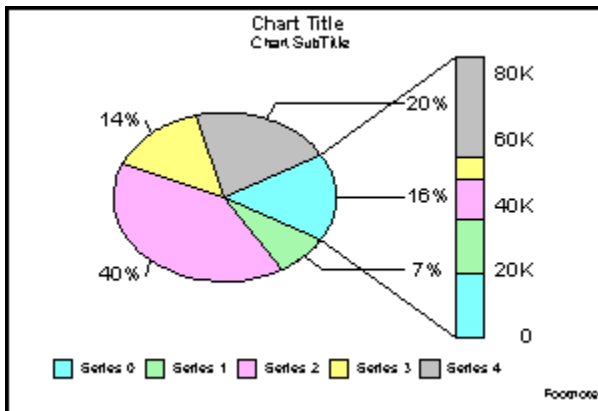
Pie Graphs

A pie graph is drawn with pie feelers and labels and the PieDepth set to 0. If a multiple pie graph type is used, the graph is imaged with three pies per row. The pie feeler text format is set to zero (standard) so that the actual value that makes up the pie slice is displayed. All groups in a series are combined to form a single pie slice.



The graph legend area at the bottom of the graph shows the series labels.

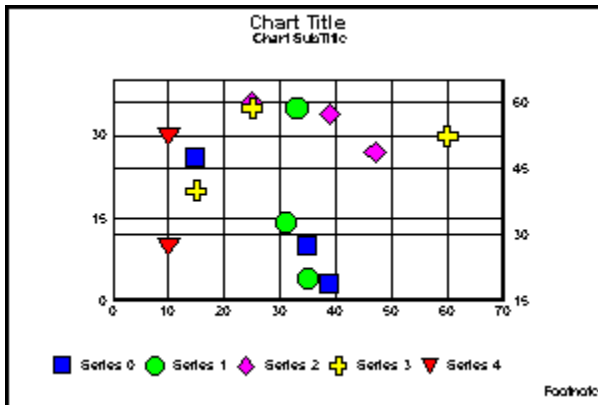
When GraphType(93) is selected, the series is mapped in the pie graph to a bar showing the values associated with each wedge.



You can use the PieBarSeries property to select the series in the pie graph that will be used to map the bar. The default pie bar series is zero.

Scatter Graphs

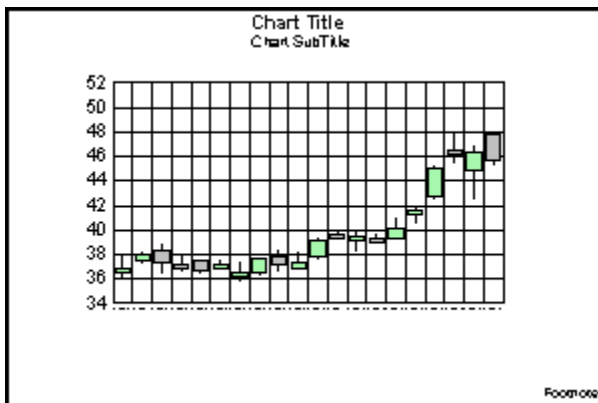
In a scatter graph, the X1 axis is drawn on the bottom of the graph and the Y1 axis is drawn on the left side of the graph. If a dual axes scatter graph is selected, the Y2 axis is drawn on the right side of the graph.



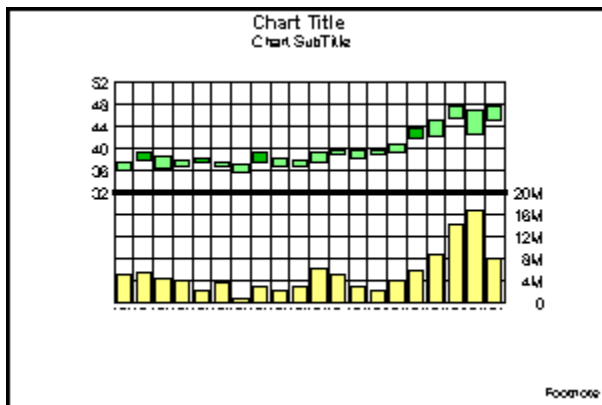
Note: The X-axis for Scatter graphs is expected to be numeric. While you are allowed to substitute with an alphanumeric X-axis, not all features will work correctly.

Stock Graphs

The High/Low Stock, Candle graph is a Japanese style stock graph. Risers are defined by open and close values. Ticks poke out from the top and bottom for high and low values. The bars will be one color if the close value is greater than the open value. Another color is used if the open value is greater than the close.



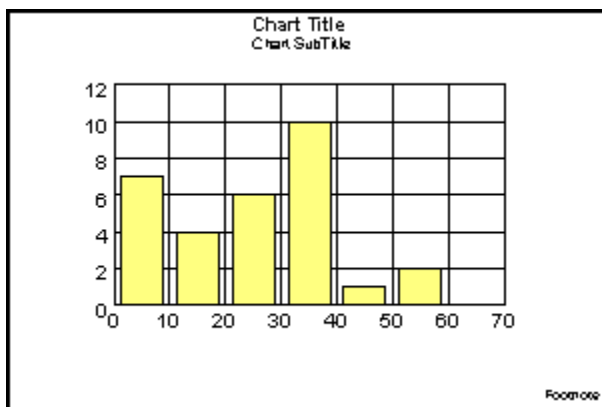
In stock graphs, green risers show high values and blue risers show low values. In single axis stock graphs, the Y1 axis is drawn on the left side of the graph and the O1 axis is drawn on the bottom of the graph. For dual-axes and bi-polar stock graphs, the Y1 axis is drawn on the left side and the Y2 axis is drawn on the right side.



If close values are included in the graph, a line is drawn in the riser at the close value. If volume values are included, the volumes are shown in the bottom of the graph on the Y2 axis and high, low, open, and/or close values are shown in the top of the graph.

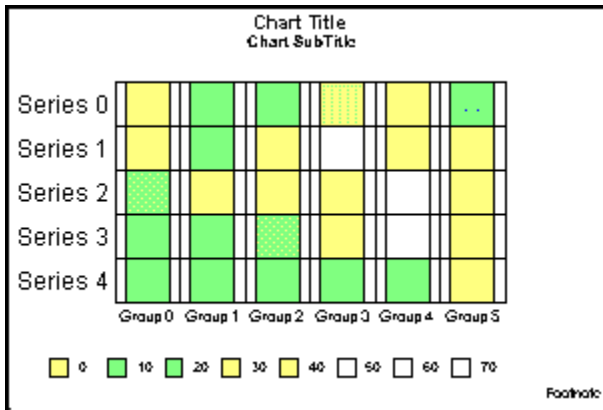
Histograms

The vertical histogram is drawn with the X1 axis at the bottom of the graph and the Y1 axis on the left side as shown in the following illustration. If a horizontal histogram is used, the Y1 axis is drawn at the bottom of the graph and the X1 axis is drawn along the left side.



Spectral Maps

The spectral map graph type is a row or column matrix of markers that are colored according to data values. The default graph with sample data draws the O2 (series) axis labels on the left side of the graph and the O1 (group) axis labels at the bottom of the graph.

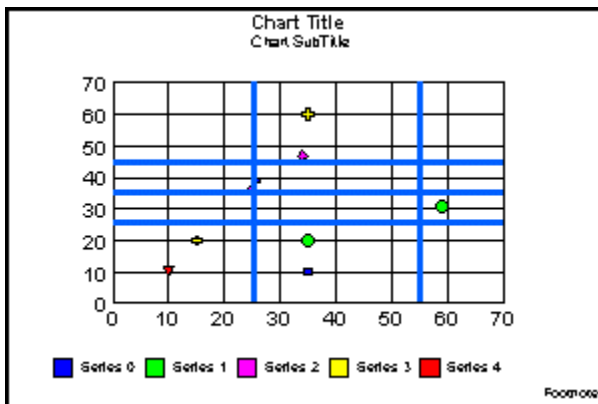


In a spectral map, the risers are colored by height (rather than by series or by group). The graphing engine chooses the color of quantitative data representations based on the data values supplied to the graph. You can use the `getColorByHeight()` and `gradient` methods to modify the gradient.

Bubble Graphs

You can create single or dual axes bubble graphs, with or without labels.

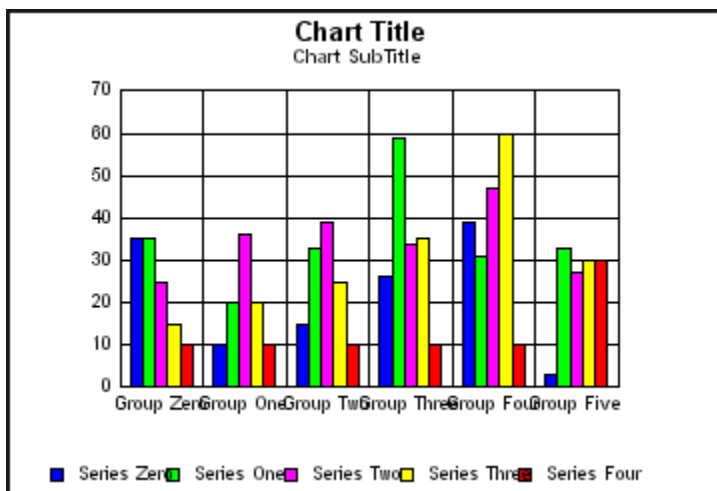
In the default configuration, the Y1 axis labels are drawn on the left side of the graph and the X1 axis labels are drawn at the bottom of the graph. Three quadrant lines are drawn for the Y1 axis and two quadrant lines are drawn for the X1 axis.



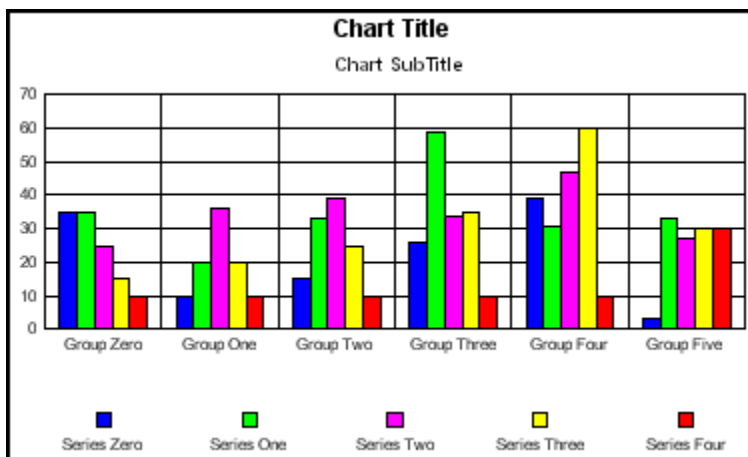
Note: The X-axis for Bubble graphs is expected to be numeric. While you are allowed to substitute with an alphanumeric X-axis, not all features will work correctly.

Chart Layout Manager

When a chart draws a large data set or contains lengthy labels and titles, chart objects may overlap and obscure objects behind them. In the following example, the ordinal axis/group labels and series labels are long and overlap each other.



The chart layout manager can automatically correct this problem with a single property.



Additionally, Perspective includes methods that allow you to enable/disable and define how automatic chart layout is done for individual objects.

The Place property enables/disables the automatic chart layout manager. This functionality allows you to automatically setup the spacing of various elements of a chart. Prior to drawing a chart, certain Perspective methods may be set depending on what the automatic chart layout manager determines the optimum chart to be.

When `setPlace (true)` is used, you can use the following methods to further control automatic placement of chart elements:

- ❑ `setPlaceAlign();` For any text object, allow/disallow horizontal alignment. By default, horizontal alignment is allowed.
- ❑ `setPlaceMargin();` Define the margin to draw around the entire chart. The default value is default new Rectangle (-15500, -15500, 31000, 31000).
- ❑ `setPlaceMaxNumWordWrapLines();` For any text object, define the maximum number of lines that can be wrapped when `setPlaceWordWrap()` is enabled. The default value is 2.
- ❑ `setPlaceMinFontSize();` For any text object, define the minimum font size that can be used by the automatic chart layout manager to resize a text object. The default value is 4 point.
- ❑ `setPlacePosition();` Define how the automatic chart layout manager can position objects. In the current implementation, only the legend object can be repositioned. The default value is zero.
- ❑ `setPlaceRect();` Set the rectangle to be used for initial placement of objects. In the current implementation, only object IDs return by `getPlaceVertO1LabelsTitle()` or `getPlaceHorzO1LabelsTitle()` can be used.
- ❑ `setPlaceResize();` For any chart object, allow/disallow the automatic chart layout manager to move/resize the object. By default, resizing of all chart objects is allowed.
- ❑ `setPlaceSkip();` For primary ordinal axis (O1) labels, allow/disallow the automatic chart layout manager to skip labels. By default, label skipping is allowed.
- ❑ `setPlaceTruncateCount();` For any text object, set the number of characters that can be truncated.
- ❑ `setPlaceWordWrap();` For any text object, allow/disallow word wrapping. By default, word wrapping is allowed.

Setting the Color of the Page Surrounding the Graph

You can use the PAGECOLOR StyleSheet attribute to set the color of the HTML page that surrounds the image directly from the graph procedure.

Syntax: **How to Set the Color of the Page Surrounding the Graph**

```
[TYPE=REPORT,] ... PAGECOLOR={colorcolor|RGB(r g b)|RGB(#hexcolor)},$
```

where:

TYPE=REPORT

Is optional. If omitted, TYPE defaults to REPORT.

color

Is a supported color. For a list of supported colors see the *Creating Reports With WebFOCUS Language* manual.

RGB{*r g b*}

Specifies the font color using a mixture of red, green, and blue.

(*r g b*) is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. Note that using the three color components in equal intensities results in shades of gray.

RGB(*#hexcolor*)

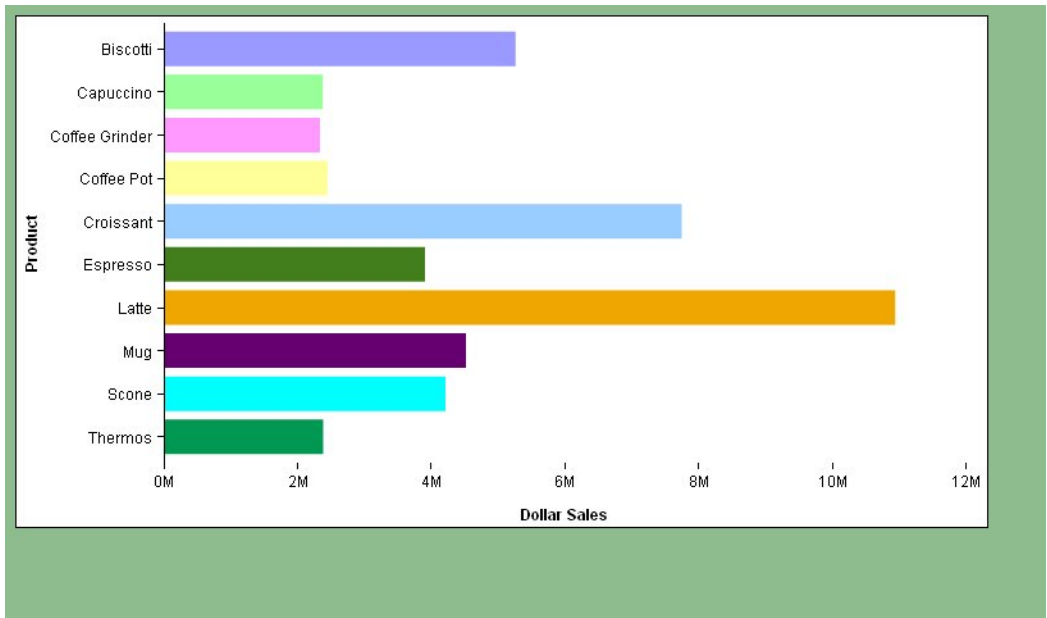
Is the hexadecimal value for the color. For example, FF0000 is the hexadecimal value for red. The hexadecimal digits can be in upper or lower case and must be preceded by a pound sign (#).

Example: **Setting the Color of the HTML Page**

The following request against the GGSALES data source sets the color of the page to pale green.

```
GRAPH FILE GGSALES
SUM DOLLARS
BY PRODUCT
ON GRAPH PCHOLD FORMAT JSCHART
ON GRAPH SET STYLE *
PAGECOLOR=PALE GREEN,$
END
```


The resulting chart is surrounded by a pale green HTML page:



Graph-Wide Parameters

These properties and methods control the overall appearance of the graph:

☐ BiDirectional (true/false)

Reverses the text alignment of all objects and the location of legend text next to label markers. This property is provided for international support only. It can be used to make the entire graph read from right-to-left (`BiDirectional(true)`) or left-to-right (`BiDirectional(false)`). However, note that `BiDirectional(true)` does not reverse the values of the associated properties and methods. They retain their original values. For example, `getTextJustHoriz()` would return zero (left) even though visually the text is aligned right by `BiDirectional(true)`. Use this property with great care. Your application must keep track of the `BiDirectional` setting and handle the reverse values supplied by associated properties and methods.

☐ GraphType (0...106)

Selects the type of graph (for example, 3D, vertical bar, pie, stock, and so on) to be drawn. [Graph Types and 3D Preset Viewing Angles](#) on page 1213 includes an example of each of the supported graph types. Note that a few graph type values are reserved for future use, so no graph will be produced if those values are selected.

☐ UseOffScreen (true/false)

Controls whether the graph will be drawn directly to the graphics device (true) or using an off-screen image buffer (false). Set this property to false when the drawing speed of the graph seems very slow. When this property is set to false, it can increase the drawing speed by a factor of three in some situations.

Graph Frame Attributes

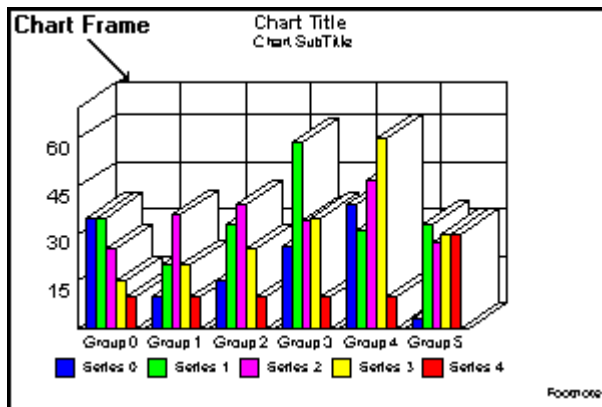
The graph frame is the outer edge of the graph on which grid lines and risers (bar, area, markers) are drawn.

These properties and methods define how the graph frame is imaged:

☐ FrameDisplay (true/false) enables (true)/disables (false) the display of the graph frame text string. True is the default value.

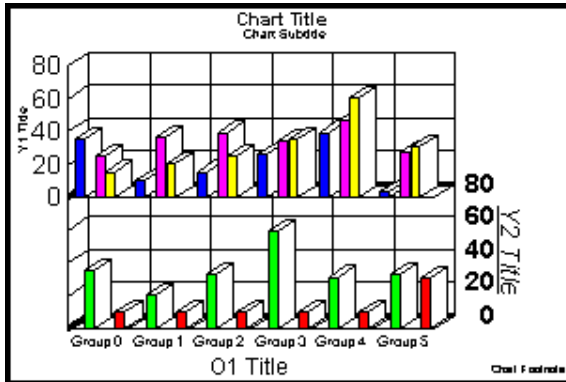
☐ DepthAngle (0...360) and DepthRadius (0...100)

To draw a graph frame with a 2.5D effect as shown here, use the DepthAngle and Depth Radius properties to set the angle and radius of the 2.5D effect. This example graph frame was drawn with the default values DepthAngle(45) and DepthRadius(25).



Axis Attributes

Depending on the graph type, the graph may include one or two ordinal axes (O1 and O2) and multiple numeric axes (X1, Y1, Y2, Y3, Y4, Y5). The following illustration shows a graph with one ordinal axis (O1) and two numeric axes (Y1 and Y2). Note that the secondary ordinal axis (O2) only appears in 3D graphs.



- ☐ An O1 (group or category) axis is included in all graph types except bubble, histogram, and scatter graphs.
- ☐ An O2 (series) axis is only included in 3D graphs.
- ☐ An X1 axis is only included in bubble, histogram, and scatter graphs.
- ☐ A Y1 axis is included in all graph types except pie.
- ☐ A Y2 axis is only included in dual-axes graphs.
- ☐ A Y3, Y4, and/or Y5 axis is only included in the special multi-Y axis graphs (GraphTypes 103...105).

See [Time Scale Axis Graphs](#) on page 221 for information about how to set up a time scale axis.

Axis Properties

This table shows the default values for properties that affect the axes and their grid lines, labels, titles, and so on. Blank cells indicate the property is not supported or applicable for the axis.

Property	O1 Axis	O2 Axis (see note)	X1 Axis	Y1 Axis	Y2 Axis
AxisDescending			false	false	false

Property	O1 Axis	O2 Axis (see note)	X1 Axis	Y1 Axis	Y2 Axis
AxisLineDisplay	true		true	true	true
AxisSide	0		0	see note	see note
ExcludeMaxLabel	false	false	false	false	false
ExcludeMinLabel	false	false	false	false	false
LabelAutofit	true	true	true	false	true
LabelAutoskip	0				
LabelDisplay	true	true	true	true	true
LabelFormat			0	0	0
LabelFormatPattern			"#.#"	"#.#"	"#.#"
LabelMargin	0				
LabelRotate	0	0	0	0	0
LabelSkipBegin	0				
LabelSkipCount	0				
LabelStagger	false		false	false	false
LogScale			false	false	false
LabelWrap	false	false			
MajorGridDisplay	true	*	true	true	true
MajorGridStep			10.0	10.0	10.0
MajorGridStepAuto			true	true	true
MajorGridStyle	0		0	0	0
MinorGridCount	1				

Property	O1 Axis	O2 Axis (see note)	X1 Axis	Y1 Axis	Y2 Axis
MinorGridDisplay	false		false	false	false
MinorGridStep			10.0	10.0	10.0
MinorGridStepAuto			true	true	true
MinorGridStyle	0		0	0	0
MustIncludeZero			false	false	false
OffScaleDisplay			true	true	true
ScaleMax			100.0	70.0	80.0
ScaleMaxAuto			true	true	true
ScaleMin			0.0	0.0	0.0
ScaleMinAuto			true	true	true
TitleAutofit	true	true	true	true	true
TitleDisplay	false	false	false	false	false
TitleString	Null String	Null String	Null String	Null String	Null String
ZeroLineDisplay			false	false	false

Note:

- ❑ The O2-axis only appears in 3D graphs and spectral maps. Grid3DFloorDisplayX/Z, Grid3DLeftWallDisplayY/Z, Grid3DRightWallDisplayX/Y, and Grid3DRiserDisplayX/Y/Z enable/disable the display of grid lines in a 3D graph.
- ❑ For dual (Y1/Y2) axes graphs, the default AxisSide is 0 (left) for the Y1 axis and 1 (right) for the Y2-axis. For graphs with more than two Y-axes, the default AxisSide is 0 (left) for odd numbered axes (Y1, Y3, Y5) and 1 (right) for even numbered axes (Y2, Y4).

Ordinal Axis Properties

Use these properties to define how ordinal axes are imaged in a graph:

- ❑ `O1AxisLineDisplay` (true/false); Enables (true)/disables (false) the display of the base line on the primary ordinal axis.
- ❑ `O1AxisSide` (0...3); Controls which side of the graph the ordinal axis is imaged. Although the vast majority of all graphs will image their ordinal text on the bottom of a vertical graph or left of a horizontal graph, it is possible to image it to the top/right or both sides. 1 is the default value (bottom for vertical graphs/left for horizontal graphs).

See [Grid Lines](#) on page 87 and [Labels and Titles](#) on page 117 for additional information about how to control the display of labels and titles on an ordinal axis.

Ordinal Axis Methods

Use these methods to determine if the graph includes the primary or secondary ordinal axis:

- ❑ `isO1AxisPresent()`; Returns true or false indicating whether or not the primary ordinal axis is present in the graph.
- ❑ `isO2AxisPresent()`; Returns true or false indicating whether or not the secondary ordinal axis is present in the graph.

These properties indicate whether or not the axis is present regardless of whether or not the axis base line is enabled for display. The primary ordinal axis is present in most graph types. The secondary ordinal axis is only present in 3D graph types.

Numeric Axis Properties

Use these properties to define how numeric axes are imaged in the graph:

- ❑ `X1/Y1/Y2/Y3/Y4/Y5 AxisDescending` (true/false)

The `AxisDescending` properties change the direction of values displayed on the axis to descending or ascending. True is the default value (ascending values).

- ❑ `X1/Y1/Y2AxisLineDisplay` (true/false)

The `AxisLineDisplay` properties enable (true)/disable (false) the display of the bottom line on an axis.

- ❑ `X1/Y1/Y2AxisSide` (0...2)

The AxisSide properties control the side of the graph where the axis is imaged. For vertical graphs, the default settings draw the Y1 axis on the left side of the graph and the Y2 on the right. For horizontal graphs, Y1 is drawn on the bottom of the graph and Y2 on the top.

❑ X1/Y1/Y2OffScaleDisplay (true/false)

The OffScaleDisplay properties determine whether or not values out of range are imaged on the axis.

❑ X1/Y1/Y2/Y3/Y4/Y5LogScale (true/false)

The LogScale properties specify the scale (logarithmic or linear) for the numeric axis. False is the default value (linear).

❑ X1/Y1/Y2ScaleMax (Double), X1/Y1/Y2ScaleMaxAuto (true/false), X1/Y1/Y2ScaleMin (Double), and X1/Y1/Y2ScaleMinAuto (true/false)

The ScaleMin and ScaleMax properties define the range (minimum and maximum) of values that will be imaged on the axis. The ScaleMinAuto and ScaleMaxAuto properties tell the graphing engine to automatically calculate the minimum and maximum values. If these properties are enabled (true), values set by ScaleMin and ScaleMax are ignored.

❑ X1/Y1/Y2/Y3/Y4/Y5ZeroLineDisplay (true/false)

The ZeroLineDisplay properties enable (true)/disable (false) the drawing of a zero line on an axis. True is the default value.

❑ X1/Y1/Y2MustIncludeZero (true/false)

The MustIncludeZero properties determine whether or not the numeric axes include zero. False is the default value.

See [Labels and Titles](#) on page 117 for additional information about how to control the display of labels and titles on a numeric axis.

Numeric Axis Methods

Use these methods to get information about specific numeric axes in a graph:

- ❑ getX1MajorGridStepAutoValue(); If major grid steps are automatically calculated, this method returns the grid step value for the X-axis.
- ❑ getX1ScaleMaxAutoValue(); If the maximum scale value is automatically calculated, this method returns the maximum scaling value that will be used on the X-axis.
- ❑ getX1ScaleMinAutoValue(); If the minimum scale value is automatically calculated, this method returns the minimum scaling value that will be used on the X-axis.

- ❑ `getY1MajorGridStepAutoValue()`; If major grid steps are automatically calculated, this method returns the grid step value for the Y1-axis.
- ❑ `getY1ScaleMaxAutoValue()`; If the maximum scale value is automatically calculated, this method returns the maximum scaling value that will be used on the Y1-axis.
- ❑ `getY1ScaleMinAutoValue()`; If the minimum scale value is automatically calculated, this method returns the minimum scaling value that will be used on the Y1-axis.
- ❑ `getY2MajorGridStepAutoValue()`; If major grid steps are automatically calculated, this method returns the grid step value for the Y2-axis.
- ❑ `getY2ScaleMaxAutoValue()`; If the maximum scale value is automatically calculated, this method returns the maximum scaling value that will be used on the Y2-axis.
- ❑ `getY2ScaleMinAutoValue()`; If the minimum scale value is automatically calculated, this method returns the minimum scaling value that will be used on the Y2-axis.
- ❑ `isX1AxisPresent()`; Returns true or false indicating whether or not the X-axis is present in the graph.
- ❑ `isY1AxisPresent()`; Returns true or false indicating whether or not the primary numeric axis is present in the graph.
- ❑ `isY2AxisPresent()`; Returns true or false indicating whether or not the secondary numeric axis is present in the graph.
- ❑ `isZ1AxisPresent()`; Returns true or false indicating whether or not the Z-axis is present in the graph.

These methods indicate whether or not the axis is present regardless of whether or not the axis base line is enabled for display. The X1-axis is present in scatter graphs, bubble graphs, histograms, and so on. The Y1-axis is present in most graph types. The Y2-axis is only present in dual-axes graphs.

Axis Independent Methods

In addition to the axis-specific properties and methods, you can use these methods to get/set axis attributes.

- ❑ `get/setAxisAssignment()`; Get/set the axis assignment to determine where a series is assigned to an axis.
- ❑ `get/setAxisDescending()`; Get/set the order (ascending or descending) in which an axis is drawn.

- ❑ `get/setAxisOffset()`; Get/set axis offset in multi-Y graphs.
- ❑ `get/setAxisSide()`; Get/set the side of the graph where an axis is imaged.
- ❑ `get/setAxisSize()`; Get/set the size of an axis in multi-Y graphs.
- ❑ `get/setDisplayOffscale()`; Get/set whether or not off-scale values are drawn on an axis.
- ❑ `get/setLogScale()`; Get/set whether the axis uses linear or logarithmic scaling.
- ❑ `get/setLogScaleBase()`; Get/set the scale base when logarithmic scaling is used on a numeric axis.
- ❑ `get/setScaleMax()`; Get/set the maximum value imaged on an axis.
- ❑ `get/setScaleMaxAuto()`; Get/set automatic calculation of the ScaleMax value.
- ❑ `get/setScaleMin()`; Get/set the minimum value imaged on an axis.
- ❑ `get/setScaleMinAuto()`; Get/set automatic calculation of the ScaleMin value.
- ❑ `get/setScaleMustIncludeZero()`; Get/set whether or not the axis must include a zero value.

When you use these methods, you may supply the object ID of an axis as an input parameter to identify the axis where the attribute is applied. If an axis is not specified, the attribute is retrieved from or assigned to the first item in the selection list. If the object ID does not identify an axis or the first item in the selection list is not an axis, a set operation will apply the attribute to the object but it will not affect the appearance of the graph.

Tick Mark Properties and Methods

Tick marks can be enabled on any axis independent of grid lines.

The following properties control drawing of tick marks on the primary ordinal axis:

```
IgnoreTickSkip
01MajorTickDisplay
01MajorTickStyle
01MinorTickCount
01MinorTickDisplay
01MinorTickStyle
```

The following properties control drawing of independent tick marks on a numeric axis:

X1/Y1/Y2MajorTickDisplay
X1/Y1/Y2MajorTickStep
X1/Y1/Y2MajorTickStepAuto
X1/Y1/Y2MajorTickStyle
X1/Y1/Y2MinorTickDisplay
X1/Y1/Y2MinorTickStep
X1/Y1/Y2MinorTickStepAuto
X1/Y1/Y2MinorTickStyle

Note that tick marks can be drawn on Y3-, Y4-, and Y5-axes using the methods listed below.

The following methods can be used to enable and control the appearance of independent tick marks on O1-, X1- and all Y-axes:

- ❑ `get/setTickCount()`; Get/set the number of minor ticks to draw between major ticks.
- ❑ `get/setTickLength()`; Get/set the length of tick marks in virtual coordinates.
- ❑ `get/setTickStep()`; Get/set the number of tick steps on an axis when `setTickStepAuto(objectID, FALSE)` is used.
- ❑ `get/setTickStepAuto()`; Enable/disable automatic calculation of the number of ticks.
- ❑ `get/setTickStyle()`; Get/set tick style (inside, outside, or spanning) the axis baseline.
- ❑ `getTickWidth()`; Get/set the width of tick marks in virtual coordinates.
- ❑ `get/setDrawTicksPerpendicular()`; Force tick marks to be perpendicular to axis base line or not.

When these methods are used, the object ID of tick marks may be supplied as an input parameter to identify the axis tick marks to which the attribute is applied. If a tick mark object is not specified, the attribute is retrieved from or assigned to the first item in the selection list. If the object ID does not identify a tick mark object or the first item in the selection list is not a tick mark object, a "set" operation will apply the attribute to the object but it will not have any effect on the appearance of the chart.

Properties for Dual Axes Graphs

These properties control the axes in dual-axes graphs:

- ❑ `DualAxisLineDisplay (true/false)`; Enables (true)/disables (false) the display of a line that separates the two halves of a dual axes graph.
- ❑ `DualAxisSplitPosition (0...100)`; Sets the position within the graph frame where the dual-Y split position will be created.

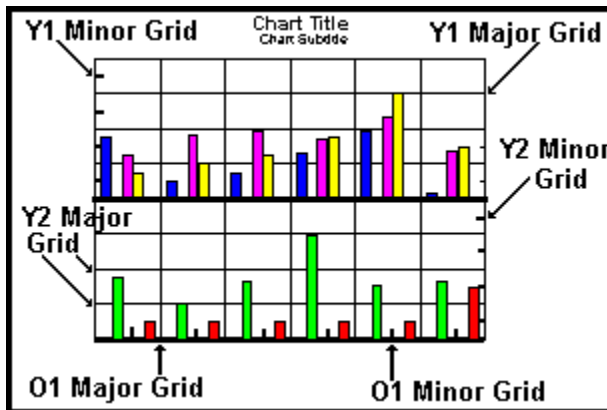
Autofitting Axis Text

Use these properties to define how axis text is autofitted:

- ❑ `AxisTextAutofitMax ( value )`; Defines the maximum size of axis text in virtual coordinates.
- ❑ `AxisTextAutofitMin ( value )`; Defines the minimum size of axis text in virtual coordinates.
- ❑ `AxisTextAutofitMode ( 0...2 )`; Defines the mode to be used for autofitting axis text.
- ❑ `AxisTextAutofitPercent ( 0...100% )`; When `AxisTextAutofitMode` is set to 2, this property defines the percent to be used for autofitting axis text. All axis text is maintained within `AxisTextAutofitPercent` of the axis with the smallest font size as determined by Autofit.

Grid Lines

Each axis in a graph can include or exclude major and minor grid lines. Properties and methods also allow you to specify the number and style of grid lines displayed on each axis. The following illustration shows a bi-polar graph where major and minor grid lines have been enabled on the ordinal (O1) axis and both numeric (Y1 and Y2) axes.



Ordinal Axis Grid Lines

Use these properties to control the grid lines on the ordinal axis:

- ❑ `O1MajorGridDisplay (true/false)`
- ❑ `O1MajorGridStyle (0...5)`
- ❑ `O1MinorGridCount (0...50)`
- ❑ `O1MinorGridDisplay (true/false)`

❑ O1MinorGridStyle (0...5)

These properties control the number, display, and style of ordinal axis major and minor grid lines. Major and minor grid lines can appear in one of the following styles:

- 0= No grid lines.
- 1= Normal grid lines, height of frame.
- 2= Normal grid lines extended beyond the height of frame.
- 3= Small tick marks from frame edge inward.
- 4= Small tick marks from frame edge outward.
- 5= Ticks span across the frame edge.

All graphs can include major and minor grid lines. In the default configuration, major grid lines are drawn and minor grid lines are not drawn. Different styles can be assigned to major and minor grid lines so that they are easily distinguished in the graph. Use of the WebFOCUS grid setting (SET GRID = ON) displays major grids for both X- and Y- axes.

Numeric Axis Grid Lines

Use these methods to control the appearance of grid lines on the numeric axes in a graph.

- ❑ X1/Y1/Y2MajorGridDisplay (true/false)
- ❑ X1/Y1/Y2MajorGridStyle (0...5)
- ❑ X1/Y2/Y2MajorGridStep (double)
- ❑ X1/Y1/Y2MajorGridStepAuto (true/false)
- ❑ X1/Y1/Y2MinorGridDisplay (true/false)
- ❑ X1/Y1/Y2MinorGridStyle (0...5)
- ❑ X1/Y1/Y2MinorGridStep(double)
- ❑ X1/Y1/Y2MinorGridStepAuto (true/false)

The GridDisplay properties control the display of major and minor grid lines on the numeric axes. True is the default value for major grid lines and false is the default value for minor grid lines. The GridStyle properties select the style of numeric axis grid lines:

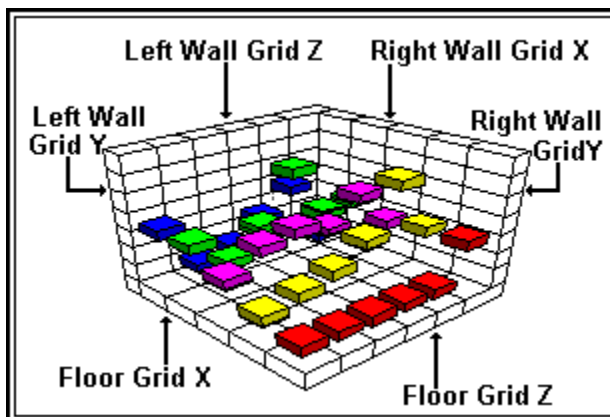
- 0= No grid lines.
- 1= Normal grid lines, height of frame.
- 2= Normal grid lines extended beyond the height of frame.
- 3= Small tick marks from frame edge inward.
- 4= Small tick marks from frame edge outward.
- 5= Ticks span across the frame edge.

Typically different styles are used for major and minor grid lines so that they are easily distinguished in the graph.

The GridStep and GridStepAuto properties control the number of numeric axis grid lines according to the number of values displayed on an axis. If the GridStepAuto property is enabled, the number of grid steps is automatically calculated and the value assigned to the associated GridStep property will be ignored. If the GridStepAuto property is disabled, the GridStep property controls the number of major or minor grid lines that will be imaged. For example, grid steps 1, 6, 11, 16...51 will be imaged if the range of values in the graph is 1...51 and this property is set to 5. If this property is set to 10, grid steps 1, 11, 21, 31...51 would appear.

3D Grid Lines

Grid lines can appear on the floor and walls of the cube in a 3D graph.



The following properties enable and disable these grid lines:

- ☐ Grid3DFloorDisplayX/Z (true/false); Shows/hides X- and Z-axis grid lines on the floor of the cube.

- ❑ `Grid3DLeftWallDisplayY/Z` (true/false); Shows/hides Y- and Z-axis grid lines on the left wall of the cube.
- ❑ `Grid3DRightWallDisplayX/Y` (true/false); Shows/hides X- and Y-axis grid lines on the right wall of the cube.

Grids can also appear on risers in 3D graphs using these properties:

- ❑ `Grid3DRiserDisplayX` (true/false); Enables/disables the display of X-axis grid lines on risers in a 3D-surface graph. It is only applicable when a 3D-surface graph is selected.
- ❑ `Grid3DRiserDisplayY` (true/false); Enables/disables the display of Y-axis grid lines on risers in a 3D graph.
- ❑ `Grid3DRiserDisplayZ` (true/false); Enables/disables the display of Z-axis grid lines on risers in a 3D-surface graph. It is only applicable when a 3D-surface graph is selected.

Axis Independent Methods for Grids

In addition to the axis-specific grid properties, you can use these methods to get/set grid attributes:

- ❑ `get/setGridCount()`; Get/set the number of grid lines on an axis.
- ❑ `get/setGridStep()`; Get/set the number of grid steps on an axis.
- ❑ `get/setGridStepAuto()`; Get/set whether grid steps are automatically calculated on an axis.
- ❑ `get/setGridStyle()`; Get/set the style of grid lines to be imaged on an axis.

When you use these methods, you may supply the object ID of a grid as an input parameter to identify the grid to which the attribute is applied. If a grid is not specified, the attribute is retrieved from or assigned to the first item in the selection list. If the object ID does not identify a grid object or the first item in the selection list is not a grid object, a set operation will apply the attribute to the object but it will not have any effect on the appearance of the graph.

Curve Fit Lines

The following properties and methods are available for drawing curve fit lines across one or more series of risers:

- ❑ `get/setCurveFitType()`; These methods get/set a curve fit type (for example, linear, quadratic, polynomial, and so on). The `setCurveFitType()` method enables the drawing of the curve fit line.

- ❑ `get/setCurveFitPolynomialOrder()`; If `setCurveFitType()` selects a polynomial fit line, these methods `get/set` the order for the polynomial fit.
- ❑ `CurveFitEquationDisplay`; This property enables/disables the drawing of the equation associated with a curve fit line.
- ❑ `CurveFitHighOrderFirst`; When the `CurveFitEquationDisplay` property is true, this property selects the format of the equation.
- ❑ `CurveFitNumSegments`; This property defines the number of segments to use in drawing a curve fit line.

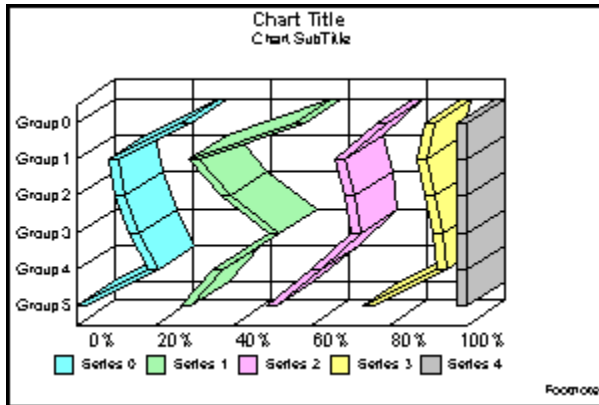
Java 2D Line Objects

- ❑ When Java 2D line objects are enabled with `setBasicStroke(1...5)`, the following methods can be used to format lines in a chart:
- ❑ `get/setLineBasicStrokeType()`; Defines a line's basic stroke type (0 = JDK 1.1.8, 1...5 selects one of the Java 2D `BasicStroke` constructors).
- ❑ `getLineCap()`; Defines the decoration applied to the ends of unclosed subpaths and dash segments of a specified line as defined by the Java 2D `BasicStroke` object (for example, `CAP_BUTT`, `CAP_ROUND` or `CAP_SQUARE`).
- ❑ `get/setLineDash()`; These methods define a dash pattern for a specified line object.
- ❑ `getLineDashFloat()`; Returns the dashing pattern as an array of floating point numbers.
- ❑ `get/setLineDashPhase()`; When a line dashing pattern has been specified with `setLineDash()`, these methods define the offset to start the dashing pattern for a specified line object.
- ❑ `get/setLineJoin()`; Defines the decoration applied where line path segments meet (for example, `JOIN_ROUND`, `JOIN_BEVEL`, or `JOIN_MITER`).
- ❑ `get/setLineMiterLimit()`; When `setLineJoin()` applies `JOIN_MITER` to a line object, these methods define the limit to trim the miter join.

Also see the `java.awt.BasicStroke` constructors for more information about these methods.

Risers and Markers

The risers and markers in a graph are initially determined by the graph type that is selected with the `GraphType` property. For example, `GraphType 17` selects a vertical bar graph and the risers are drawn in the graph as vertical bars that are auto-shaded and imaged with a 2.5D effect by default. `GraphType 54` selects a horizontal line graph in which risers are drawn as lines that are auto-shaded and imaged with a 2.5D effect by default.



You can use properties and methods to change how risers and markers are imaged in the graph. For more information about risers and markers in pie and stock graphs, see [Pie Graph Properties and Methods](#) on page 141 and [Stock Graph Properties and Methods](#) on page 142.

Riser and Marker Properties

These properties control risers and markers in a graph:

- ☐ `ConnectLineMarkers` (true/false). In line graphs where a 2.5D effect is not applied to the graph, the line risers are displayed as markers. Use this property to enable (true)/disable (false) the display of a line that connects the markers.
- ☐ `ConnectScatterMarkers` (true/false). In a scatter graph, use this property to enable (true)/disable (false) the display of connecting lines between markers.
- ☐ `CubeSquareRisers` (true/false). Use this if you want the risers/markers in a graph to always be square regardless of the shape and size of a 3D cube (for example).
- ☐ `DataLineThickness` (1...100). For a 2.5D effect graph, this property defines the thickness (in pixels) of the fake 3D line.
- ☐ `DisplayBarAsPictograph` (true/false). When a texture file has been loaded using `setTextureURL()`, you can use this property to enable the texture file image on bar risers.

- ❑ **MarkerDisplay** (true/false). When lines are drawn between markers in a line or scatter graph, use this property to remove the markers from the graph and only display the connecting lines.
- ❑ **MarkerSizeDefault** (0...100). Use this property to define the default size of markers in a 2D graph. See the `get/setMarkerSize` methods if you want to change the size of markers.
- ❑ **ReverseGroups** (true/false), **ReverseSeries** (true/false), **SeriesAreRows** (true/false). These properties can be used to control how data is assigned to risers and markers in a graph.
- ❑ **Riser3DThicknessY** (0...100), **RiserBarGroupSpacing** (0...100), **RiserWidth** (0...100). These properties can be used to control the size of bar risers in 3D and 2D graphs.
- ❑ **RiserBorderMode** (0...2). Determines when riser borders are drawn (always, never, or only for small data sets (fewer than 900 risers)).
- ❑ **ScaleFromZero** (true/false). In applications where the data range can include negative numbers, this property determines whether the risers will be drawn pointing up and down from a zero line or whether all risers will draw straight up from the bottom of the graph.
- ❑ **SeriesDefaultBorderColor**(color), **SeriesDefaultTransparentBorderColor** (true/false), **UseSeriesBorderDefaults** (true/false). When **UseSeriesBorderDefaults** is enabled (true), riser borders will be drawn with the color defined with the **SeriesDefaultBorderColor** property if **SeriesDefaultTransparentBorderColor** is false. If **SeriesDefaultTransparentBorderColor** is true and **UseSeriesBorderDefault** is enabled, transparent borders are drawn around riser edges.

Riser and Marker Methods

In addition to the riser-specific properties, you can use these methods to `get/set` riser and marker attributes:

- ❑ `get/setMarkerShape()`; Get/set the shape of markers in a series.
- ❑ `get/setMarkerSize()`; Get/set the size of markers in a series or series and group.
- ❑ `get/setSeriesType()`; Get/set the type (bar, area, or marker) of a series of risers.

You can use the `setMarkerShape()` method to set the shape of markers in a graph. Select from the following predefined shapes:

Marker Shape Number	Description	Image
1	Square	

Marker Shape Number	Description	Image
2	Circle	
3	Diamond	
4	Plus	
5	Triangle Down	
6	Triangle UP	
7	Pirate Plus	
8	Triangle Right	
9	Triangle Left	
10	House	
11	Hexagon	
12	Fat X	
13	Five Star	

Marker Shape Number	Description	Image
14	Six Star	
15	Hourglass	
16	Sideways Hourglass	
17	Horizontal Line	
18	Vertical Line	
19	Area	
20	CirclePlus	
21	CircleMinus	

You can use the `setSeriesType()` method to assign a riser type to individual series. This method allows you to use different types of risers for series in a graph. For example, series one could use bar risers, series two could use line markers, and series three could use area risers.

When you use these methods, you may supply the object ID of a riser or marker as an input parameter to identify the object to which the attribute is applied. If a riser or marker is not specified, the attribute is retrieved from or assigned to the first item in the selection list. If the object ID does not identify a riser or marker object or the first item in the selection list is not a riser or marker, a set operation will apply the attribute to the object but it will not have any effect on the appearance of the graph.

Exceptional Risers

You can use the following methods to define the use of exceptional/highlighted risers in a graph. This is how conditional styling is implemented behind the scenes.

- ❑ `get/setExceptionalRiser();` Gets/sets exceptional risers at a particular series and group.
- ❑ `getExceptionalRisers();` Determines if there are any exceptional risers in a graph.
- ❑ `isExceptionalAllowed();` Determines if exceptional risers are allowed in a graph.
- ❑ `setNoExceptionalRiser();` Disallows exceptional risers in a graph.

Riser Automatic Color Rendering/Styling

There are two methods you can use to automatically render the color and style of graph risers:

- ❑ **Repeated Shading.** You can automatically shade successive groups of risers by lightening or darkening the colors used for the risers in the previous group. This type of shading is implemented using color mode 4, a looping index, and a factor for lightening or darkening the colors.
- ❑ **Gradient Coloring.** You can automatically create a gradient effect between the colors for the first and last risers by defining a starting color and ending color for the risers. This type of coloring is implemented using color mode 5, a starting color, and an ending color.

Syntax: **How to Implement Automatic Repeated Riser Shading**

You can automatically shade groups of risers by lightening or darkening the color values in each successive group. You implement this type of shading by setting the color mode to 4, setting a looping index equal to the number of risers in each group, and setting an automatic color lightening or darkening factor:

```
setColorMode(4);  
setSeriesLooping(s);  
setAutoColorLerpFactor(f);
```

where:

```
setColorMode(4);
```

Implements automatic repeated riser shading.

```
s
```

Identifies the number of risers in each group on the graph.

f

Is a double-precision number between -1 and 1 that represents the factor for lightening or darkening each successive group of risers. A positive factor specifies that each successive group of risers will be lighter than the previous group. A negative number specifies that each successive group of risers will be darker than the previous group. Zero indicates that each successive group of risers will be shaded exactly the same as the previous group.

Example: Implementing Automatic Repeated Riser Shading in a Pie Chart

The following request implements repeated lightening of groups of risers. It sets the color mode to 4, the series looping index to 5, and the automatic color factor to 1:

```
-DEFAULTH &WF_STYLE_HEIGHT='405';
-DEFAULTH &WF_STYLE_WIDTH='770';
GRAPH FILE EMPLOYEE
-* Created by Info Assist for Graph
SUM EMPLOYEE.PAYINFO.SALARY
BY EMPLOYEE.EMPINFO.LAST_NAME
WHERE EMPLOYEE.EMPINFO.LAST_NAME NE 'SMITH';
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET VZERO OFF
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET UNITS PIXELS
ON GRAPH SET LOOKGRAPH PIEMULTI
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 1
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
setDepthRadius(5);
setCurveFitEquationDisplay(false);
setDisplay(getDataText(0), false);
.
.
.
setDisplay(getDataText(63), false);
setPieDepth(70);
setPieFeelerTextDisplay(1);
setPlace(true);
```

```

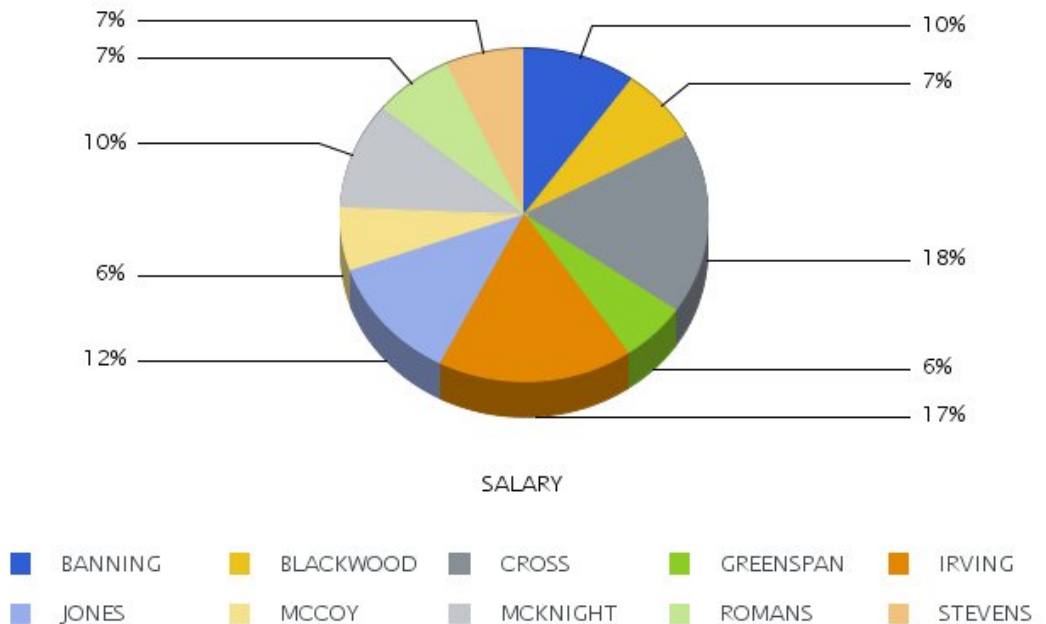
-*Autoshade base on every 5 series setSeriesLooping(5);
setColorMode(4);  setAutoColorLerpFactor(1);  -*

*END
INCLUDE=IBFS:/CFG/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END

```

On the output, the riser colors repeat after five car values because the looping index is 5:

- ❑ The riser color for JONES is a lighter version of the riser color for BANNING.
- ❑ The riser color for MCCOY is a lighter version of the riser color for BLACKWOOD.
- ❑ The riser color for MCKNIGHT is a lighter version of the riser color for CROSS.
- ❑ The riser color for ROMANS is a lighter version of the riser color for GREENSPAN.
- ❑ The riser color for STEVENS is a lighter version of the riser color for IRVING.



Syntax: How to Implement Automatic Gradient Riser Coloring

You can automatically create a gradient color effect between the risers in a graph. You implement this type of coloring by setting the color mode to 5, setting a starting color for the gradient, and setting an ending color for the gradient:

```
setColorMode(5);
setAutoColorGradientStartColor({Color.start_color|new Color(r1,g1,b1)});;
setAutoColorGradientEndColor({Color.end_color|new Color(r2,g2,b2)});;
```

where:

`setColorMode(5);`

Implements automatic gradient riser shading.

`start_color`

Identifies the starting color for the gradient.

`(r1,g1,b1)`

Is the code for the red, green, and blue components of the starting color.

`end_color`

Identifies the ending color for the gradient.

`(r2,g2,b2)`

Is the code for the red, green, and blue components of the ending color.

Example: Implementing Automatic Gradient Riser Coloring

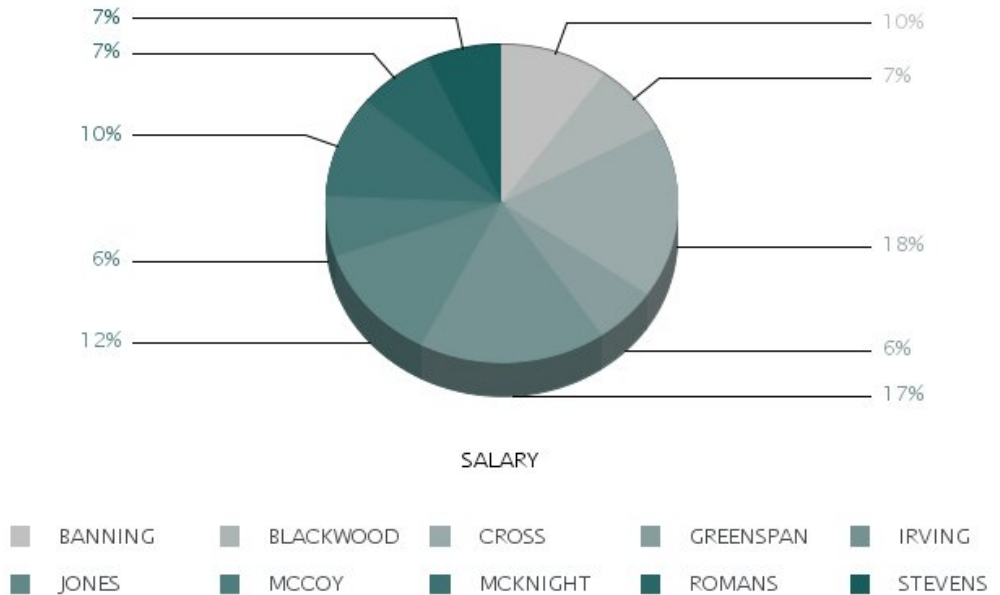
The following request implements automatic gradient riser coloring. It sets the color mode to 5, the start color to dark gray (192,192,192) and the end color to green (25,92,92):

```
-DEFAULTH &WF_STYLE_HEIGHT='405';
-DEFAULTH &WF_STYLE_WIDTH='770';
GRAPH FILE EMPLOYEE
-* Created by Info Assist for Graph
SUM EMPLOYEE.PAYINFO.SALARY
BY EMPLOYEE.EMPINFO.LAST_NAME
WHERE EMPLOYEE.EMPINFO.LAST_NAME NE 'SMITH';
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET GRAPHDEFAULT OFF
ON GRAPH SET VZERO OFF
ON GRAPH SET HAXIS &WF_STYLE_WIDTH
ON GRAPH SET VAXIS &WF_STYLE_HEIGHT
ON GRAPH SET UNITS PIXELS
ON GRAPH SET LOOKGRAPH PIEMULTI
ON GRAPH SET GRMERGE ADVANCED
ON GRAPH SET GRMULTIGRAPH 0
ON GRAPH SET GRLEGEND 0
ON GRAPH SET GRXAXIS 1
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setReportParsingErrors(false);
setSelectionEnableMove(false);
setDepthRadius(5);
setCurveFitEquationDisplay(false);
setDisplay(getDataText(0), false);
.
.
.
setDisplay(getDataText(63), false);
setPieDepth(70);
setPieFeelerTextDisplay(1);
setPlace(true);

-*Gradient
effect setColorMode(5);      setAutoColorGradientStartColor(new
Color(192,192,192));
    setAutoColorGradientEndColor(new Color(25,92,92)); -*

*END
INCLUDE=IBFS:/CFG/FILE/IBI_HTML_DIR/javaassist/intl/EN/ENIADefault_combine.sty,$
ENDSTYLE
END
```


On the output, the riser colors form a gradient from the start color (dark gray) to the end color (green):



Legends

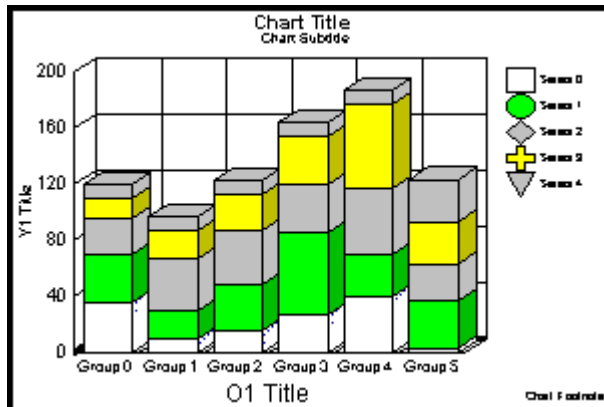
The following properties and methods control graph legends.

Legend Properties

These properties are used to control how the legend area, legend markers, and legend text are imaged in a graph:

- ☐ **LegendAutomatic** (true/false); Enables/disables auto-fitting of text inside the legend box. True is the default value.
- ☐ **LegendDisplay** (true/false); Enables/disables drawing of the legend box. True is the default value.
- ☐ **LegendMarkerPosition** (0...4); Sets the location and format of the graph legends. 0 is the default value.
- ☐ **LegendMarkersPerRow()**; Controls the number of markers per row shown in the legend area.
- ☐ **LegendOrient**(0...2); Controls orientation of markers and labels in the legend area.

- ❑ `LegendReverse(true/false)`; Controls the order of legend markers and labels.
- ❑ `LegendScroll(0...2)`; Controls the legend scroll bar.
- ❑ `LegendSeriesCount()`; When `LegendScroll` is enabled, this property defines the number of series at a time to appear in the scrolled legend area.
- ❑ `LegendSeriesStart()`; When `LegendScroll` is enabled, this property defines the first series to appear in the scrolled legend area.
- ❑ `LegendTextAutofit(true/false)`; Enables (true)/disables (false) auto-fitting of the legend text. True is the default value.
- ❑ `UseSeriesShapes(true/false)`; Enables/disables the use of the shapes defined by the `setMarkerShape()` method. This property is ignored if the `LegendMarkerPosition` property sets the legend text to display on top of the legend marker.



- ❑ `SquareMarkers(true/false)`; Can be used to force markers in the legend area to fit in a perfectly square rectangle regardless of their shape. This property is ignored if the `LegendMarkerPosition` property sets the legend text to display on top of the legend marker.

Note: All scroll, zoom, and pan operations require that the graph be rendered as an applet in the browser.

Legend Methods

You can use these methods to control how the legend area, legend markers, and legend text are imaged in a graph:

- ❑ `setLegendOrient(int newValue)`; Sets the orientation of the legends in a graph.

- ❑ `getLegendMinWidth();` Allows a user to resize the legend so that none of the labels wrap or truncate. Note that since it specifically calculates width, it is applicable to vertical legends only.
- ❑ `get/setLegendRect();` Gets/sets the size and location of the legend rectangle. Also see [Virtual Coordinate System](#) on page 47 for additional information about the virtual coordinate system.
- ❑ `setLegendTextAutofit(boolean newValue);` Enables (true)/disables (false) auto-fitting of the legend text.

Setting the Legend Position

You can select the position of the legend on your graph using `setLegendPosition`.

The code has been written so that additional presets can be added later if desired (for example place the legend inside the chart in the upper right corner). These positions are independent of Place Default Elements (PDE).

Place Default Elements should fix positioning issues. The user can define the legend rectangle by calling `setLegendAutomatic(false)` which instructs the legend engine to leave it alone. Similarly, calling `setLegendPosition(-1)` tells the legend engine to leave the legend's position alone.

Syntax: How to Set the Legend Position

`setLegendPosition (value);`

where:

value

Is one of the following:

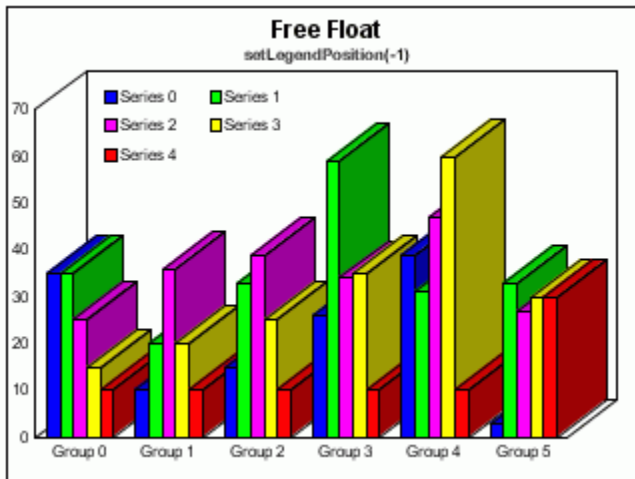
- ❑ `FREE FLOAT = -1;`
- ❑ `AUTO = 0;`
- ❑ `BOTTOM = 1;`
- ❑ `RIGHT = 2;`
- ❑ `LEFT = 3;`
- ❑ `TOP = 4;`
- ❑ `RIGHT TOP = 5;`
- ❑ `RIGHT BOTTOM = 6;`
- ❑ `LEFT TOP = 7;`

- ❑ LEFT BOTTOM = 8;
- ❑ BOTTOM LEFT = 9;
- ❑ BOTTOM RIGHT = 10;
- ❑ TOP LEFT = 11;
- ❑ TOP RIGHT = 12;

Example: Free Float Legend

```
restoreDefaults();
setDisplay(getFootnote(), false);
setDisplay(getY1MajorGrid(), false);
setDisplay(getO1MajorGrid(), false);
setGridStyle(getO1MinorGrid(), 0);
setLegendAutomatic(false);
setRect(getLegendArea(), new
Rectangle(-11318, 5359, 9066, 5438));
setLegendPosition(-1);
setDisplay(getFootnote(), false);
setSubtitleString("setLegendPosition(-1)");
setTitleString("Free Float");
setPlace(true);
```

The output is:



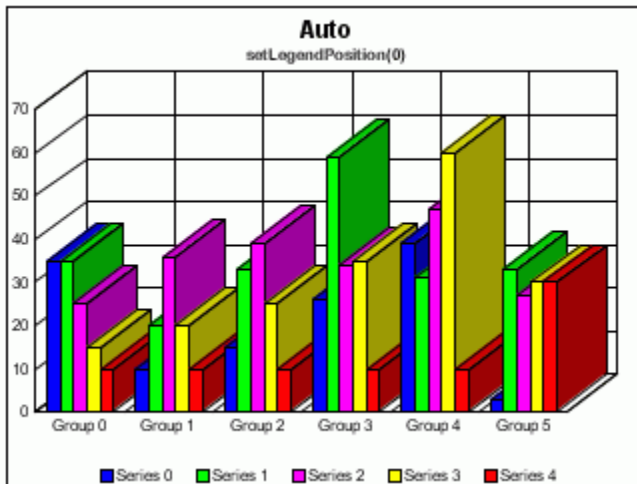
Example: Auto Legends

```

restoreDefaults();
setLegendPosition(0);
setDisplay(getFootnote(), false);
setSubtitleString("setLegendPosition(0)");
setTitleString("Auto");
setPlace(true);

```

The output is:

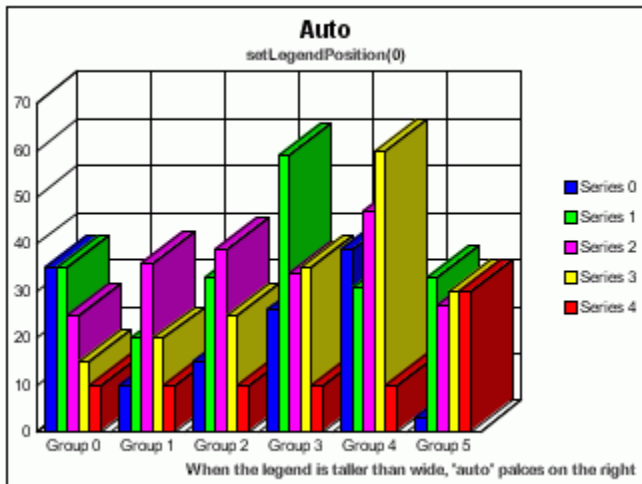


```

restoreDefaults();
setLegendPosition(0);
setRect(getLegendArea(), new
Rectangle(8237, -1290, 6637, 11349));
setTitleString("Auto");
setSubtitleString("setLegendPosition(0)");
setFootnoteString("When the legend is taller
    than wide, 'auto' places on the right");
setPlace(true);

```

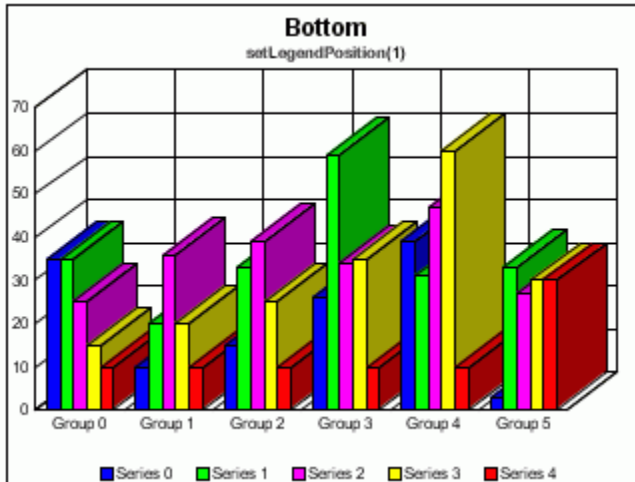
The output is:



Example: Bottom Legend

```
restoreDefaults();  
setLegendPosition(1);  
setDisplay(getFootnote(), false);  
setSubTitleString("setLegendPosition(1)");  
setTitleString("Bottom");  
setPlace(true);
```

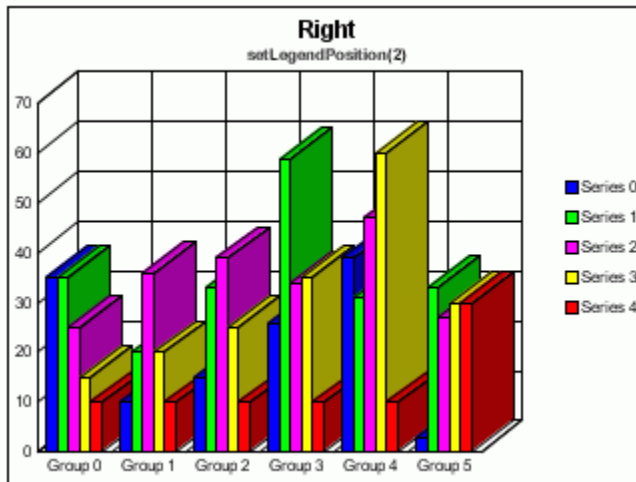
The output is:



Example: Right Legend

```
restoreDefaults();
setLegendPosition(2);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(2)");
setTitleString("Right");
setPlace(true);
```

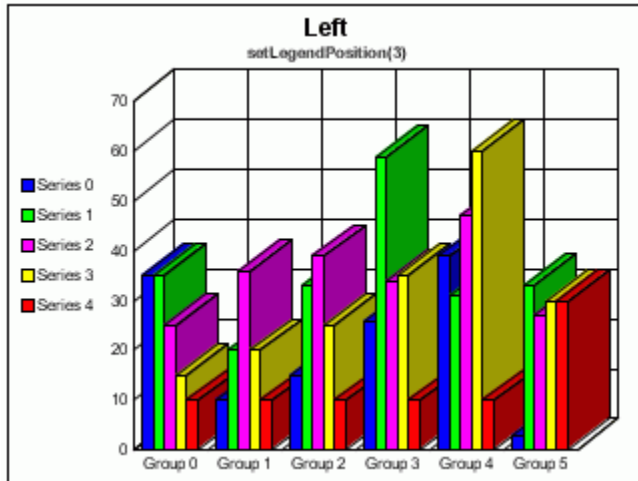
The output is:



Example: Left Legend

```
restoreDefaults();
setLegendPosition(3);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(3)");
setTitleString("Left");
setPlace(true);
```

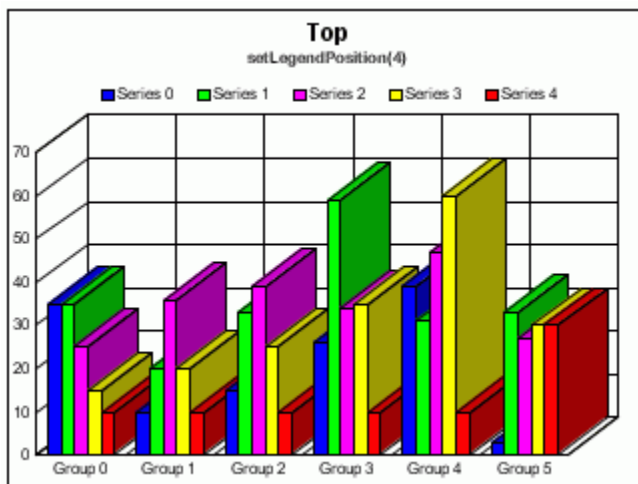

The output is:



Example: Top Legend

```
restoreDefaults();
setLegendPosition(4);
setDisplay(getFootnote(), false);
setSubtitleString("setLegendPosition(4)");
setTitleString("Top");
setPlace(true);
```

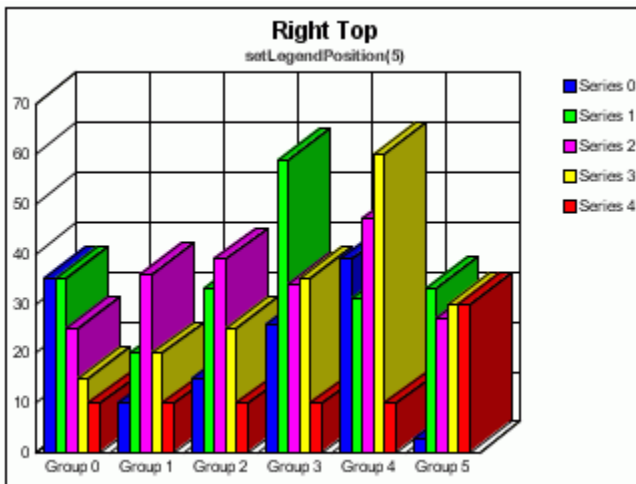
The output is:



Example: Right Top Legend

```
restoreDefaults();
setLegendPosition(5);
setDisplay(getFootnote(), false);
setSubtitleString("setLegendPosition(5)");
setTitleString("Right Top");
setPlace(true);
```

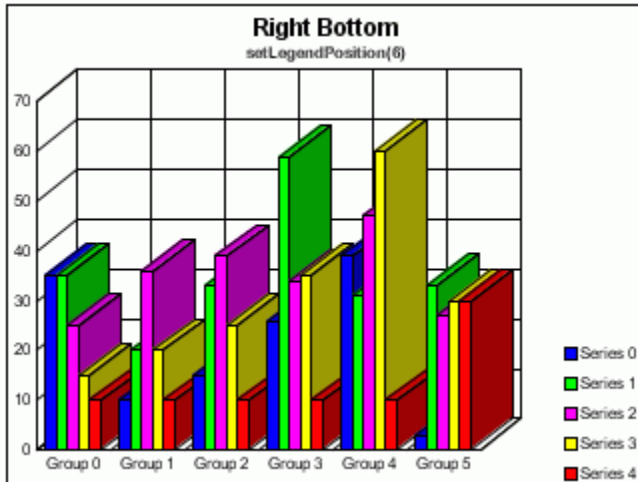
The output is:



Example: Right Bottom Legend

```
restoreDefaults();
setLegendPosition(6);
setDisplay(getFootnote(), false);
setSubtitleString("setLegendPosition(6)");
setTitleString("Right Bottom");
setPlace(true);
```

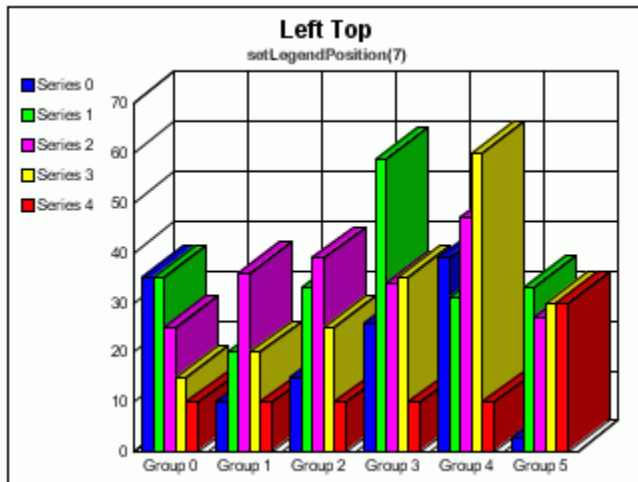
The output is:



Example: Left Top Legend

```
restoreDefaults();
setLegendPosition(7);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(7)");
setTitleString("Left Top");
setPlace(true);
```

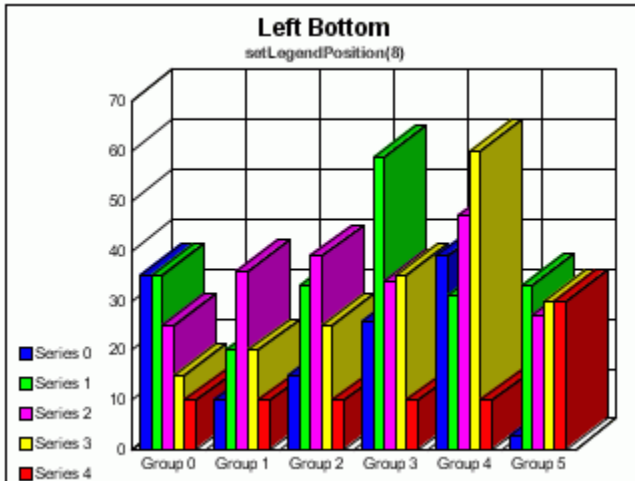
The output is:



Example: Left Bottom Legend

```
restoreDefaults();
setLegendPosition(8);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(8)");
setTitleString("Left Bottom");
setPlace(true);
```

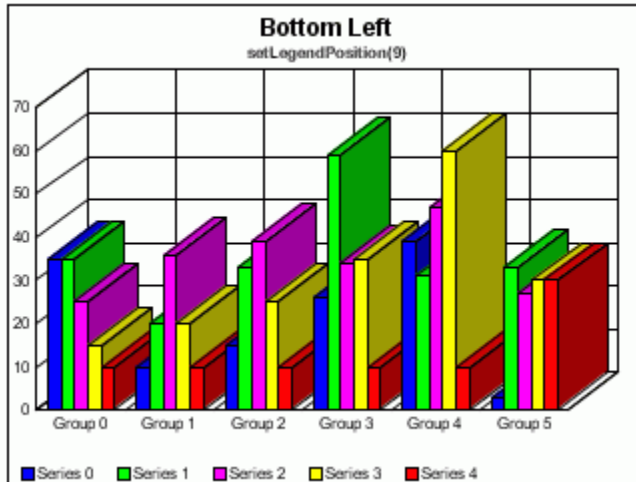
The output is:



Example: Bottom Left Legend

```
restoreDefaults();
setLegendPosition(9);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(9)");
setTitleString("Bottom Left");
setPlace(true);
```

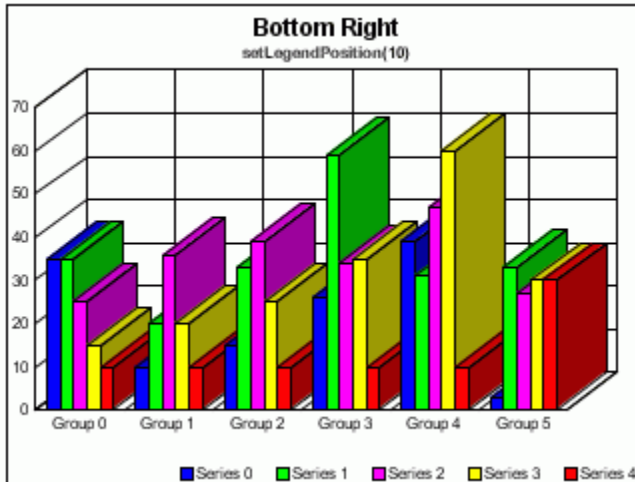
The output is:



Example: Bottom Right Legend

```
restoreDefaults();
setLegendPosition(10);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(10)");
setTitleString("Bottom Right");
setPlace(true);
```

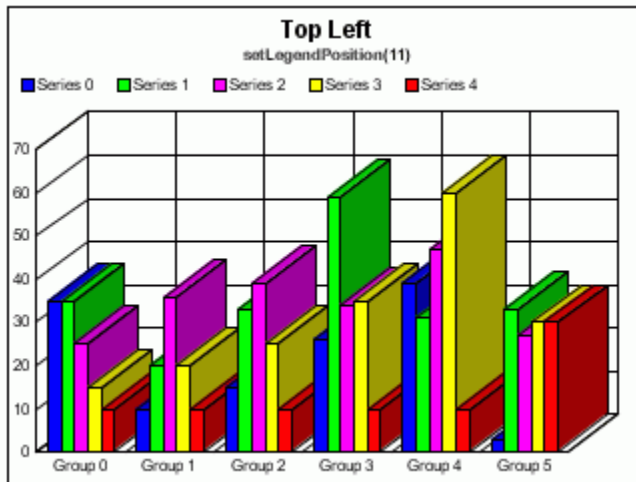
The output is:



Example: Top Left Legend

```
restoreDefaults();
setLegendPosition(11);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(11)");
setTitleString("Top Left");
setPlace(true);
```

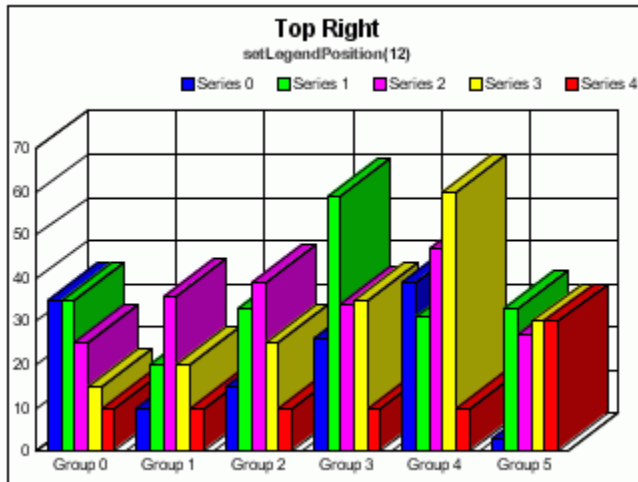
The output is:



Example: Top Right Legend

```
restoreDefaults();
setLegendPosition(12);
setDisplay(getFootnote(), false);
setSubTitleString("setLegendPosition(12)");
setTitleString("Top Right");
setPlace(true);
```


The output is:



Labels and Titles

The titles and labels that are included in the graph consist of the graph-wide titles (footnote, title, and subtitle) and ordinal and numeric axis labels and titles. All labels and titles can be defined, displayed, and auto-fitted using the properties and methods described in this section.

Graph-Wide Titles

You can use these properties to define, display, and auto-fit the graph title, subtitle, and footnote:

- ☐ FootnoteAutofit(true/false)
- ☐ FootnoteDisplay(true/false)
- ☐ FootnoteString (String)
- ☐ SubtitleAutofit(true/false)
- ☐ SubtitleDisplay(true/false)
- ☐ SubtitleString (string)
- ☐ TitleAutofit (true/false)
- ☐ TitleDisplay (true/false)
- ☐ TitleString (string)

Ordinal Axis Labels and Titles

You can use these properties to control the appearance of labels and titles on an ordinal axis:

- ❑ `O1/O2ExcludeMaxLabel(true/false)`
- ❑ `O1/O2ExcludeMinLabel(true/false)`
- ❑ `O1/O2LabelAutofit (true/false)`
- ❑ `O1LabelAutoskip (0...2)`
- ❑ `O1/O2LabelDisplay (true/false)`
- ❑ `O1/O2LabelRotate (0...2)`
- ❑ `O1LabelStagger(true/false)`
- ❑ `O1/O2LabelWrap (true/false)`
- ❑ `O1LabelSkipBegin (value)`
- ❑ `O1LabelSkipCount (value)`
- ❑ `O1/O2TitleAutofit (true/false)`
- ❑ `O1/O2TitleDisplay (true/false)`
- ❑ `O1/O2TitleString (string)`

You can use the `ExcludeMinLabel` and `ExcludeMaxLabel` properties to include/exclude the minimum and maximum labels on the ordinal axis.

The `LabelDisplay` and `LabelAutofit` properties determine whether labels are displayed and auto-fitted on the ordinal axis. See [Label and Title Methods](#) on page 119 for methods that you can use to define ordinal axis labels.

You can use the `LabelRotate`, `LabelStagger`, and `LabelWrap` properties to control how labels are imaged on the ordinal axis.

You can use the `LabelAutoSkip`, `LabelSkipBegin`, and `LabelSkipCount` properties to omit some of the labels on the O1-axis. When `LabelAutoSkip` selects manual skip mode, `LabelSkipBegin` and `LabelSkipCount` properties define the beginning label and interval of labels to be omitted. When `LabelAutoSkip` selects automatic label skip mode, labels are omitted as necessary when the graph is made smaller in order to maintain a reasonable, readable label size (a minimum 8-point font size).

You can use the `TitleAutofit`, `TitleDisplay`, and `TitleString` properties to define, display, and auto-fit the title on an ordinal axis.

Numeric Axis Labels and Titles

These properties control the appearance of labels and titles on a numeric axis:

- ☐ X1/Y1/Y2ExcludeMaxLabel(true/false)
- ☐ X1/Y1/Y2ExcludeMinLabel (true/false)
- ☐ X1/Y1/Y2LabelAutofit (true/false)
- ☐ X1/Y1/Y2LabelDisplay (true/false)
- ☐ X1/Y1/Y2LabelRotate (0...2)
- ☐ X1/Y1/Y2LabelStagger (true/false)
- ☐ X1/Y1/Y2TitleAutofit (true/false)
- ☐ X1/Y1/Y2TitleDisplay (true/false)
- ☐ X1/Y1/Y2TitleString (String)

You can use the ExcludeMinLabel and ExcludeMaxLabel properties to include/exclude the minimum and maximum labels on a numeric axis.

The LabelDisplay and LabelAutofit properties determine whether labels are displayed and auto-fitted on a numeric axis.

The LabelRotate and LabelStagger properties control how labels are imaged on a numeric axis.

The TitleAutofit, TitleDisplay, and TitleString properties define, display, and auto-fit the title on a numeric axis.

Label and Title Methods

These methods control the appearance of labels and titles:

- ☐ get/setDataLabel(); Get/sets a label string at a specified series and group.
- ☐ get/setAutoSkip(); Gets/sets automatic, manual, or no skipping of labels on an axis.
- ☐ get/setExcludeMaxLabel(); Gets/sets whether or not maximum labels are imaged for an axis in a graph.
- ☐ get/setExcludeMinLabel(); Gets/sets whether or not minimum labels are imaged for an axis in a graph.
- ☐ get/setGroupLabel(); Gets/sets the label string assigned to a group object.
- ☐ get/setLabelStagger(); Gets/sets the label stagger attribute for an axis object.

- ❑ `get/setSeriesLabel()`; Gets/sets the label string assigned to a series object.
- ❑ `get/setSkipBegin()`; When you use `setAutoSkip()` to select manual skipping of labels on an axis, use these properties to `get/set` the first label to be skipped.
- ❑ `get/setSkipCount()`; When you use `setAutoSkip()` to select manual skipping of labels on an axis, use these properties to `get/set` the interval of labels to be skipped.

When you use these methods, you may supply an object ID as an input parameter to assign or retrieve the attribute to or from a specific label. If you do not specify a label object, the methods assign the attribute to or retrieve the attribute from the first item in the selection list. If the object ID does not identify a label object or the first item in the selection list is not a label object, a set operation applies the attribute to the object but it will not have any effect on the appearance of the graph.

Formatting Date and Time Values for Y-Axis Labels

Military time uses a 24-hour time format where numbers between zero (0) and one (1), or times using a 12-hour format with AM and PM indicators, are converted to times between 0:00 and 23:00. Numbers greater than one are cycled so that they map to one of these hours as well, except for cumulative military time, in which case they add to the original value.

The following table illustrates the mapping of hours between formats, using Excel general format as the initial value:

General Format	Military Cycle	AM/PM Cycle	Military Cumulative
0	0:00	12:00AM	0:00
0.25	6:00	6:00 AM	6:00
0.5	12:00	12:00PM	12:00
0.75	18:00	6:00PM	18:00
1.00	0:00	12:00AM	24:00
1.25	6:00	6:00 AM	30:00
1.75	18:00	6:00PM	42:00
2	0:00	12:00AM	48:00

Two methods support these time formats for y-axis labels:

- ❑ `setTextFormatPreset` must be set to 32 for the object to be formatted, such as Y1 Labels. The data for the Y1 axis must be in Java Date Format, which is based on `java.text.DateFormat`. This is defined as the milliseconds since January 1, 1970, 00:00:00 GMT
- ❑ `setSimpleDateFormat(string)` specifies the date and time options to display, where *string* can be a combination of the following format codes:

Format Character	Description	Example
G	Era designator	AD
Y	Four-digit year	2014
M	Month in year	July or 07
d	Day in month	10
D	Day in year	360
E	Day in week	Tuesday
F	Day of week in month	2 (second Wed. in July)
h	Hour in A.M./P.M. format (1~12)	12
H	Hour in military cycle format (0~23)	22
K	Hour in A.M./P.M. (0~11)	10
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	234
w	Week in year	40
W	Week in month	1

Format Character	Description	Example
a	A.M./PM. marker	PM
z	Time zone	Eastern Standard Time
'	Escape for text	Delimiter
"	Single quotation mark	`

The default format for the SimpleDateFormat string is: HH:MM

For example:

```
setTextFormatPreset(getY1Label(), 32);
setSimpleDateFormat("E yyyy.MM.dd 'at' hh:mm:ss a zzz");
```

A sample of output for these API calls is:

```
Sun 2004.07.18 at 04:14:09 PM PDT
```

Example: Specifying Date and Time Formats for Y-Axis Labels

The following request generates varying time values to use as the measure on the chart and displays y-axis labels in the format "E yyyy.MM.dd 'at' hh:mm:ss a zzz":

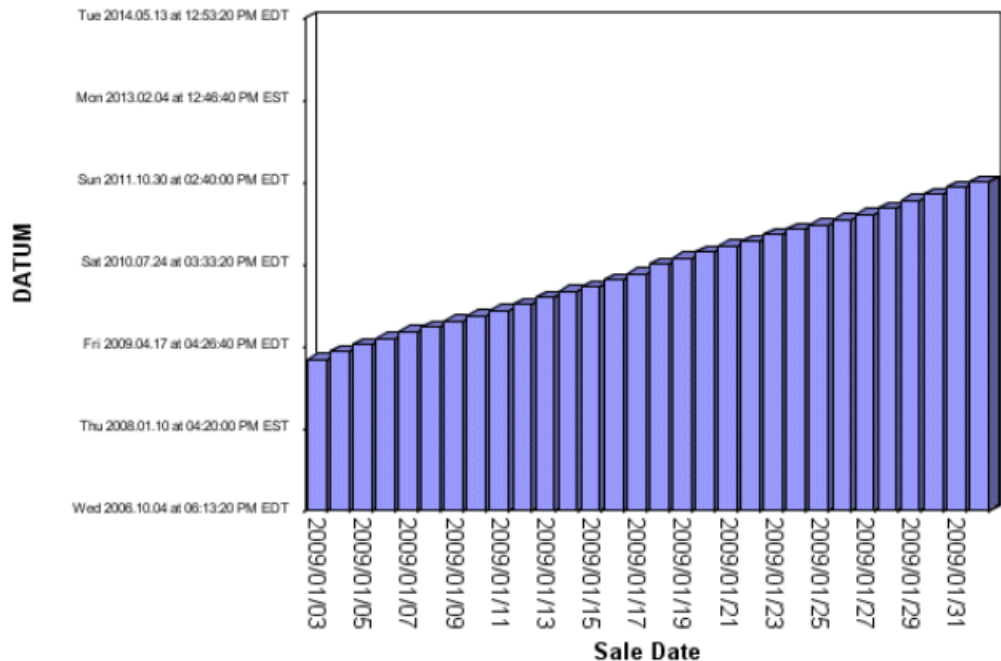
```

DEFINE FILE WF_RETAIL_LITE
  ZZ/I11 WITH ID_SALES = ZZ + 1;
  KUM/I11 WITH ID_SALES = ZZ * 5;
  DATUM/YYMD WITH ID_SALES = DATEADD(TIME_DATE, 'D', ZZ);
END

GRAPH FILE WF_RETAIL_LITE
  SUM DATUM
  BY TIME_DATE
  ON GRAPH PCHOLD FORMAT PNG
  ON GRAPH SET LOOKGRAPH VBAR
  ON GRAPH SET STYLE *
  *GRAPH_SCRIPT
    setPlace(true);
    setFontSizeAbsolute(getY1Label(),true);
    setFontSizeInPoints(getY1Label(),6);
    setPlaceResize(getY1Label(),0);
    setTextFormatPreset(getY1Label(), 32);
    setSimpleDateFormat("E yyyy.MM.dd 'at' hh:mm:ss a zzz");
  *END
  ENDSTYLE
END

```

The output is:



Using Nested/Multi-Dimensional Labels

Nested/multi-dimensional labels are supported on the O1 axis with these properties and methods:

- ❑ `setO1LabelCallback()`; This method assigns a callback function to the nested labels interface.
- ❑ `getO1LabelCallback()`; This method determines if a callback function has been assigned with `setO1LabelCallback()`.
- ❑ `NestedLabels (true/false)`; This property enables/disables nested labels.

Use the following procedures to implement the nested labels interface:

1. Create your own version of the nested labels callback. It must implement all of the abstract methods defined in TDGNestedLabel.Java. They are:

```
abstract int getNumLevels();
abstract int getNumLabelsOnLevel(int nLevel);
abstract Vector getAllLabels(int nLevel);
abstract String getLabel(int nGroup, int nLevel);
abstract int getLabelGrouping(int nGroup, int nLevel);
```

2. Register the callback with the setO1LabelCallback() method. For example:

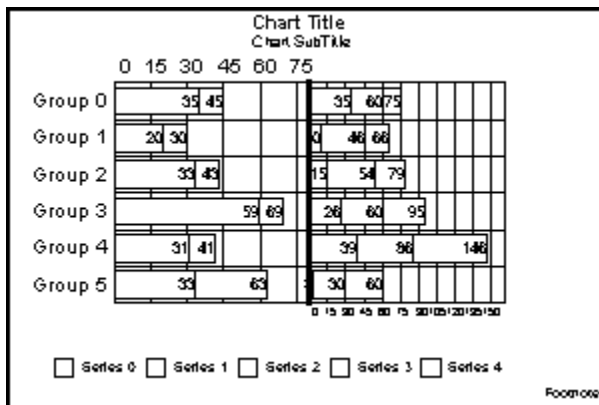
```
m_chart.setO1LabelCallback (TDGNestedLabel cb);
```

3. Enable the NestedLabel property. For example:

```
m_chart.setNestedLabels (true);
```

Data Text

In addition to labels and titles, you may also enable/disable the display of data values (data text) in a graph.



Use the following properties and methods to display data text and define where and how the data text is displayed:

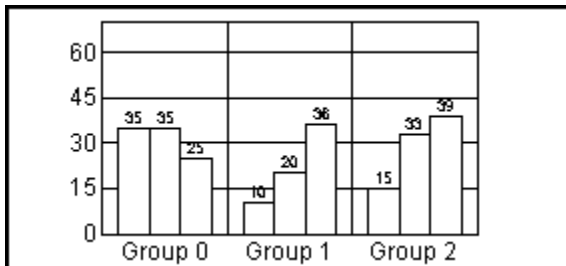
- ☐ **DataltemsAlongSeries** (true/false); This property determines whether data items are aligned parallel to the series (both rows or both columns).
- ☐ **DataTextAngleDefault** (0...360); This property sets an angle from center-point that all data text is drawn from. 90 is the default value.
- ☐ **DataTextDisplay** (true/false); This property enables/disables the display of data values next to risers or markers in a graph.

- ❑ `DataTextPosition` (0...5); This property can be used to define the position where data text is displayed. 0 is the default value.
- ❑ `DataTextRadiusDefault` (0...100); This property sets the default radius that data text will be drawn out from the center of a riser.
- ❑ `get/setDataTextAngle()`; These methods `get/set` the angle from the center point that all or selected data text is drawn.
- ❑ `get/setDataTextRadius()`; These methods `get/set` the radius for the data text position of a specified data point.

The `DataTextPosition` property determines where data text labels are displayed in the graph. If this property is set to zero, the values set by `DataTextAngleDefault` or `setDataTextAngle` and `DataTextRadiusDefault` or `setDataTextRadius` will be used to determine where data text is displayed. The following examples show the data text positions that can be selected with `DataTextPosition` (1...5).

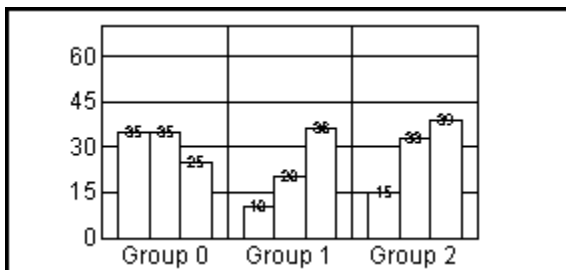
Example:

```
setDataTextPosition(1);
```



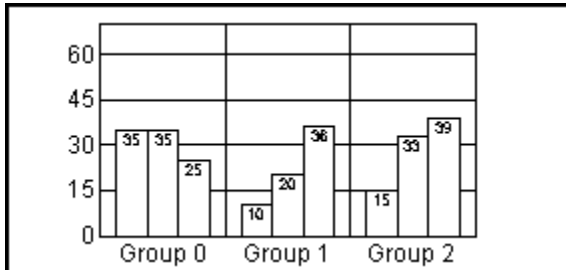
Example:

```
setDataTextPosition(2);
```

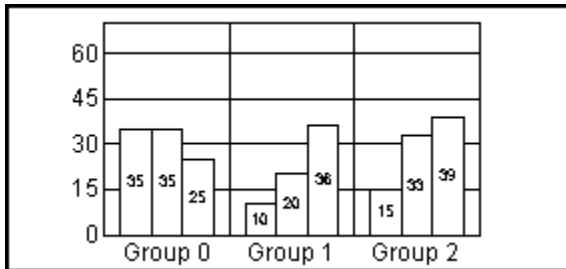


Example:

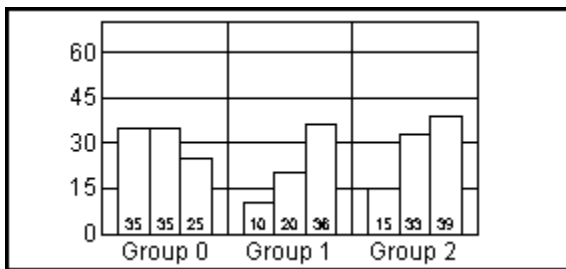
```
setDataTextPosition(3);
```

**Example:**

```
setDataTextPosition(4);
```

**Example:**

```
setDataTextPosition(5);
```



Data text is typically drawn for all series and groups.

Customizing Data Text

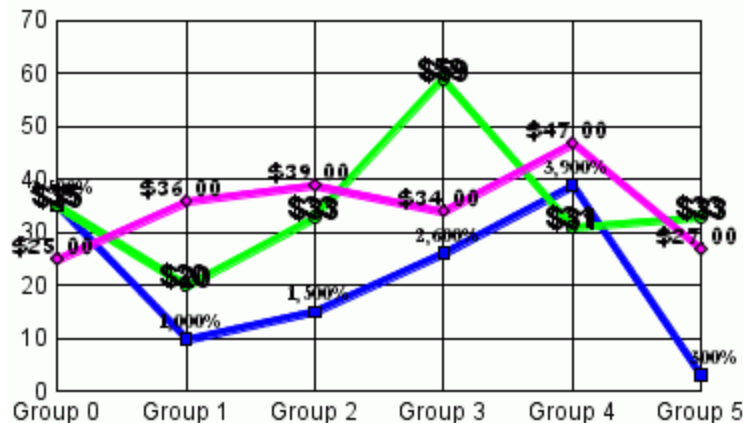
Data text is normally drawn for all series and groups using the same numeric format (for example, general, \$, %, and so on). You can use `setDisplay()` to show/hide data text for individual series.

```
setDisplay(getDataText(0), false); // Hide data text for series 0
setDisplay(getDataText(1), true);  // Show data text for series 1
```

When `setCustomDataText()` enables customized data text for a particular series, you can change the numeric format of the individual series of data text as well as specify font characteristics.

Example:

```
setDepthAngle (0);
setDepthRadius (0);
setGraphType (41);
setCustomDataText (getSeries (2), true);
setCustomDataText (getSeries (1), true);
setCustomDataText (getSeries (0), true);
setTextFormatPreset(getDataText (2), 5);
setTextFormatPreset(getDataText (1), 6);
setTextFormatPreset(getDataText (0), 2);
setDataTextDisplay (true);
setViewableSeries (3);
setFontName(getDataText(2), "Comic Sans MS");
setFontName(getDataText(0), "Garamond");
setFontSizeAbsolute(getDataText(1), true);
setFontSize(getDataText(1), 14);
setTextJustHoriz(getDataText(1),1);
setShadowDisplay(getDataText(1), true);
```



get/setCustomDataText()

When data text is displayed with setDataTextDisplay(true), these methods get/set custom formatting of an individual series of data text. When custom data text is enabled for a series, use the setTextFormatPreset() and/or setTextFormatPattern() methods to define the format of data text for individual series.

Syntax:

```
boolean getCustomDataText( IdentObj idSeries );  
void setCustomDataText( IdentObj idSeries, boolean bNewValue );
```

where:

idSeries

Is the object ID returned by a getObjectID() method (for example, getSeries()).

bNewValue

Is one of the following values:

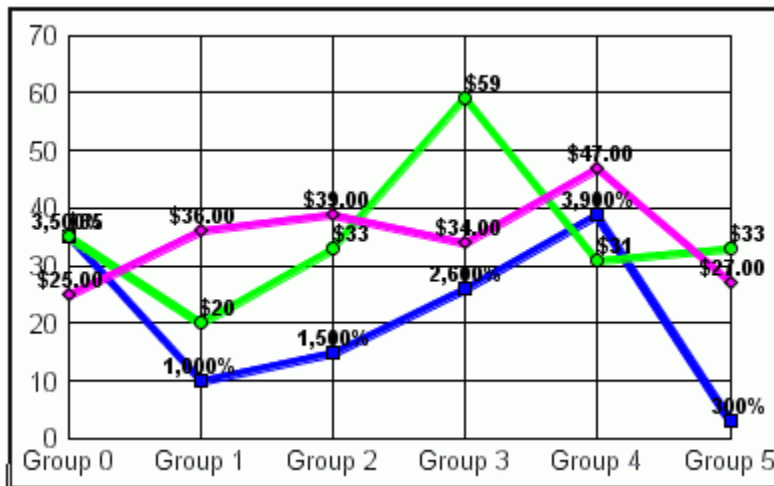
- ☐ true = enable custom data text for the specified series.
- ☐ false = disable custom data text for the series.

Return:

boolean; true = custom data text is enabled for the specified series, false = custom data text is not enabled

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setCustomDataText(getSeries(2), true);
setCustomDataText(getSeries(1), true);
setCustomDataText(getSeries(0), true);
setTextFormatPreset(getDataText(2), 5);
setTextFormatPreset(getDataText(1), 6);
setTextFormatPreset(getDataText(0), 2);
setDataTextDisplay(true);
setViewableSeries(3);
```



Formatting Numbers and Numeric Labels

The following methods are supported for defining the format of data text and numeric chart labels:

- ☐ Select a preset pattern using a ...Text/LabelFormat property.
- ☐ Define a standard Java format pattern using one of the ...Text/LabelFormatPattern properties.

Preset Number Formats

You can use the following ...Format properties to select one of the preset patterns or to tell the graphing engine to use a format pattern:

- ☐ DataTextFormat()
- ☐ PieFeelerTextFormat()

- ❑ PieRingTotalFormat()
- ❑ X1LabelFormat()
- ❑ Y1LabelFormat()
- ❑ Y2LabelFormat()

The following methods can be used to set the format of labels on a specified axis and to specify custom data text formatting:

- ❑ setTextFormatPreset(); assign a preset format to an object
- ❑ setCustomDataTextFormat(); assign a preset format to a series

When you use these properties and methods, -2 or -1 tells the graphing engine that the format will be defined by one of the ...FormatPattern properties or methods. 0...21, 24...30 selects one of the formats shown in the following table.

Value	Format	Example
-2	#/100 Note: The number format is the custom format specified by setTextFormatPattern.	123 = 1.23
-1	# Note: The number format is the custom format specified by setTextFormatPattern.	123 = 123
0/1=	#	123 = 123
2=	#%	123 = 12,300%
3=	#.##%	123 = 12,300.0%
4=	###%	123 = 12,300.00%
5=	\$#.##	123 = \$123.00
6=	\$#	123 = \$123
7=	#K (show K for values over 999)	1,000 = 1K

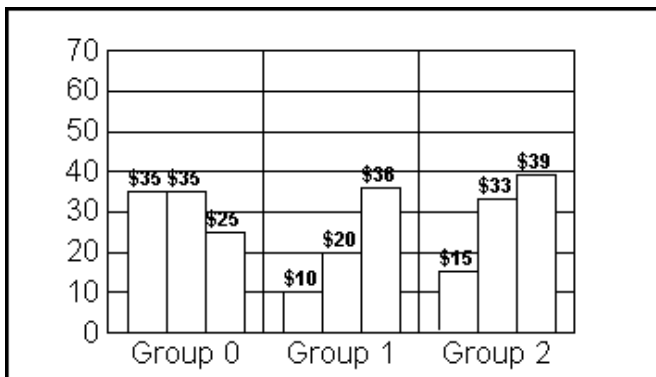
Value	Format	Example
8=	\$#K (show K for values over 999)	1,000 = \$1K
9=	#M (show M for millions)	1,000,000 = 1M
10=	\$#M (show M for millions)	1,000,000 = \$1M
11=	#B (show B for billions)	1,000,000,000 = 1B
12=	\$#B (show B for billions)	1,000,000,000 = \$1B
13=	#T (show T for trillions)	1,000,000,000,000 = 1T
14=	\$#T (show T for trillions)	1,000,000,000,000 = \$1T
15=	Number with thousands separators, no decimal places.	1,234 = 1K
16=	Number with thousands separators, two decimal places.	1,234 = 1.23K
17=	General currency format for current locale.	
18=	Short date format: MM/DD/YY.	10/01/03
19=	Medium date format: Mon. DD, YYYY.	Oct. 01, 2003
20=	Long date format: Month DD, YYYY.	October 01, 2003
21=	Full date format: Day, Month DD, YYYY.	Wednesday, October 01, 2003
24/25	# Negative numbers are in brackets	-123= (123)
26	Number with thousands separators, no decimal places, negative numbers in brackets.	-1,234= (1K)
27	Number with thousands separators, two decimal places, negative numbers in brackets.	-1,234 = (1.23K)
28	##%/100	123 = 123%

Value	Format	Example
29	#.##%/100	123 = 123.0%
30	#.###%/100	123 = 123.00%

Example:

The following example code selects the dollar format for data text labels:

```
setDataTextFormat(6);
```

**Number Format Patterns**

You can use the following ...FormatPattern properties to specify a standard Java format pattern:

- ☐ DataTextFormatPattern (PatternString)
- ☐ PieFeelerTextFormatPattern (PatternString)
- ☐ PieRingTotalFormatPattern (PatternString)
- ☐ X1LabelFormatPattern (PatternString)
- ☐ Y1LabelFormatPattern (PatternString)
- ☐ Y2LabelFormatPattern (PatternString)

The following methods can be used to define format pattern strings:

- ☐ setTextFormatPattern(); assign a format pattern to an object.
- ☐ setCustomDataTextFormatPattern(); assign a custom data text format pattern to a series.

When you use these properties, the `PatternString` defines the format of data text or labels in the following format:

```
pattern:= subpattern{;subpattern}
subpattern:= {prefix}integer{.fraction}{suffix}
prefix:= '\\u0000'..'\\uFFFFD' - specialCharacters
suffix:= '\\u0000'..'\\uFFFFD' - specialCharacters
integer:= '#'* '0'* '0'
fraction:= '0'* '#'*
```

Notation:

`X*`

Means there are 0 or more instances of X.

`(X|Y)`

Is either X or Y.

`X..Y`

Is any character from X up to Y, inclusive.

`S - T`

Are characters in S, except those in T.

The first subpattern is for positive numbers. The second (optional) subpattern is for negative numbers. In both cases, a comma (,) can occur inside the integer portion. Here are the special characters used in the parts of the subpattern, with notes on their usage.

`0`

Is a digit.

`#`

Is a digit, zero shows as absent.

`.`

Is a placeholder for decimal separator.

`,`

Is a placeholder for grouping separator.

`;`

Separates formats.

`-`

Is the default negative prefix.

%

Divides by 100 and show as percentage.

x

Determines that any other characters can be used in the prefix or suffix.

,

Is used to quote special characters in a prefix or suffix.

If there is no explicit negative subpattern, a minus sign (-) is prefixed to the positive form. That is, 0.00 alone is equivalent to 0.00;-0.00. Illegal formats, such as "#.#.#" or mixing '_' and '*' in the same format, will cause a ParseException. From that ParseException, you can find the place in the string where the error occurred. The grouping separator is commonly used for thousands, but in some countries for ten-thousands. The interval is a constant number of digits between the grouping characters, such as 100,000,000 or 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "#####,####" == "##,####,####".

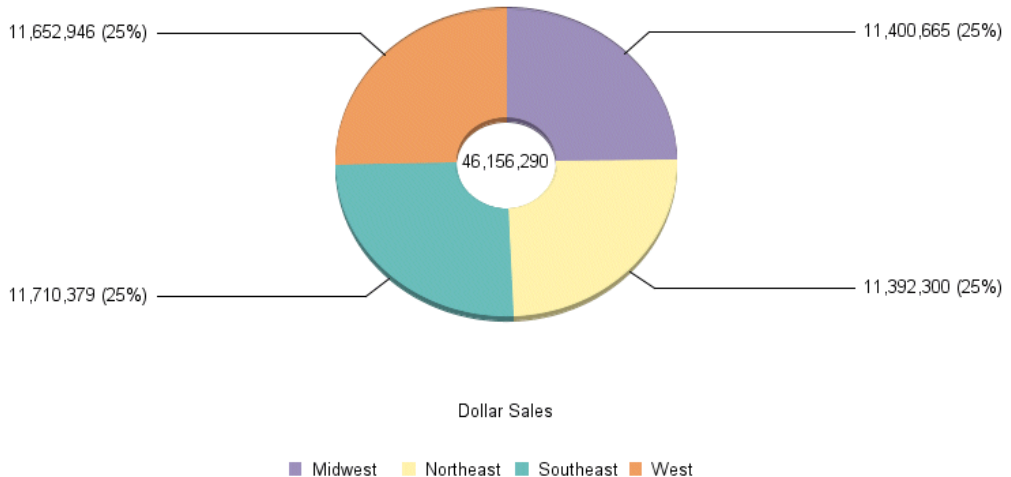
This class only handles localized digits where the 10 digits are contiguous in Unicode, from 0 to 9.

Example: Formatting a Pie Ring Label

The following commands use the `setTextFormatPreset()` and `setTextFormatPattern()` methods to set the pie ring label format to set the numeric format to `###,###`:

```
setTextFormatPreset(getPieRingLabel(), -1);
setTextFormatPattern(getPieRingLabel(), "###, ###");
```

The output shows that the pie ring label (the number at the center of the chart) is formatted with a comma as the group separator and no decimal digits:



Special Formatting Using Macros

You can use the following properties in bubble, stock, and scatter graphs to apply special formatting to data text values:

- ❑ `DataSetTemplateBubble()`; Customizes the data text labels and inserts macros for the actual data any place in a defined string when data text labels are enabled in a bubble graph.
- ❑ `DataSetTemplateHiLo()`; Customizes the data text labels and inserts macros for the actual data any place in a defined string when data text labels are enabled in a stock graph.
- ❑ `DataSetTemplateScatter()`; Customizes the data text labels and inserts macros for the actual data any place in a defined string when data text labels are enabled in a scatter graph.

Formatting Text Objects

Use the following properties and methods to format and display text objects in the graph:

- ❑ `FontSizeAbsolute`: This method enables/disables the use of absolute font sizing (that is, the font size is not changed regardless of the virtual coordinates system). If this property is enabled (true), use `setFontSize()` to specify an absolute font size that will not be changed regardless of the size of the window where it is drawn. If this property is disabled (false), use `setFontSizeVC()` to specify a font size that will be scaled when the window size changes.

- ❑ `get/setFontName();` These methods `get/set` the font name of a text object.
- ❑ `setFontSize();` Sets an absolute font size in points.
- ❑ `get/setFontSizeVC();` These methods `get/set` the font size in virtual coordinates of a text object. See [Virtual Coordinate System](#) on page 47 for additional information about the virtual coordinates system.
- ❑ `get/setFontStyle();` These methods `get/set` the font style (for example, bold, italic, and so on) of a text object.
- ❑ `get/setTextJustHoriz();` These methods `get/set` the horizontal justification attribute of a text object in a graph.
- ❑ `get/setTextJustVert();` These methods determine the vertical justification attribute of a text object in a graph.
- ❑ `get/setTextRotation();` These methods `get/set` the rotation attribute of a text object in a graph.
- ❑ `get/setTextString();` These methods `get/set` the string of characters assigned to a text object in a graph.
- ❑ `get/setTextWrap();` These methods `get/set` the wrapping attributes of a text object in a graph.

Java 2D Text Objects

All Perspective for Java chart text objects (for example, title, subtitle, labels, and so on) are Java 2D Text objects. You can use the following methods to apply color, textures, and gradients to these objects and to specify strike through and underline types:

- ❑ `get/setShadowFillType();` Get/Set shadow fill type (color, gradient, or texture) for a text object.
- ❑ `get/setShadowGradientEndColor();` For a gradient shadow fill type, `get/set` the gradient end color.
- ❑ `get/setShadowGradientStartColor();` For a gradient shadow fill type, `get/set` the gradient start color.
- ❑ `get/setShadowTextureURL();` For a texture shadow fill type, `get/set` the URL where the texture file is defined.
- ❑ `get/setTextBackgroundColor();` For color shadow fill type, set the background color of a text object.
- ❑ `get/setTextStrikethrough();` Get/set strike through mode for a text object.

- ❑ `get/setUseTextBackgroundColor()`; Enable/disable use of the text background color set by `setTextBackgroundColor()`.

Autofitting

You can use these properties and methods to control automatic fitting of text objects in a graph:

- ❑ `AxisTextAutofitMax ( value )`; This property defines the maximum size of axis text in virtual coordinates.
- ❑ `AxisTextAutofitMin ( value )`; This property defines the minimum size of axis text in virtual coordinates.
- ❑ `AxisTextAutofitMode ( 0...2 )`; This property defines the mode to be used for autofitting axis text.
- ❑ `AxisTextAutofitPercent ( 0...100% )`; When `AxisTextAutofitMode` is set to 2, this property defines the percent to be used for autofitting axis text. All axis text is maintained within `AxisTextAutofitPercent` of the axis with the smallest font size as determined by `Autofit`.
- ❑ `FootnoteAutofit (true/false)`; This property enables (true)/disables (false) automatic fitting of the graph footnote text string. True is the default value.
- ❑ `get/setAutofit()`; These methods get/set the auto-fitting attribute of an object in a graph.
- ❑ `O1/O2/X1/Y1/Y2LabelAutofit (true/false)`; These properties automatically fit/size all text labels on an axis.
- ❑ `O1/O2/X1/Y1/Y2TitleAutofit (true/false)`; These properties automatically fit/size the title on an axis.
- ❑ `SubtitleAutofit (true/false)`; This property enables (true)/disables (false) auto-fitting of the graph subtitle string.
- ❑ `TextAutofitMax/TextAutofitMin(Integer)`; These properties specify the maximum and minimum font size (in virtual coordinates) that can be used by the graphing engine to automatically fit and size label and title objects within their bounding box.
- ❑ `TitleAutofit (true/false)`; This property enables (true)/disables (false) auto-fitting of the graph title string.
- ❑ `LegendTextAutofit (true/false)`; This property enables (true)/disables (false) auto-fitting of the legend text. True is the default value.

Graph-Specific Properties and Methods

The topics in this section describe properties and methods specific to 3D cubes, pie graphs, stock graphs, bubble graphs, funnel graphs, multi-Y axis graphs, Pareto graphs, and waterfall graphs.

3D Cube Characteristics

When you select a 3D graph type with the `GraphType` property, use these properties to control the appearance of the 3D cube:

- ❑ `CubeFocusFactor (0...100)`; This property sets the focus factor for setting perspective distortion in a 3D graph.
- ❑ `CubeIsometricProjection (true/false)`; This property enables (true)/disables (false) isometric projection.
- ❑ `CubeLightSourceX/Y/Z (0...100)`; These properties set the cube light source for the X-, Y-, and Z-coordinates (unit space).
- ❑ `CubeProportional(true/false)`; Define the axis size proportional to the number of series (false) or groups (true).
- ❑ `CubeRiserInterpolation()`; This property controls whether risers in 3D graphs are interpolated (faster) or explicitly calculated (more accurate).
- ❑ `CubeSizeX/Y/Z (0...100)`; These properties set the size of the 3D cube in the (user) X-, Y-, and Z-directions.
- ❑ `CubeTranslationX/Y/Z (0...100)`; These properties set the translation of a 3D cube (in 3D cube coordinates) in the X-, Y-, and Z-direction.
- ❑ `CubeViewerX/Y/Z (0...100)`; These properties set the viewer location (in 3D coordinates) in the X-, Y-, and Z-direction.
- ❑ `CubeWallThickX/Y/Z (0...100)`; These properties set the thickness of the 3D-cube wall in the (user) X-, Y-, and Z-direction.
- ❑ `CubeZoomFactor (0...100)`; This property sets the global scaling factor for zooming in/out.
- ❑ `Display3DFloor (true/false)`; This property enables/disables the display of the floor of the cube in a 3D graph.
- ❑ `Display3DLeftWall (true/false)`; This property enables/disables the display of the left wall of the 3D cube.

- ❑ **Display3DRightWall** (true/false); This property enables/disables the display of the right wall of the 3D cube.
- ❑ **Viewing3DAnglePreset** (-1...15); This property selects an entry in the table of preset viewing angles for 3D graphs.

-1=	Custom (as set by CubeViewerX/Y/Z, CubeWallThickX/Y/Z, and so on)
0=	Standard (the default)
1=	Tall and Skinny
2=	From the Top
3=	Distorted
4=	Short and Fat
5=	Groups Eye
6=	Group Emphasis
7=	Few Series
8=	Few Groups
9=	Distorted Standard
10=	Shorter and Fatter
11=	Thick Wall for Series
12=	Thick Wall Standard
13=	California Special
14=	Blast-O-Vision
15=	Special

[Graph Types and 3D Preset Viewing Angles](#) on page 1213 includes an illustration of each of these preset viewing angles. Also see [3D Grid Lines](#) on page 89 and [Riser and Marker Properties](#) on page 92 for information about properties that you can use to control the appearance of grid lines and risers in the 3D cube.

Pie Graph Properties and Methods

When you select a pie graph is selected with the GraphType property, you can use these properties and methods to control attributes that are specific to a pie graph:

- ❑ OtherPercentage(0...100%); This property sets the percentage at which a slice in a pie graph will be grouped into the other slice when the OtherSeries property is enabled.
- ❑ OtherSeries(true/false); This property enables/disables the bundling of small slices (less than OtherPercentage) into a single slice that is labeled other.
- ❑ PieDepth (0...100); This property specifies the depth of the pie crust in a pie graph or a funnel graph. 30 is the default value.
- ❑ PieFeelerTextDisplay (0...3); This property enables (true)/disables (false) the display of feelers and data text in a pie graph.
- ❑ PieFeelerTextFormat (0...14); This property determines the format of values that are drawn next to feelers in a pie graph.
- ❑ PieLabelDisplay (0...3); This property determines the format of labels displayed next to feelers in a pie graph. 1 is the default value.
- ❑ PieRingSize (0...100); This property determines the size of the ring (inner circle) in a ring pie. 30 is the default value.
- ❑ PieRingTotalDisplay (true/false); This property enables (true)/disables (false) the display of a total value in the center of a pie ring graph.
- ❑ PieRingTotalFormat: When the PieRingTotalDisplay property is set to true and the total data value is displayed in the center of a ring pie graph, this property sets the format of the data value.
- ❑ PieRotate (0...359); This property rotates a pie or funnel graph a specified number of degrees. 0 is the default value.
- ❑ PieSorting(0...3); This property defines the order in which pie slices are drawn.
- ❑ PiesPerRow (integer); This property specifies the number of pies to be drawn in a row in multi-pie graphs. 2 is the default value.

- ❑ `PieTilt (0...100)`; This property tilts a pie or funnel graph a specified number of degrees. 15 is the default value.
- ❑ `PieBarSeries (0...# of Series in graph)`; This property selects the series that will be used to map the bar in a pie-bar graph. 0 is the default value.
- ❑ `get/setPieSliceDelete()`; These methods get/set a boolean value indicating whether or not slices have been deleted from a pie graph.
- ❑ `get/setPieSliceDetach()`; These methods get/set the distance that slices are detached from a pie graph.
- ❑ `restoreAllSlices()`; This method restores all detached/deleted slices from a pie graph.
- ❑ `isChartPieType()`; This method returns a boolean indicating whether or not the `GraphType` property is set to one of the pie graphs.

Stock Graph Properties and Methods

When you select a stock graph with the `GraphType` property, use these properties and methods to control attributes that are specific to a stock graph:

- ❑ `DataTextTemplateHiLo`; This property assigns a special format macro to data text labels when they are enabled for display in a stock graph.
- ❑ `Stock52WeekHighValue()`; This property gets and sets the 52-week high value assigned to a stock graph.
- ❑ `Stock52WeekLowValue()` This property gets and sets the 52-week low value assigned to a stock graph.
- ❑ `Stock52WeekHighDisplay (true/false)`; This property enables (true)/disables (false) the display of a 52-week high line in a stock graph.
- ❑ `Stock52WeekLowDisplay (true/false)`; This property enables (true)/disables (false) the display of a 52-week low line in a stock graph.
- ❑ `StockCloseSplitDisplay (true/false)`; This property enables (true)/disables (false) the display of split risers at the stock close value.
- ❑ `StockCloseTicksDisplay (true/false)`; This property enables (true)/disables (false) the tick marks at the stock close value.
- ❑ `StockMovingAverageDisplay (true/false)`; This property enables (true)/disables (false) the display of a moving average line in a stock graph.

- ❑ `StockOpenTicksDisplay` (true/false); This property enables (true)/disables (false) the tick marks at the stock open values.
- ❑ `StockTickLength` (Integer); This property determines the length of tick marks in a stock graph.
- ❑ `ViewableGroupsStock`: This property specifies the maximum number of groups to be imaged in a stock graph.
- ❑ `ViewableSeriesStock`: This property specifies the maximum number of series to be imaged in a stock graph.
- ❑ `isChartStockType()`; This method returns a boolean indicating whether or not the `GraphType` property is set to one of the stock graphs.

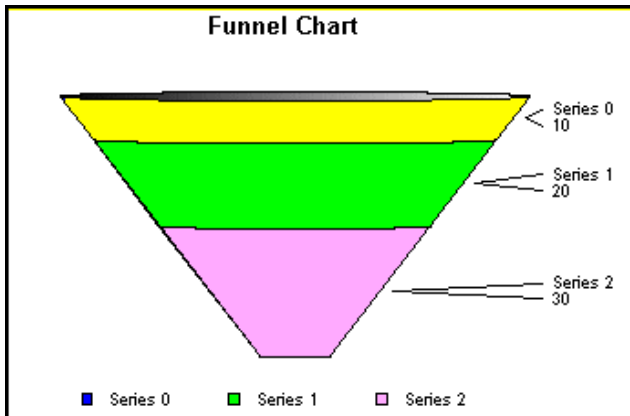
Bubble Graph Properties and Methods

The following properties and methods format special items that are only available in bubble graphs:

- ❑ `LabelTextTemplateBubble`: This property assigns a special format macro to data text labels when they are enabled for display in a bubble graph.
- ❑ `QuadrantLineCountX`: This property selects a number of quadrant lines on the X-axis in a bubble graph.
- ❑ `QuadrantLineCountY`: This property selects a number of quadrant lines on the Y-axis in a bubble graph.
- ❑ `getQuadrantLine()`; This method gets the object ID of the quadrant lines in a bubble graph.
- ❑ `getQuadrantLineValueX()`; This method gets the vertical location of an X quadrant line.
- ❑ `getQuadrantLineValueY()`; This method gets the horizontal location of a Y quadrant line.
- ❑ `setQuadrantLineValueX()`; This method sets the vertical location of an X quadrant line.
- ❑ `setQuadrantLineValueY()`; This method sets the horizontal location of a Y quadrant line.

Funnel Graph Properties and Methods

A funnel graph is basically a pie graph 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.



Example:

Use the following methods to define the initial setup of a funnel graph:

```
setTransparentBorderColor(getFrame(), true);
setDisplayFunnelGroup(int nGroup); // Group to show
setGraphType(106);
```

You must set which group of data you wish to display.

The funnel graph contains graph specific objects:

- ☐ Feelers.
- ☐ Funnel value labels.
- ☐ Funnel labels (frequently referred to as the series labels).

The following methods return the object ID of these funnel graph objects:

```
getFunnelFeeler()
getFunnelValueLabel()
getFunnelLabel()
```

The following property defines which group to display in the funnel graph. 0 is the default value.

```
setDisplayFunnelGroup(int nGroup)
```

The following property defines the location of labels in a funnel graph.

```
setFunnelLabelValueLocation(int nLocation)
```

where:

nLocation

Can be set to a value in the range 0...2. 0 is the default value.

0=	Labels on top of values drawn on the right of the funnel.
1=	Labels on the left of the funnel/Values on the right of the funnel.
2=	Values on the left of the funnel/Labels on the right of the funnel.

The following pie graph properties can be used in funnel graphs:

- ❑ `setPieDepth(0...100)`; Sets the depth of the funnel graph. 30 is the default value.
- ❑ `setPieRotate(0...259)`; Sets the length of the funnel tip. 50 is the default value.
- ❑ `setPieTilt(0...100)`; Sets the amount of tilt toward the viewer.

For every funnel graph, it is recommended that you set the transparent border color of the graph frame to true. This overrides the default behavior (normally false for other graph types) of showing a black rectangle around the frame and will create a better looking funnel graph.

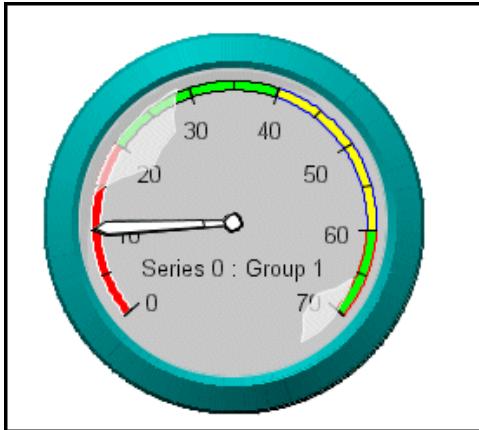
```
setTransparentBorderColor(getFrame(), true)
```

To change the color of the funnel feeler, use `setBorderColor()`. For example:

```
setBorderColor(getFunnelFeeler(), Color.red)
```

Gauge Graph Properties & Methods

When the GraphType property selects a gauge graph (setGraphType(110)), the following properties can be used to modify the appearance of the graph:



- ❑ GaugeBackgroundStyle; Defines the style of the gauge background.
- ❑ GaugeDescending; Draw gauge values in ascending or descending order.
- ❑ GaugeIgnoreBackgroundStyle; Ignore/obey the GaugeBackgroundStyle property.
- ❑ GaugeMajorAutoStep; Enable/disable automatic calculation of the number of major grid/tick steps. If this property is disabled (FALSE), Y1MajorGridStep defines the number of major grid steps.
- ❑ GaugeMajorTickRelativeTickLength; Get/set the relative length of major ticks.
- ❑ GaugeMinorAutoStep; Enable/disable automatic calculation of the number of minor grid/tick steps. If this property is disabled (FALSE), Y1MinorGridStep defines the number of minor grid steps.
- ❑ GaugeMinorTickRelativeTickLength; Get/set the relative length of minor ticks.
- ❑ GaugeNeedleStyle; Get/set the gauge needle style.
- ❑ GaugeRelativeInnerRadius; Get/set the relative size of the inner radius.
- ❑ GaugeRelativeThickness; Get/set gauge's relative thickness.
- ❑ GaugeStartAngle; Get/set gauge start angle.
- ❑ GaugeStopAngle; Get/set gauge stop angle.

- ❑ `GaugeTitlePosition`; Set the title positioning. See [How to Set the Title Position in a Gauge Chart](#) on page 147.

The `setFontSize...()` methods can be used to modify the size of gauge labels. For example:

```
IdentObj identObj = getGaugeLabel();
setFontSizeAbsolute( identObj, true );
setFontSize( identObj, 9 );
```

The `setTextFormatPreset()` method can be used to change the format of gauge labels:

```
IdentObj identObj = getGaugeLabel();
setTextFormatPreset( identObj , 5 /*$0*/ );
```

The `setURL()` method also works with gauges. A URL can be attached to a gauge (multi gauge graph) as shown in the following example:

```
IdentObj identObj = getGaugeBackground();
// group 1
identObj.setGroupID(1);
setURL( identObj,
"http://www.google.com" );
```

- ❑ The `setDisplay()` method displays the value of the gauge on its background. See [How to Control the Display of the Value in the Gauge Chart](#) on page 148.
- ❑ The `setY1ExcludeMinLabel()` method displays the minimum value of the gauge on its background. [How to Control the Display of the Minimum Value on the Gauge Scale](#) on page 149.
- ❑ The `setY1ExcludeMaxLabel()` method displays the maximum value of the gauge on its background. See [How to Control the Display of the Maximum Value on the Gauge Scale](#) on page 150.

Also see the Gauge Graph objects.

Syntax: **How to Set the Title Position in a Gauge Chart**

```
setGaugeTitlePosition(int);
```

where:

int

Is a value from 1-3:

1 places the title below the chart.

2 places the title in the center (this is the default).

3 places the title above the chart.

Syntax: **How to Control the Display of the Value in the Gauge Chart**

```
setDisplay(getGaugeValue(), newValue);
```

where:

`newValue`

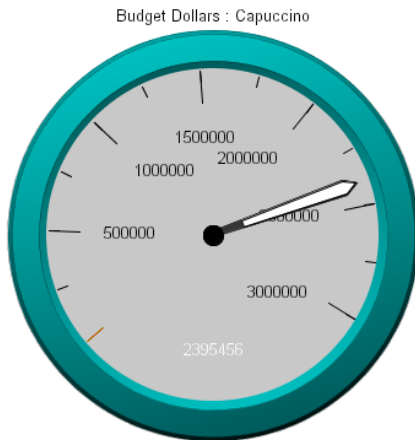
Is either true or false.

When `newValue` is true, the gauge value displays on the gauge background; when it is false, the value is not displayed. The default is true.

Example: **Controlling the Display of the Value in the Gauge Chart**

```
GRAPH FILE GGSales
SUM GGSales.SALES01.BUDDOLLARS
ACROSS GGSales.SALES01.PRODUCT
WHERE (GGSales.SALES01.PRODUCT EQ 'Capuccino');
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setTextFormatPreset(getY1Label(),1);
setDisplay(getGaugeValue(), true);
setPlace(true);
*END
ENDSTYLE
END
```


The graph displays as the following:



Syntax: **How to Control the Display of the Minimum Value on the Gauge Scale**

```
setY1ExcludeMinLabel(newValue);
```

where:

`newValue`

Is either true or false.

When `newValue` is true, the minimum label is not displayed on the gauge scale; when it is false, the minimum value displays. The default is false.

Example: **Controlling the Display of the Minimum Value on the Gauge Scale**

```
GRAPH FILE CAR
SUM HEIGHT
ACROSS CAR.ORIGIN.COUNTRY
WHERE (CAR.ORIGIN.COUNTRY EQ 'FRANCE');
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setY1ExcludeMinLabel(true);
setPlace(true);
*END
ENDSTYLE
END
```

The graph displays as the following:



Syntax: **How to Control the Display of the Maximum Value on the Gauge Scale**

```
setY1ExcludeMaxLabel(newValue);
```

where:

`newValue`

Is either true or false.

When `newValue` is true, the maximum label is not displayed on the gauge scale; when it is false, the maximum value is displayed. The default is false.

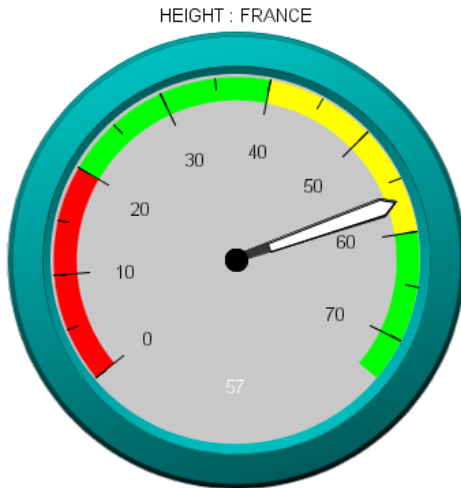
Example: Control the Display of the Maximum Value on the Gauge Scale

```

GRAPH FILE CAR
SUM HEIGHT
ACROSS CAR.ORIGIN.COUNTRY
WHERE (CAR.ORIGIN.COUNTRY EQ 'FRANCE');
ON GRAPH SET LOOKGRAPH GAUGE1
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setY1ExcludeMaxLabel(true);
setPlace(true);
*END
ENDSTYLE
END

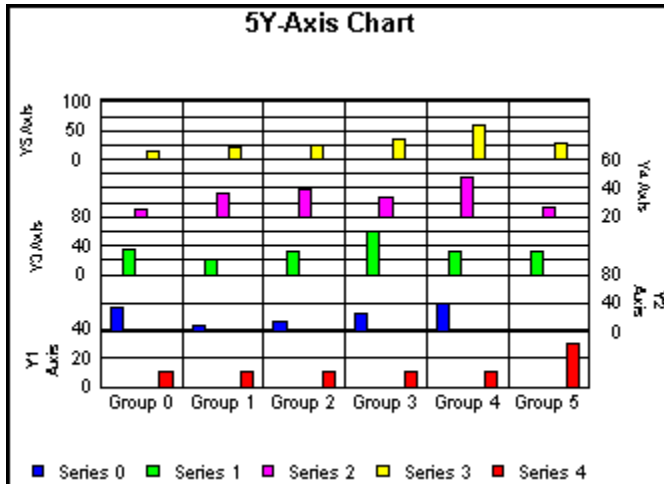
```

The graph displays as the following:



Multi-Y Axes Graph Properties and Methods

In addition to dual-axes graphs, 3Y-, 4Y-, and 5Y-axes graphs (GraphType 103...105) are also available.



Example:

Series must be assigned to the axes using `setAxisAssignment()`.

```
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setAxisAssignment(getSeries(4), 5);
setGraphType(105)
```

By default, the series risers assigned to each axis are staggered. The `AlignMultiYBars` property can be used to align the bars on each axis.

By default, axis labels are staggered with odd numbered (1/3/5) axis labels on the left and even numbered (2/4) axis labels on the right. These default settings can be changed with `setAxisSide()`.

Example:

```
setAxisSide(getY3Axis(),1); //Y3 Axis on Right
setAxisSide(getY4Axis(),0); //Y4 Axis on Left
```

The size of each axis can be changed using the following method:

```
setAxisSize(IdentObj id, int newValue)
```

where:

newValue

Is 0...100. 1 (automatic) is the default value.

Example:

```
setAxisSize(getY1Axis(), 20);
```

If the AxisSize is set to 0 (automatic) for all axes, automatic sizing of axes and offsets will occur. This is the default setting for each axis. In other words, axes and offsets will be sized proportionally dividing the available space of the frame evenly. If an even distribution between each axis is desired, there is no need to set the axis size or the axis offset.

Syntax: How to Set the Axis Offset From the Base of the Frame

The axis offset from the base of the frame can also be set using the following method:

```
setAxisOffset(IdentObj id, int newValue);
```

where:

newValue

Is 0...100. 0 is the default value.

Example:

```
setAxisOffset(getY1Axis(), 20);
```

With this feature, axes are allowed to overlap. In most cases this may not be a desirable effect. If the axis sizes are set to non-zero values, appropriate relationships between the offsets and sizes must be set.

```
setAxisOffset(getY1Axis(), 0);
setAxisOffset(getY2Axis(), 20);
setAxisOffset(getY3Axis(), 40);
setAxisOffset(getY4Axis(), 50);
setAxisOffset(getY5Axis(), 80);
setAxisSize(getY1Axis(), 20);
setAxisSize(getY2Axis(), 20);
setAxisSize(getY3Axis(), 10);
setAxisSize(getY4Axis(), 30);
setAxisSize(getY5Axis(), 20);
```

The percentages of available frame space in this example are:

- ❑ 20% for the Y1 axis
- ❑ 20% for the Y2 axis

- ❑ 10% for the Y3 axis
- ❑ 30% for the Y4 axis
- ❑ 20% for the Y5 axis

The following methods return the object ID of the Y3-, Y4-, and Y5-axis titles:

```
getY3Title()  
getY4Title()  
getY5Title()
```

Example:

The following displays the Y3-axis title:

```
setTextString(getY3Title(), "Y3 Title");  
setDisplay(getY3Title(), true);
```

Pareto Graph Properties and Methods

When you select a Pareto graph (GraphType 102), the X-axis scale is for group numbers. The Y-axis scale is the percent of the total accumulation for each selected series. Each graph displays a single series. Use the DisplayParetoSeries property to define the single series to be shown in the Pareto graph.

Example:

```
setDisplayParetoSeries (1);  
setGraphType(102);
```

The default Pareto series is zero.

The ParetoDisplayThreshold property can be used to split the frame and accumulation line per the desired percent. (1...100). 1 (no split) is the default.

Example:

```
SetParetoY2MajorGridStep(5);
```

The ConnectParetoBars property can be used to connect/disconnect the bars in a Pareto graph.

Waterfall Graph Properties and Methods

When you select a waterfall graph (GraphType 100 (vertical) or 101 (horizontal)), groups within the graph are identified as subtotals, totals, or extra. The waterfall graph automatically performs the cumulative sum when using subtotal or total. Groups labeled as extra are ignored in waterfall calculations.

Note: Values in a given group in a stacked waterfall must be all negative or all positive. Zero is treated as a positive number. If the data contains mixed signs, the sign of the first series controls whether the entire stack will be treated as all positive or all negative.

See [Data in a Waterfall Graph](#) on page 208 for more information about setting up the data required for a waterfall graph and restrictions.

The following methods can be used to select a vertical or horizontal waterfall graph:

```
setGraphType(100); // WATERFALL_VERT = 100;
setGraphType(101); // WATERFALL_HORZ = 101;
```

The `setWaterfallGroupMode()` method assigns each group in the data set to be graphed as a normal, a subtotal, total, or extra group.

```
void setWaterfallGroupMode(int nGroup, int nGroupMode);
```

where:

nGroup

Is a group number.

nGroupMode

Can be one of the following:

```
int WATERFALLGROUPMODE_NORMAL = 0;
int WATERFALLGROUPMODE_SUBTOTAL = 1;
int WATERFALLGROUPMODE_TOTAL = 2;
int WATERFALLGROUPMODE_EXTRA = 3;
```

By default the group mode is set to normal. If a waterfall group is set to subtotal or total, the data for this group is ignored and the cumulative total up to this point is shown for this group. Only one group can be designated as total or extra. Total is computed the same way as a subtotal except there is only one of them. For the best visual presentation, the group designated as extra should be the last group in the graph. A group designated as extra is not used in performing any waterfall cumulative sums. It appears as stacked data.

Colors and Shading

These properties and methods control the color and shading of objects in a graph:

- ❑ Autoshading (true/false); This property enables (true)/disables (false) automatic shading of 3D cube walls and riser faces. When depth effect is applied to a 2D graph with `DepthRadius`, this property also applies shading to the faces of the 2.5D risers.

- ❑ `ColorMode (0...2)`; This property sets the color mode to be used in the graph (automatic, color-by-series, or color-by-group). For 3D surface graphs, automatic mode will cause the graph to be colored by height. For bar, line, and area graphs, automatic mode will color the graph by series if there is more than one series or by groups if there is only one series. 0 (automatic) is the default value.
- ❑ `ExactColorByHeight (true/false)`; This property enables (true)/disables (false) exact coloring by height in spectral maps and 3D surface graphs. When this property is enabled (true), the actual data value is used to get a color from the color-by-height gradient. When this property is disabled (false), the colors on the spectral legend are the only ones used and the data value is compared to determine which of those buckets it falls into.
- ❑ `SeriesLooping (1...# of series in graph)`; Use this property to select the interval at which series color values are repeated. The default setting (32) defines different colors for the first 32 series (0...31). Starting with series 32, the colors are repeated (that is, series 32...64 have the same colors as 0...31, respectively).
- ❑ `get/setBorderColor()`; These methods get/set the border color of an object.
- ❑ `get/setFillColor()`; These methods get/set the fill color of an object.
- ❑ `get/setFillType()`; These methods get/set the fill type of an object in a graph. The fill type can be a color, gradient, or texture.
- ❑ `getSelectionBorderColor()`; This method returns the border color of an object.
- ❑ `getSelectionFillColor()`; This method returns the fill color of an object.
- ❑ `get/setTransparentBorderColor()`; These methods get/set the transparent border color of an object.
- ❑ `get/setTransparentFillColor()`; These methods get/set the transparent fill color of an object.
- ❑ `setSeriesBorderColor()`; This method sets the border color of risers for a specified series in a graph.
- ❑ `setSeriesFillColor()`; This method sets the fill color of risers for a specified series in a graph.

- ❑ `getColorByHeight()`; When a graph is colored by height (rather than by series or by group), the graphing engine chooses the color of quantitative data representations based on the data values supplied to the graph. This happens in spectral map graphs and in 3D surface graphs. If, for example, the data ranges from 7 to 74 and the Y1 axis scale, therefore, goes from 0 to 80, the graphing engine might have the color blue assigned to 0, red assigned to 80, and various interpolated shades for values between 0 and 80. Note that spectral map graphs do include a Y1 axis even though it is not visible in the graph. The `getColorByHeight()` method returns the `IdentObj` of the color by height gradient (in this case a blue to red one). With the object ID, your application can modify the gradient. Its end colors can be changed or the gradient could be made more complex.

When you select a data point, you can change its color as described above, except that if `ExactColorByHeight` is false, the graphing engine changes the color of the appropriate spectral legend marker's value instead of the color for the exact data value.

Example:

```
setExactColorByHeight(false);
```

When you use data ranges from 23 to 57, the gradient goes from blue (20) to red (60). There are spectral markers at 20, 25, 30, ...60. When you click on marker 20 and change it to black, the colors are interpolated from black (20) to red (60). When you click on data point 31 and change its color to green, a new pin with the color green is added at 0.25 (31 is colored as if it were 30 and 30 is 25% of the way from 20 to 60). Now the colors are interpolated as follows: black (20) to green (30); green (30) to red (60).

```
setExactColorByHeight(true);
```

When the user clicks on data point 48 and changes its color to yellow, a new pin is added at 0.7 (48 is 70% between 20 and 60) with the color yellow. Now the colors are interpolated as follows: black (20) to green (30); green (30) to yellow (48); yellow (48) to red (60).

You can use the following properties and methods to control the display of shadows in a graph.

- ❑ `get/setShadowColor()`; These methods `get/set` the color of a shadow applied to an object.
- ❑ `get/setShadowDisplay()`; These methods `get/set` whether or not a drop shadow is displayed for a particular object in a graph.
- ❑ `get/setShadowXOffset()`; These methods `get/set` the offset in the X-direction of a drop shadow for a particular object in a graph.
- ❑ `get/setShadowYOffset()`; These methods `get/set` the offset in the Y-direction of a drop shadow for a particular object in a graph.

- ❑ `ShadowXOffsetDefault (0...100)`; This property sets the default value for the X-offset of a drop shadow. 30 is the default value.
- ❑ `ShadowYOffsetDefault (0...100)`; This property sets the default value for the Y-offset of a drop shadow. 30 is the default value.

Defining Colors

You can set the color of most graph objects. A color can be defined as `rgb` values or hex values that specify the intensities of the red, green, and blue components of the color. For some objects, such as rectangular series risers and chart backgrounds, you can also define a transparency value for the fill color.

Syntax: How to Define a Color for a Graph Object

```
setobjectColor(new Color(newValue));
```

where:

setobjectColor

Is the API call to set the color for the specified object.

newValue

Is the default color value in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

r, *g*, *b*, *t* is the desired intensity of red, green, and blue, respectively, and a transparency value. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. For transparency, 0 is fully transparent and 255 is fully opaque.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#). The hexadecimal color value 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).

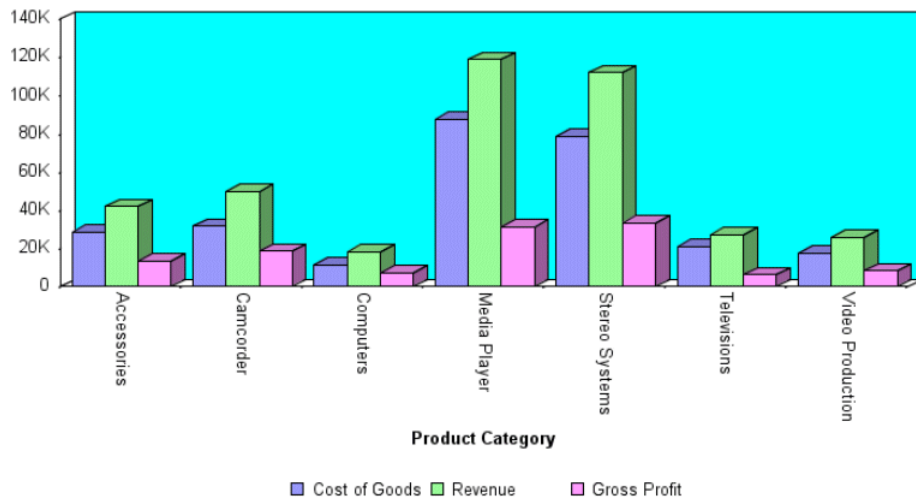
For example, black is #000000, which corresponds to `rgb(0,0,0)` . Red is #FF0000, which corresponds to `rgb(255,0,0)`. White is #FFFFFF, which corresponds to `rgb(255,255,255)`.

Example: Setting a Fill Color for the Chart Frame

The following request sets the chart frame fill color to aqua (rgb 0,255,255):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
  setFillColor(getFrame(),new Color(0,255,255));
setPlace(true);
*END
  ENDSTYLE
  END
```

The output is:



To express the color as a hex value, the equivalent API call is:

```
setFillColor(getFrame(),new Color(#00FFFF));
```

Reference: Usage Notes for Color Transparency

Color transparency support is limited and only applies to transparent rectangular regions (so, for example, it is not supported for pie slices). In addition:

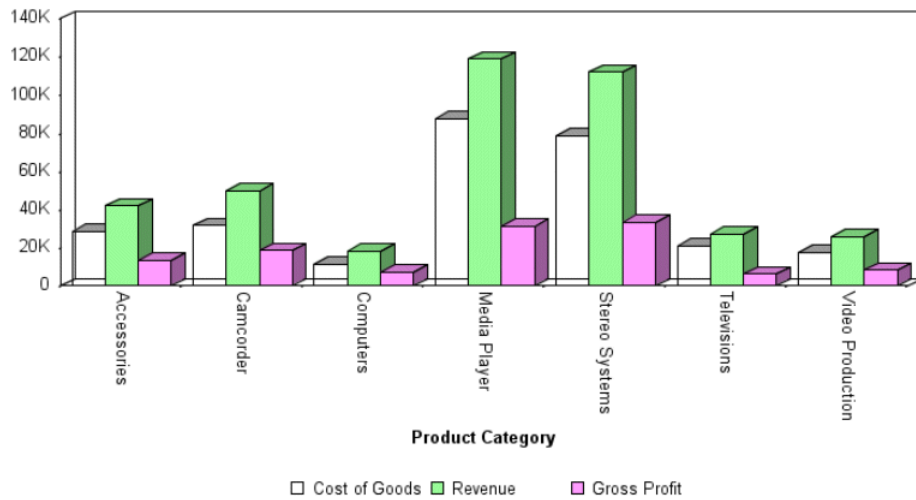
- ❑ Transparent background fill colors are supported only when the background color is transparent. The border color of the background should also be transparent, if you are going to use a transparent fill color for the background.
- ❑ Transparency is only supported for PNG or GIF chart formats.
- ❑ Transparent colors are supported in 2D charts, but not in 2.5D or 3D charts. In some cases, they may be usable in 2.5D charts but are not officially supported.
- ❑ Transparent colors are useful for alternate format regions, to have grid lines show up more clearly.

Example: Setting a Color Transparency for a Series Fill Color

The following request sets the series 0 (cost of goods) fill color to fully transparent (transparency 0):

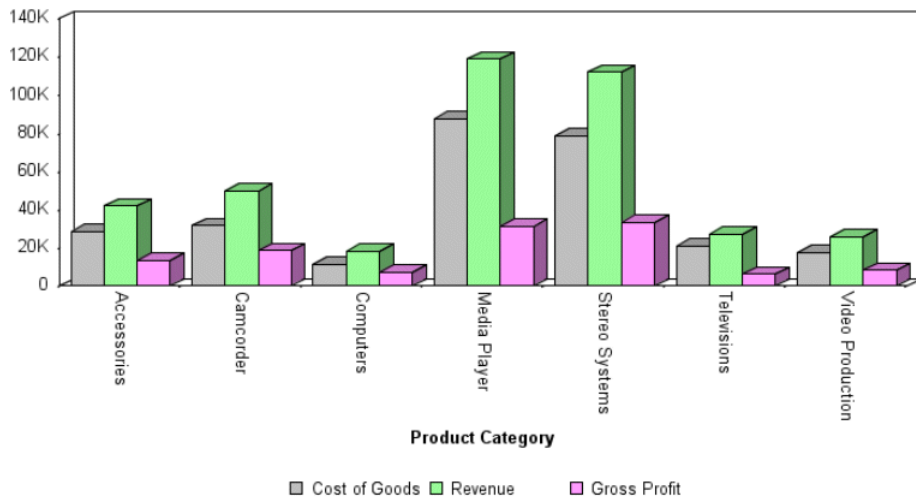
```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setFillColor(getSeries(0),new Color(192,192,192,0));
setPlace(true);
*END
ENDSTYLE
END
```

The output is:



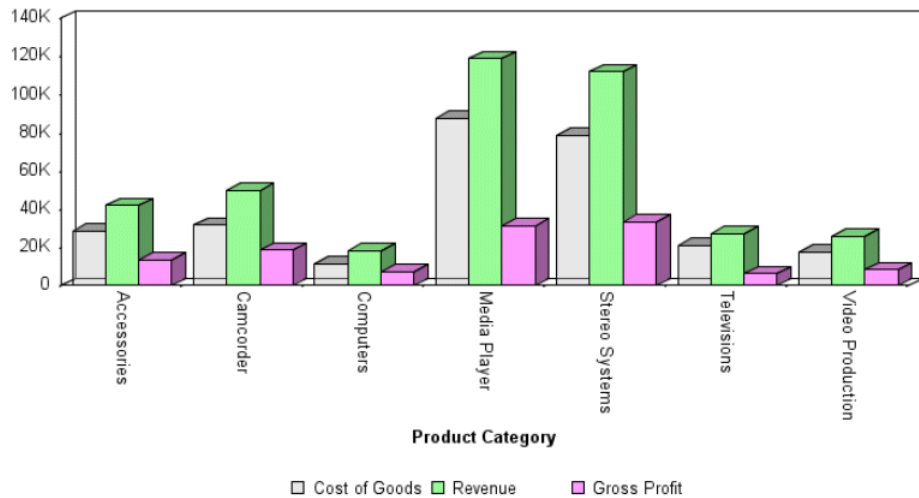
Changing the transparency value to 255 makes the fill totally opaque:

```
setFillColor(getSeries(0),new Color(192,192,192,255));
```



Changing the transparency value to 100 makes the fill partially transparent:

```
setFillColor(getSeries(0),new Color(192,192,192,100));
```

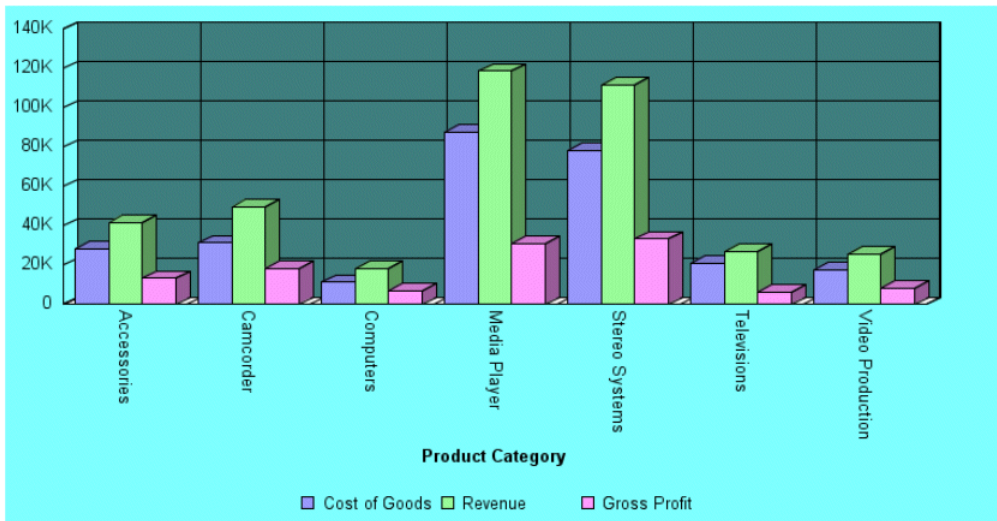


Example: Setting a Transparency Value for the Chart Background and Frame

The following request makes the chart frame black with a transparency value of 128, makes the chart background aqua with a transparency value of 128, and uses the transparent fill color for the frame side:

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US GROSS_PROFIT_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET LOOKGRAPH VBAR
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
  setFillColor(getFrame(),new Color(0,0,0,128));
  setFillColor(getChartBackground(),new Color(0,255,255,128));
  setTransparentFillColor(getFrameSide(),true);
setO1AxisLineDisplay(true);
setO1MajorGridDisplay(true);
setO1MajorGridStyle(0);
setY1AxisLineDisplay(true);
setY1MajorGridDisplay(true);
setY1MajorGridStyle(0);
setPlace(true);
*END
ENDSTYLE
END
```

The output is:



Textures and Gradients

You can use these properties and methods to apply textures and gradients to graph objects. To use a gradient or texture, the `setFillType()` method must select a gradient (`setFillType(2);`) or texture (`setFillType(3);`).

- ❑ `get/setGradientDirection();` These methods `get/set` the gradient pin direction.
- ❑ `get/setGradientNumPins();` These methods `get/set` the number of pins in a gradient.
- ❑ `get/setGradientPinLeftColor();` These methods `get/set` the left color of the gradient pin.
- ❑ `get/setGradientPinPosition();` These methods `get/set` the gradient pin position.
- ❑ `get/setGradientPinRightColor();` These methods `get/set` the right color of the gradient pin.
- ❑ `getTexture();` This method determines if a texture has been applied to a specific object in a graph or any object in a graph. When you supply a texture name (that is, a .GIF file name) as an input parameter, the method identifies whether or not the named texture is applied to any object in the graph. If you specify an object ID (for example, `getTexture (getSeries(1) );`), the method identifies whether or not a texture is applied to the specific object.
- ❑ `get/setTextureDisplayMode();` These methods `get/set` the texture display mode assigned to a specific object or all objects in a graph.
- ❑ `setTextureURL();` This method loads a texture from a file and applies it to object(s) in a graph. You must set the fill type of the object(s) to texture using the `setFillType()` method (for example, `setFillType (getSeries(1), 3);`).

Example:

The following code shows how you can use the `setFillType()` and texture methods to assign a texture to series one in a graph.

```
setFillType(getSeries(1), 3 );  
setTextureDisplayMode(getSeries(1), 0 );  
setTextureURL(getSeries(1), "Hard_Rock_Green.gif" );
```

Panning, Rotating, and Scaling

You can use the following properties and methods to pan, rotate, and scale the cube in a 3D graph:

- ❑ `CubeZoomFactor (0...100);` This property sets the global scaling factor for zooming in and out.

- ❑ CubePanX (0...100); This property pans the 3D graph cube (in 2D virtual coordinates) in the X direction within the graph frame. 0 is the default value.
- ❑ CubePanY (0...100); This property pans the 3D graph cube (in 2D virtual coordinates) in the Y direction within the graph frame. 0 is the default value.
- ❑ get/setCubeRotationMatrix(); These methods get/set the rotation matrix of the cube in a 3D graph.

Note: All scroll, zoom, and pan operations require that the graph be rendered as an applet in the browser (that is, SET GRAPHEDIT = {ON|OFF}).

Graph Editing Characteristics

You can use these properties to enable/disable and select the level at which graph objects can be edited in the user interface:

- ❑ HighlightBarResize (true/false); When you enable this property (true), the riser's value is shown when the user clicks and drags the riser in a bar graph.
- ❑ ReshapeEnable (true/false); This property enables (true)/disables (false) the user's ability to move and resize objects in a graph. True is the default value.
- ❑ ResizeBarMode (true/false); This property enables (true)/disables (false) the user's ability to resize bars in a graph. False is the default value. When this property is enabled, the user can use the control key and mouse to resize the bars in a graph. This action also adjusts the value associated with the riser.
- ❑ SelectionEnable (0...5); This property defines the user's ability to select objects in a graph and the operations that can be performed when one or more objects are selected. When you use:
 - ❑ SelectionEnable(1), the user can select individual objects (risers, markers, labels, and so on) in a graph. The graphing engine highlights each object that is selected.
 - ❑ SelectionEnable(2), the user can only select data-related objects (bars, legends, and so on). The graphing engine highlights the selected data objects and all related objects.
 - ❑ SelectionEnable(3), the user cannot select individual objects. When one object is selected, all related objects are also selected and highlighted.
 - ❑ SelectionEnable(4), the user can click on a single point and have the graph zoom at the rate of 50% on both axes with the selected point used as the center of the zoom rectangle.

- ❑ `SelectionEnable(5)`, the user can drag out a rectangle and zoom in on the data within the rectangle area. When you use this mode, a single mouse click backs up one data-zoom level. A double-click restores the original data state.

See [Data Zooming](#) on page 191 for more information about using `SelectionEnable(4)` and `SelectionEnable(5)`.

- ❑ `SelectionEnableMove (true/false)`; This property enables (true)/disables (false) the user's ability to select and move individual objects (risers, markers, labels, and so on) in a graph. True is the default value.

Also see [Colors and Shading](#) on page 155 for more information about how objects are highlighted when they are selected in the user interface.

Annotations

Multiple instances of two objects are all that is required to create annotations. They are the label and box drawn behind it (if any). Both the label and box have attributes associated with them. These attributes can be set using methods in `Perspective for Java` that are available for other chart objects (for example, `setTextString()`, `setFillColor()`, and so on)

Object IDs

The following object IDs are used to create annotations:

- ❑ `IdentObj getAnnotation(int index)`
- ❑ `IdentObj getAnnotationBox(int index)`

where:

[*index*](#)

Is a value in the range zero (0) to the maximum number of annotations set by `setMaxNumAnnotations(int nMax)`. The default maximum number of annotations is 30.

Note that the title, subtitle and footnote objects have the same behavior as annotations. Their Object IDs are returned by the following methods:

- ❑ `IdentObj getFootnote()`
- ❑ `IdentObj getFootnoteBox()`
- ❑ `IdentObj getSubtitle()`
- ❑ `IdentObj getSubTitleBox()`
- ❑ `IdentObj getTitle()`

```
IdentObj getTitleBox()
```

Enable Annotation

The `setDisplay()` method must be used to enable the annotation object:

```
setDisplay(getAnnotation(index), true);
```

Define Annotation Label/String

The `setTextString()` method can be used to define an annotation's label/string:

```
setTextString(getAnnotation(index), "string");
```

Set Annotation Text Color

The `setFillColor()` method can be used to define the color of the annotation label/string:

```
setFillColor(getAnnotation(index), new Color(0,255,0));
```

Note that all other fill types are also available for annotation labels (that is, gradient, texture, and so on).

Set Annotation Font Size

The normal font size methods can be used to set the font size of an annotation label.

Example:

```
setFontSizeAbsolute(getAnnotation(0), true);
setFontSizeInPoints(getAnnotation(0), 20);
```

Set Annotation Location

The `setRect()` function can be used to specify the location of an annotation object.

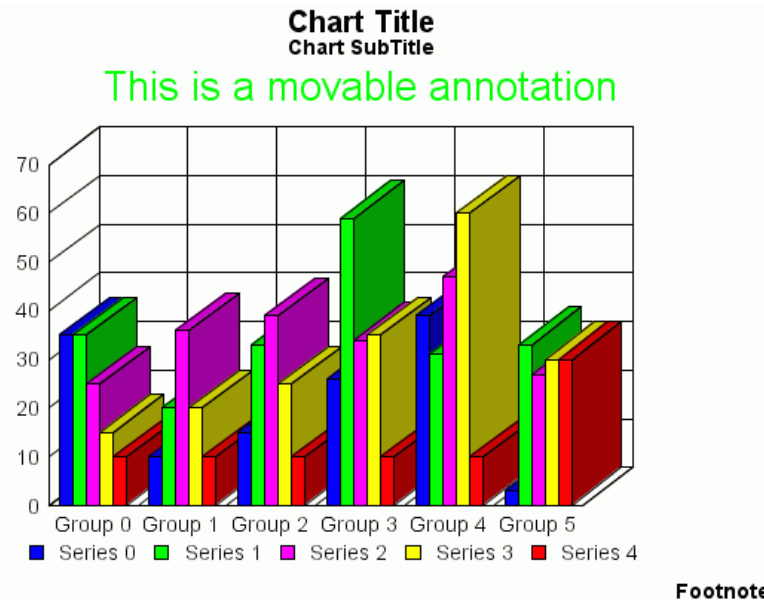
Example:

```
setRect(getAnnotation(0), new Rectangle(-15340, 11000, 29850, 1800));
```

Note: If you do not specify a location, the annotation text is drawn in the default location (the center of the chart) and may not be detectable. This is due to the fact that a rectangle is not internally stored for the annotation. Unless specified otherwise, it will use the default rectangle set by `setRect()`. It is recommended you store a rectangle location for each annotation.

Example:

```
setDisplay(getAnnotation(0), true);
setTextString(getAnnotation(0), "This is a movable annotation");
setFillColor(getAnnotation(0), new Color(0,255,0));
setRect(getAnnotation(0),new Rectangle(-15340,11000,29850,1800));
setFontSizeAbsolute(getAnnotation(0),true);
setFontSizeInPoints(getAnnotation(0),20.0);
```

**Footnote****Enable Annotation Box**

The `setDisplay()` method must be used to enable the annotation box object:

```
❑ setDisplay(getAnnotationBox(index), true);
```

Set Annotation Box Color

The `setFillColor()` method can be used to define the color of the annotation box:

```
❑ setFillColor(getAnnotationBox(0), new Color(0,0,255));
```

Enable/Disable Annotation Box Border Color

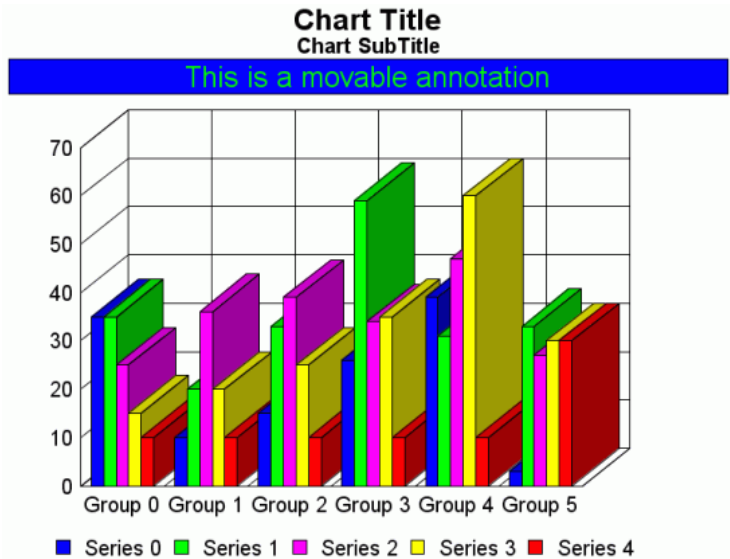
The `setTransparentBorderColor()` method can be used to enable/disable the border color around the annotation box:

- ❑ `setTransparentBorderColor(getAnnotationBox(0), false);`
- ❑ `setTransparentFillColor(getAnnotationBox(0), true);`

See [Beveled Borders](#) on page 178 for information about applying a border to an annotation box.

Example:

```
setDisplay(getAnnotation(0), true);
setTextString(getAnnotation(0), "This is a movable annotation");
setFillColor(getAnnotation(0), new Color(0,255,0));
setRect(getAnnotation(0),new Rectangle(-15340,11000,29850,1800));
setFontSizeAbsolute(getAnnotation(0),true);
setFontSizeInPoints(getAnnotation(0),20.0);
setDisplay(getAnnotationBox(0), true);
setFillColor(getAnnotationBox(0), new Color(0,0,255));
setTransparentBorderColor(getAnnotationBox(0), false);
```



Initial/Default Annotation Box Settings

The default maximum number of annotations is 30. For this first default set of annotations, the transparent fill color is set to true and border colors are omitted for the associated annotation box:

- ❑ `setTransparentFillColor(getAnnotationBox(nnn), true);`
- ❑ `setTransparentBorderColor(getAnnotationBox(nnn), true);`

Maximum Number of Annotations

The following methods can be used to get/set the maximum number of annotations:

- ❑ `void setMaxNumAnnotations(int newValue);`
- ❑ `int getMaxNumAnnotations();`

where:

newValue

Is a value in the range 0...999.

For annotations in excess of the default (30), the `setTransparentFillColor()` and `setTransparentBorderColor()` methods should be used to set transparent fill and border colors for the associated annotation box as desired:

- ❑ `setTransparentFillColor(getAnnotationBox(nnn), true | false);`
- ❑ `setTransparentBorderColor(getAnnotationBox(nnn), true | false);`

Drill-Down and URLs

The URL methods allow you to include drill-downs in HTML files. When the user selects/clicks on an object in a graph where a `setURL()` is defined, the HTML file provided as an input parameter to the method will automatically be loaded and displayed at the location specified by the `setURLTarget()`. For example, in the [Example Drill-Down](#) on page 171, `setURL()` selects a different HTML file for each series/group (slice) in a pie graph. The example `setURLTarget()` defines the location in the page where each HTML file will appear.

URL Methods

- ❑ `get/setURL();` These methods get/set a Universal Resource Locator (URL) for any object.
- ❑ `get/setURLTarget();` These methods get/set the frame target for a URL associated with any specific object.

Example Drill-Down

```

<html><head><title>
Pie Drill Down Example</title></head><body><p>
<applet code="TDGChartApplet.class" archive="javaCHARTg.jar" width="400"
height="280"><param name="TDGSCRIPT"
value="setTitleString(&quot;Pie Chart with Drilldown&quot;);
setSubtitleString(&quot;Click on the
slices to see a description&quot;);
setGraphType(55);
setSeriesLabelArray(&quot;Monday&quot;;
&quot;Tuesday&quot;;&quot;Wednesday&quot;;
&quot;Thursday&quot;;&quot;Friday&quot;);
setDataSeries(35.00);
setDataSeries(27.00);
setDataSeries(15.00);
setDataSeries(17.00);
setDataSeries(27.00); setURL(0,0,&quot;series0.html&quot;); setURL(1,
0,&quot;series1.html&quot;); setURL(2, 0,&quot;series2.html&quot;);
setURL(3, 0,&quot;series3.html&quot;); setURL(4,
0,&quot;series4.html&quot;); setURLTarget(0,0, &quot;Bottom&quot;);
setURLTarget(1,0, &quot;Bottom&quot;); setURLTarget(2,0,
&quot;Bottom&quot;); setURLTarget(3,0, &quot;Bottom&quot;);
setURLTarget(4,0, &quot;Bottom&quot;);
setToolTipDisplay(1);
"></applet></p><p><br><br>
<a href="series2.html" target="Bottom">
Show the Series2 page by force</a> </p></body></html>

```

Multiple Drill-Down From Graph

You can create multiple drill-down links from a graph by choosing any combination of supported actions. When you click a component in the graph, a pop-up menu appears listing the drill-down options. For example, you can create links to a detail report or Maintain procedure, a JavaScript function, or a URL.

This feature is available in graph code only. It applies to data fields in graph, not to headings, subheadings, footings, or subfootings.

Syntax: How to Create Multiple Drill-Down Links

TYPE=type, [subtype], DRILLMENUITEM='description'|'DrillDown_n, type_of_link

where:

type

Identifies the report component that you select in the web browser to execute the link. The TYPE attribute and its value must appear at the beginning of the declaration.

subtype

Are any additional attributes, such as COLUMN, LINE, or ITEM, that are needed to identify the report component that you are formatting.

description

Is the text that appears on the pop-up menu of drill-down options on the report output. The default value is DrillDown *n*.

n

Is a consecutive integer, such as DrillDown 1, DrillDown 2, and so on.

type_of_link

Is the type of link, for example, a link to a detail report or URL.

Example: Creating Multiple Drill-Down Links

There are two procedures shown in the following example. The bar graph procedure uses the DRILLMENUITEM and URL attributes to create a link to a website, while the DRILLMENUITEM and FOCEXEC attributes are used to create a link to the detail report.

Bar Graph:

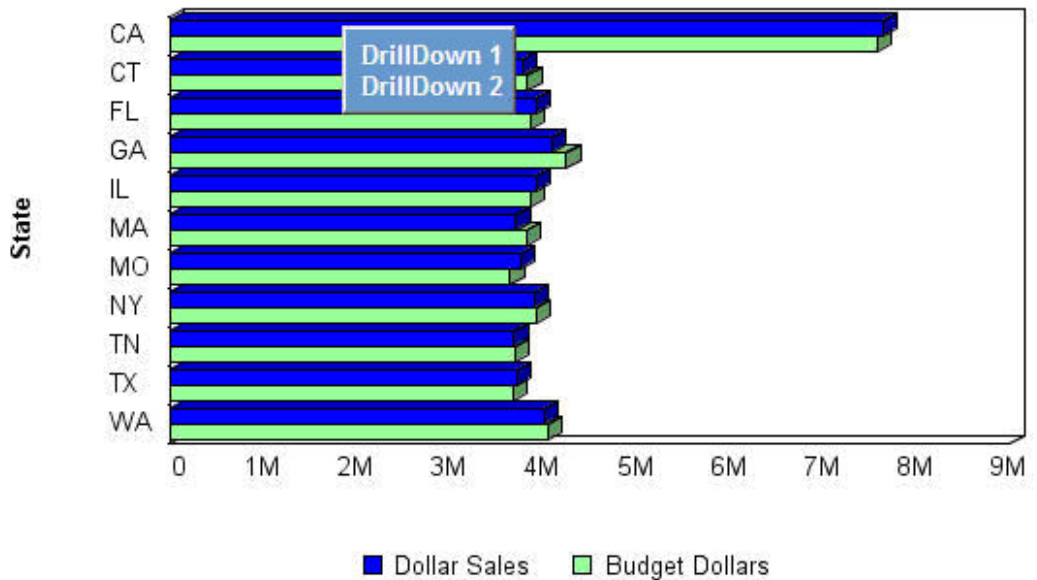
```
GRAPH FILE GGSales
SUM
    DOLLARS/D12.2CM
    BUDDOLLARS/D12.2CM
BY ST
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET STYLE *
    COLUMN=N2,
    COLOR='BLUE',
    STYLE=UNDERLINE,
    DRILLMENUITEM='DrillDown 1',
        URL=http://www.informationbuilders.com,
    DRILLMENUITEM='DrillDown 2',
        FOCEXEC=DETAILREPORT(PARAMETER=N1),$
ENDSTYLE
END
```


Detail Report:

```

TABLE FILE GGSALES
PRINT
      DOLLARS/D12.2CM
      BUDDOLLARS/D12.2CM
BY ST
WHERE (ST EQ '&PARAMETER');
ON TABLE SET PAGE-NUM OFF
ON TABLE NOTOTAL
ON TABLE SET ONLINE-FMT HTML
ON TABLE SET HTMLCSS ON
ON TABLE SET STYLE *
TYPE=REPORT,
      GRID=OFF,
      FRONT=ARIAL,
      SIZE=9,
      COLOR=BLACK,
      BACKCOLOR=NONE,
      STYLE=NORMAL,
      TOPGAP=0.013889,
      BOTTOMGAP=0.027778,$
ENDSTYLE
END

```



If you click *DrillDown1* from the image, you will be taken to the URL that was specified in the graph request. Clicking *DrillDown2* will display a detailed report, as shown in the following image.

<u>State</u>	<u>Dollar Sales</u>	<u>Budget Dollars</u>
CA	\$25,005.00	\$17,568.00
	\$23,282.00	\$14,640.00
	\$23,153.00	\$19,008.00
	\$22,785.00	\$21,960.00
	\$21,780.00	\$19,760.00
	\$19,956.00	\$21,945.00
	\$19,872.00	\$23,044.00
	\$19,162.00	\$18,576.00
	\$18,096.00	\$15,390.00
	\$17,916.00	\$16,808.00

Note: The user is unable to drill down to a MR focexec and a Reporting server focexec from the same report, due to the fact that these are mutually exclusive.

Alternate Formatting

Alternate formatting objects and methods can be used to apply different colors to sections of an axis and axis labels. You may also define a text label to show with an alternate formatted section.

Alternate Formatting Objects

The following objects support alternate formatting of an axis and axis labels.

- ❑ IdentObj getY1AltFmtFrame()
- ❑ IdentObj getY1AltFmtFrame(int nIndex)
- ❑ IdentObj getY2AltFmtFrame()
- ❑ IdentObj getY2AltFmtFrame(int nIndex)
- ❑ IdentObj getY3AltFmtFrame()
- ❑ IdentObj getY3AltFmtFrame(int nIndex)
- ❑ IdentObj getY4AltFmtFrame()

- ❑ IdentObj getY4AltFmtFrame(int nIndex)
- ❑ IdentObj getY5AltFmtFrame()
- ❑ IdentObj getY5AltFmtFrame(int nIndex)
- ❑ IdentObj getO1AltFmtFrame()
- ❑ IdentObj getO1AltFmtFrame(int nIndex)
- ❑ IdentObj getX1AltFmtFrame()
- ❑ IdentObj getX1AltFmtFrame(int nIndex)
- ❑ IdentObj getY1AltFmtLabel()
- ❑ IdentObj getY1AltFmtLabel(int nIndex)
- ❑ IdentObj getY2AltFmtLabel()
- ❑ IdentObj getY2AltFmtLabel(int nIndex)
- ❑ IdentObj getY3AltFmtLabel()
- ❑ IdentObj getY3AltFmtLabel(int nIndex)
- ❑ IdentObj getY4AltFmtLabel()
- ❑ IdentObj getY4AltFmtLabel(int nIndex)
- ❑ IdentObj getY5AltFmtLabel()
- ❑ IdentObj getY5AltFmtLabel(int nIndex)
- ❑ IdentObj getO1AltFmtLabel()
- ❑ IdentObj getO1AltFmtLabel(int nIndex)
- ❑ IdentObj getX1AltFmtLabel()
- ❑ IdentObj getX1AltFmtLabel(int nIndex)

Use setDisplay() and omit the nIndex parameter to enable alternate formatting of the axis and/or axis labels.

Example:

```
setDisplay(getY1AltFmtFrame(),true);//Enable alt format of Y1Axis
setDisplay(getY1AltFmtLabel(),true);//Enable alt format of labels
```

When alternate formatting is enabled for an axis, use the alternate formatting methods to define how many sections/colors will be defined and the location on the axis to separate the sections.

Alternate Formatting Methods

- ❑ `get/setAltFmtFrameNumColors()`; Get/set the number of different colors/sections where alternate formatting can be applied to a specified axis.
- ❑ `get/setAltFmtFrameSeparator()`; Get/set the position of alternate formatted sections on a specified axis.

get/setAltFmtFrameNumColors()

These methods `get/set` the number of colors/sections to be used for alternate formatting of an axis. Alternate formatting can be used to assign different colors to individual sections of an axis and axis labels. You may also define a text string label that is shown with the alternate formatted section.

```
int getAltFmtFrameNumColors (IdentObj idObj);  
void setAltFmtFrameNumColors (IdentObj idObj, int newValue);
```

where:

idObj

Is the object ID of an alternate formatting axis object (for example, `getY1AltFmtFrame()`, `getO1AltFmtFrame()`, and so on).

newValue

For `setAltFmtFrameNumColors()`, number of colors/sections to use for alternate formatting of the specified axis (*idObj*). It can be a value in the range 1...100. The default value is 2.

Return:

int; For `getAltFmtFrameNumColors()`, number of colors/sections that are used for alternate formatting of the specified axis (*idObj*).

get/setAltFmtFrameSeparator()

These methods `get/set` the location on an axis where alternate formatting is applied.

Syntax:

```
double getAltFmtFrameSeparator (IdentObj idObj);  
void setAltFmtFrameSeparator(IdentObj idObj, double newValue);
```

where:

idObj

Is the object ID of an alternate formatting axis object (for example, `getY1AltFmtFrame()`, `getO1AltFmtFrame()`, and so on).

newValue;

For `setAltFmtFrameSeparator()`, the location on the axis where alternate formatting will be applied. For a numeric axis, specify a value that is within the range of values on the specified axis. For the ordinal (O1) axis, specify a group number that is within the range of groups on the ordinal axis.

Return:

double; For `getAltFmtFrameSeparator()`, the location on the axis where alternate formatting has been applied.

Applying Colors and Labels

After an alternate formatting object has been enabled with `setDisplay()` and alternate formatting methods define the number of colors/sections and frame separator location, use the `setDisplay()` method again to enable each section and `setFillColor()` to apply colors to each section.

Example:

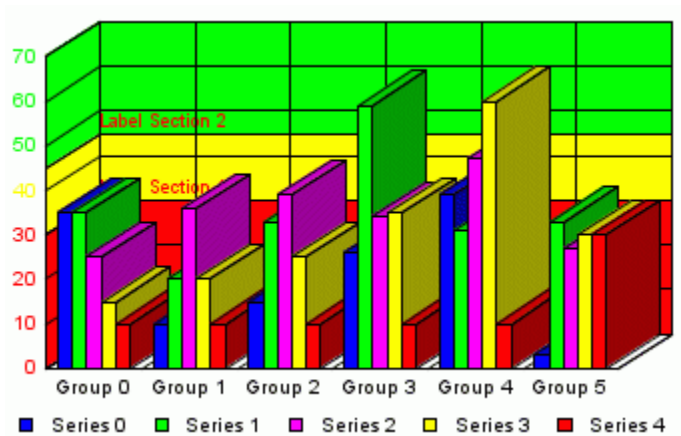
```
setDisplay(getY1AltFmtFrame(), true);
setDisplay(getY1AltFmtLabel(), true);
setAltFmtFrameNumColors(getY1AltFmtFrame(), 3);
setAltFmtFrameSeparator(getY1AltFmtFrame(0), 30.0);
setAltFmtFrameSeparator(getY1AltFmtFrame(1), 45.0);
setAltFmtMatchAxisLabelColor(true);
setDisplay(getY1AltFmtFrame(0), true);
setDisplay(getY1AltFmtFrame(1), true);
setDisplay(getY1AltFmtFrame(2), true);
setFillColor(getY1AltFmtFrame(0), new Color(255, 0, 0));
setFillColor(getY1AltFmtFrame(1), new Color(255, 255, 0));
setFillColor(getY1AltFmtFrame(2), new Color(0, 255, 0));
setFillColor(getY1AltFmtLabel(0), new Color(255, 0, 0));
setFillColor(getY1AltFmtLabel(1), new Color(255, 0, 0));
setFillColor(getY1AltFmtLabel(2), new Color(255, 0, 0));
```

Use the `setTextString()` method to define section labels.

Example:

```
setTextString(getY1AltFmtLabel(0), "Label Section 0");
setTextString(getY1AltFmtLabel(1), "Label Section 1");
setTextString(getY1AltFmtLabel(2), "Label Section 2");
```

These two code segments produce a chart in the following format:



See [AltFmtMatchAxisLabelColor](#) on page 246.

Beveled Borders

Beveled borders can be applied to annotations, title, subtitle, footnote, 2D risers, legend box and legend markers.

Object IDs

Beveled borders can be applied to the following objects:

- ☐ IdentObj getAnnotationBox(int index)
- ☐ IdentObj getBeveledLegend()
- ☐ IdentObj getBeveledLegendMarker()
- ☐ IdentObj getBeveledRiser()
- ☐ IdentObj getFootnoteBox()
- ☐ IdentObj getSubtitleBox()
- ☐ IdentObj getTitleBox()

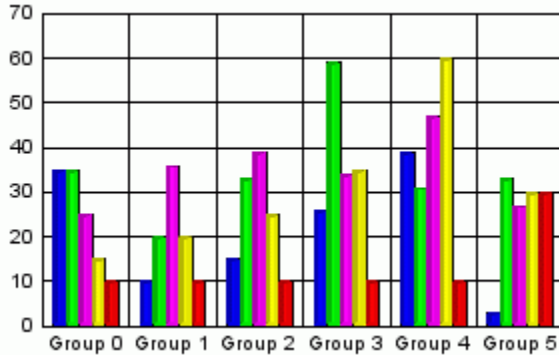
Enable/Disable Beveled Borders

The setDisplay() method enables/disables beveled borders for a specified object:

```
setDisplay(IdentObj idObj, Boolean bValue);
```

Example:

```
setDepthRadius(0);
setDisplay(getBeveledRiser(), true);
```



For the `getBeveledLegend()`, `getBeveledLegendMarker()`, and `getBeveledRiser()` object, the `setDisplay()` method is all that is needed to show default beveled borders. For annotation, footnote, subtitle, and title box objects, the `setDisplay()` method enables the bounding box around the object. The border type and insets must also be set for these objects.

get/setBorderType()

These methods get/set the border type for a specified object. The default value is NONE.

Syntax:

```
void setBorderType( IdentObj IdObj, int nBorderType );
int setBorderType( IdentObj IdObj );
```

where:

IdObj

Is the object ID returned by `getAnnotationBox()`, `getBeveledLegend()`, `getBeveledLegendMarker()`, `getBeveledRiser()`, `getFootnoteBox()`, `getSubtitleBox()`, or `getTitleBox()`.

nBorderType

One of the following border types:

- ☐ BORDER_NONE = 0;
- ☐ BORDER_BEVEL = 1;
- ☐ BORDER_BEVEL_RAISED = 2;

- ❑ `BORDER_BEVEL_LOWERED = 3;`
- ❑ `BORDER_LINE = 4;`
- ❑ `BORDER_LINE_DOUBLE = 5;`

Return:

int; For `getBorderType()`, 0...5 identifies the border type applied to a specified object.

get/set...Insets()

These methods get/set the top/bottom and right/left insets of borders of a specified object. The default value is 1 (which is essentially off).

Syntax:

```
int getBottomInset(IdentObj idObj);
void setBottomInset(IdentObj idObj, int newValue);
int getLeftInset(IdentObj idObj);
void setLeftInset(IdentObj idObj, int newValue);
int getRightInset(IdentObj idObj);
void setRightInset(IdentObj idObj, int newValue);
int getTopInset(IdentObj idObj);
void setTopInset(IdentObj idObj, int newValue);
```

where:

idObj

Object ID returned by `getAnnotationBox()`, `getBeveledLegend()`, `getBeveledLegendMarker()`, `getBeveledRiser()`, `getFootnoteBox()`, `getSubtitleBox()`, or `getTitleBox()`.

newValue

Is the inset of the border in virtual coordinates.

Return:

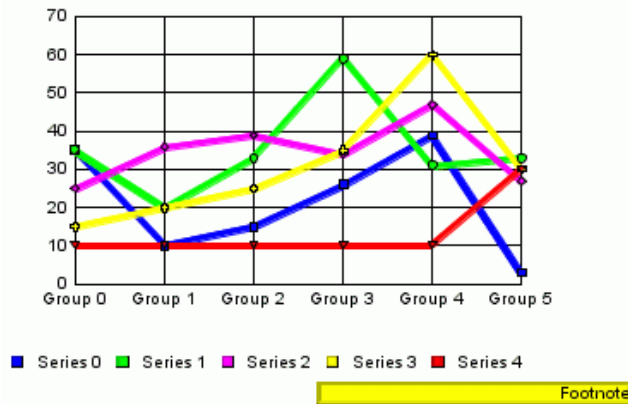
int; For `get...Inset()`, the inset of the border in virtual coordinates.

Example:

```

setDepthRadius(0);
setGraphType(41);
setDisplay(getFootnoteBox(), true);
setBorderType(getFootnoteBox(), 2);
setTopInset(getFootnoteBox(), 250);
setLeftInset(getFootnoteBox(), 250);
setBottomInset(getFootnoteBox(), 250);
setRightInset(getFootnoteBox(), 250);
setFillColor(getFootnoteBox(), new Color(255,255,0));

```

**get/setCornerArcWidth/Height()**

These methods get/set the width and height of the corner arc of a border applied to an object. The default value is 5.

Syntax:

```

int getCornerArcHeight ( IdentObj id );
void setCornerArcHeight ( IdentObj anObject, int newValue);
int getCornerArcWidth ( IdentObj id );
void setCornerArcWidth ( IdentObj anObject, int newValue);

```

where:

idObj

Is the object ID returned by getAnnotationBox(), getBeveledLegend(), getBeveledLegendMarker(), getBeveledRiser(), getFootnoteBox(), getSubtitleBox(), or getTitleBox().

newValue

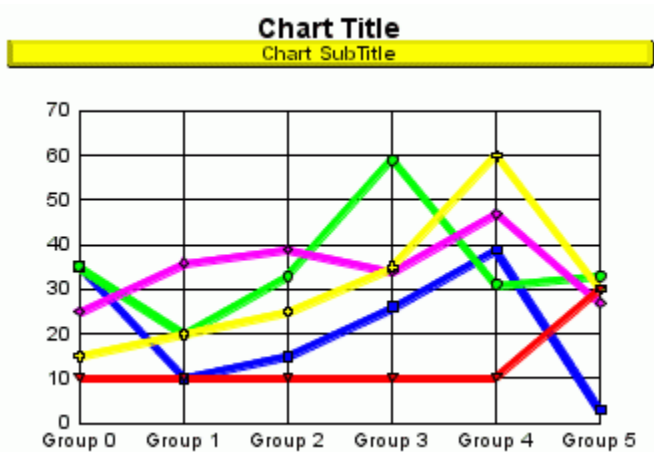
Is the height or width of corner arc in virtual coordinates.

Return:

int; Height or width of corner arc applied to object

Example:

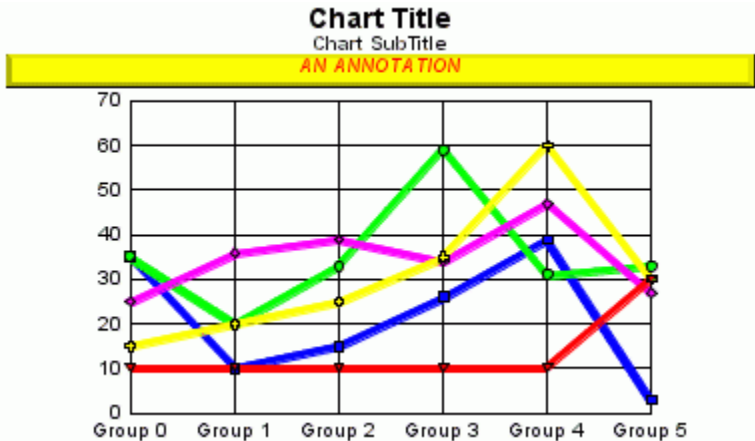
```
setDepthRadius(0);
setGraphType(41);
setDisplay(getSubtitleLabel(), true);
setBorderType(getSubtitleLabel(), 3);
setTopInset(getSubtitleLabel(), 250);
setLeftInset(getSubtitleLabel(), 250);
setBottomInset(getSubtitleLabel(), 250);
setRightInset(getSubtitleLabel(), 250);
setFillColor(getSubtitleLabel(), new Color(255,255,0));
setCornerArcWidth(getSubtitleLabel(), 500);
setCornerArcHeight(getSubtitleLabel(), 500);
```



Annotation Example

The follow example code creates an annotation and applies a beveled border to the annotation box.

```
setDisplay(getAnnotationBox(0), true);
setDisplay(getAnnotation(0), true);
setTextString(getAnnotation(0), "AN ANNOTATION");
setFillColor(getAnnotation(0), new Color (255,0,0));
setFontStyle(getAnnotation(0), 3);
setFillColor(getAnnotationBox(0), new Color(255,255,0));
setTransparentColor(getAnnotationBox(0), false);
setRect(getAnnotation(0), new Rectangle (-15340, 11000, 29850, 1800));
setBorderType(getAnnotationBox(0),3);
setTopInset(getAnnotationBox(0), 250);
setLeftInset(getAnnotationBox(0), 250);
setBottomInset(getAnnotationBox(0), 250);
setRightInset(getAnnotationBox(0), 250);
setDepthRadius(0);
setGraphType(41);
```



Reference Lines

Reference line support is implemented for various chart types excluding pie and pie bar. There can be a total of 3 reference lines for each axis (Y1, Y2, and X1). Each reference line has color and thickness. There can be text associated with each reference line, which can appear in a reference line legend. Each reference line, in addition to being assigned an axis and location, is also assigned an identifier.

Reference: APIs Related to Reference Lines

- ❑ `IdentObj getReferenceLineaxisname(int)`

This command gets the reference line object for a particular axis and specific identifier.

- ❑ `setReferenceLineValue()`

This command sets the value at which the reference line displays for the specified axis and identifier.

- ❑ `IdentObj getReferenceLineTextaxisname(int)`

This command gets the ID of the label for the specified reference line.

- ❑ `IdentObj getReferenceLine(IdentObj, int)`

This command gets the reference line object for a particular axis and specific identifier. This API returns null if the specified reference line does not exist.

- ❑ `void setReferenceLine(IdentObj , int, double)`

This command creates a reference line for a particular axis and specified identifier.

- ❑ `double getReferenceLineValue(IdentObj , int)`

This command returns the location of a reference line for a particular axis and specific identifier.

- ❑ `void removeReferenceLineObject(IdentObj, int)`

This command removes the specified reference line object.

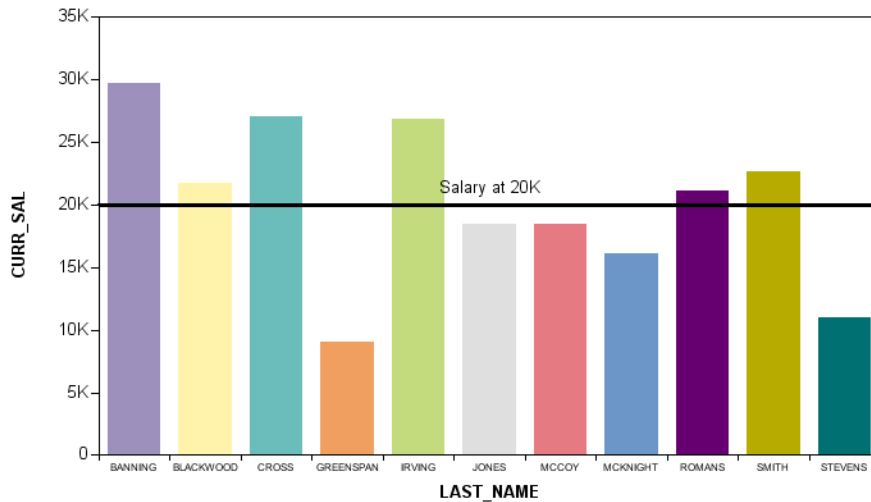
Reference: Properties Supported by Reference Lines

- ❑ Width.
- ❑ Color.
- ❑ Display.
- ❑ Value or location.

Example: Creating a Reference Line With a Label

The following code turns on a reference line for the Y1 axis, sets the value at which the line displays to 20,000, creates a label for the reference line, and turns on the display of the label:

```
setDisplay(getReferenceLineY1(0),true);
setReferenceLineValueY1(0,20000.0);
setLineWidth(getReferenceLineY1(0),2);
setTextString(getReferenceLineTextY1(0),"Salary at 20K");
setDisplay(getReferenceLineTextY1(0),true);
```

**Example: Creating a Reference Line**

The following example creates a Y1 Axis reference line at the y = 50 with the color red and a width of 3.

```
setReferenceLine(getY1Axis(),0, 50.0);
setLineWidth(getReferenceLine(getY1Axis(),0),3);
setFillColor(getReferenceLine(getY1Axis(),0), new Color(255,0,0));
```

The following line hides a reference line:

```
setDisplay(getReferenceLine(getY1Axis(),0),false);
```

The following line shows how to associate text with the reference line:

```
setTextString(getReferenceLine(getY1Axis(),50),"label");
```

APIs Related to the Reference Line Legend

- ❏ IdentObj getReferenceLineLegendText(IdentObj ,int)

Gets the reference line legend text IdentObj.

The following line hides the legend text for a reference line:

```
setDisplay(getReferenceLineLegendText(getY1Axis(),0),false)
```

Using Data in a Graph

The following topics describe the use of data in a graph, including managing data overflow and accounting for missing data.

In this chapter:

- ❑ [Graph Titles and Labels](#)
 - ❑ [Data Scrolling and Zooming](#)
 - ❑ [How Missing Data Is Handled](#)
 - ❑ [Data Requirements for Different Graph Types](#)
-

Graph Titles and Labels

The following properties are used to define graph-wide titles:

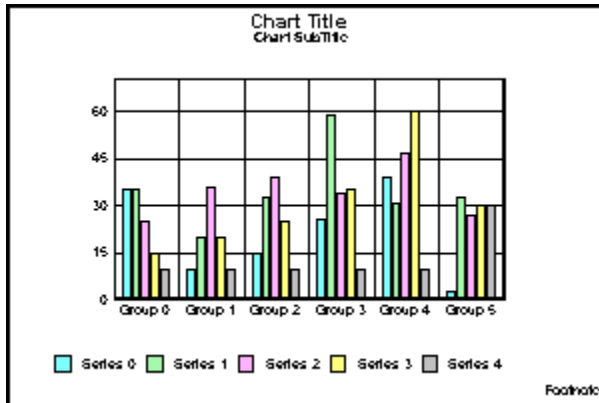
- ❑ `FootnoteString`
- ❑ `SubtitleString`
- ❑ `TitleString`

The following properties are used to define graph axis labels:

- ❑ `O1TitleString`
- ❑ `O2TitleString`
- ❑ `X1TitleString`
- ❑ `Y1TitleString`
- ❑ `Y2TitleString`

Data Scrolling and Zooming

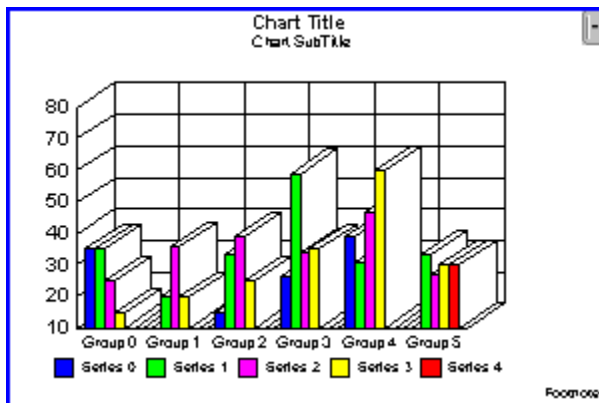
Regardless of the data that is assigned to each series/group, the range of data that is imaged on a numeric axis can be limited by the {Axis}ScaleMin and {Axis}ScaleMax properties. For example, given a graph in the following format:



The Y1ScaleMax, Y1ScaleMaxAuto, Y1ScaleMin, and Y1ScaleMinAuto properties can be used to modify the range of values displayed on the Y1 axis as shown in the following example:

```
setY1ScaleMaxAuto(false);
setY1MajorGridStepAuto(false);
setY1MinorGridStepAuto(false);
setY1ScaleMax(80.0);
setY1ScaleMinAuto(false);
setY1ScaleMin(10.0);
```

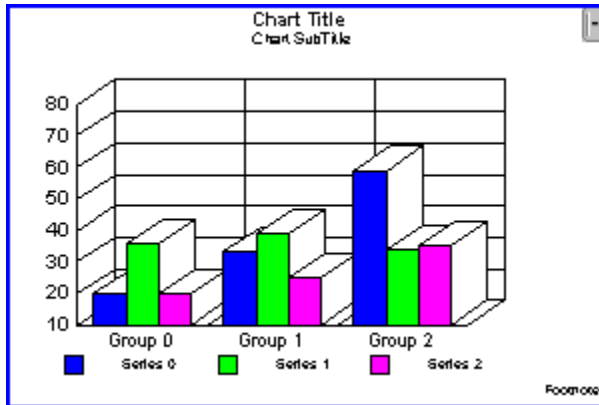
When these properties are set, the graph is imaged in the following format:



The `setDataRange()` method can be used to limit the number of series/groups (rows/columns) that are imaged in the graph.

Example:

```
setDataRange(1, 1, 3, 3);
```



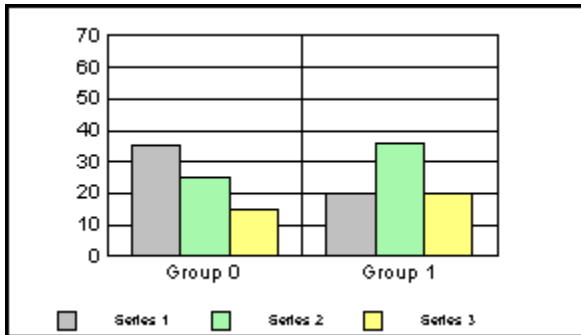
Data Scrolling

The following properties can also be used to limit the range of data that is imaged in the graph:

- ☐ `ScrollOffsetGroup`: This property specifies the first group to be imaged in the graph.
- ☐ `ScrollOffsetSeries`: This property specifies the first series to be imaged in the graph.
- ☐ `ViewableGroups`: This property specifies the maximum number of groups to be imaged in the graph.
- ☐ `ViewableGroupsStock`: This property specifies the maximum number of groups to be imaged in a stock graph.
- ☐ `ViewableSeries`: This property specifies the maximum number of series to be imaged in the graph.
- ☐ `ViewableSeriesStock`: This property specifies the maximum number of series to be imaged in a stock graph.

Example:

```
setDepthAngle(0);  
setDepthRadius(0);  
setScrollOffsetGroups(1);  
setScrollOffsetSeries(1);  
setViewableGroups(2);  
setViewableSeries(3);
```

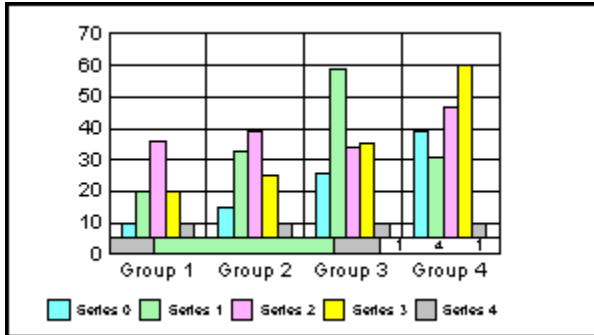


When the number of viewable groups or series is set to a value less than the number of groups/series in the data set that defines the graph, you can use these properties to display data scroll bars in the graph:

- ❑ **DataScrollerPresenceGroups:** This property controls the presence of data scroll bars on the group axis. When the number of viewable groups is set to a value less than the number of groups in the data set, the scroll bar can be used to change the first group in the graph that is displayed (that is, it effectively changes the value assigned to `ScrollOffsetGroup`).
- ❑ **DataScrollerPresenceSeries:** This property controls the presence of data scroll bars on the series axis. When the number of viewable series is set to a value less than the number of series in the data set, the scroll bar can be used to change the series group in the graph that is displayed (that is, it effectively changes the value assigned to `ScrollOffsetSeries`).

Example:

```
setDataScrollerPresenceGroups(2);
setViewableGroups(4);
```



The data scroll bar consists of two segments:

1. A segment that can be used to click and drag to the first group to be displayed.
2. A status bar that shows the number of the first group (that is, the ScrollOffsetGroup value), the total number of viewable groups, and the remaining number of groups in the data set.

In the first segment, the darker, movable thumb can be dragged to change the group offset. In the second segment, the total of the three numbers indicates the total number of groups defined in the data set. For example, the status area of the scroll bar displays 2 3 1. These numbers indicate that ScrollOffsetGroup is 2, ViewableGroups is 3, there is 1 group in the data set that is not displayed, and there are a total of six groups defined in the data set. The user can click on these numbers and edit them with the arrow keys.

Data Zooming

Important: The following is applicable only when graphs are rendered in applets using SET GRAPHEDIT=OFF.

The following properties and methods can be used to zoom in, zoom out, and identify data zooming parameters:

- ❑ `clearZoomList()`: This method clears all entries in the zoom list and returns the graph to its original state.
- ❑ `get/setZoomIndex()`: This method gets/sets the current zoom state index. It sets the level of zooming to go to. Only values inside the current zoom index may be used. Use `isValidZoomIndex()` to determine if the zoom index value is valid.

- ❑ `isValidZoomIndex(value)`: This method returns true or false indicating whether or not the value is a valid zoom index.
- ❑ `isZoomingArmed()`: This method returns true or false indicating whether or not at least one zooming operation has been performed on the graph.
- ❑ `ZoomDirection`: This property selects the direction (X-axis, Y-axis, or both) in which data zooming operations are performed by `doZoomIn()` and `doZoomOut()`.
- ❑ `ZoomPercentage`: This property specifies the percentage of zoom to be performed by `doZoomIn()` and `doZoomOut()`.

In addition to these properties and methods, you can also enable data zooming and scrolling via the user interface with the `SelectionEnable` property. When `setSelectionEnable(5)` is used, zooming and scrolling functions are enabled. When this mode is used, it allows the user to drag out a rectangle and zoom in on the data within the rectangle area. A single mouse click will back up one data-zoom level. Double click will restore the original data state. Note that user interface-based zooming and scrolling is mutually exclusive with selecting graph items. When `setSelectionEnable(5)` is used, graph objects cannot be selected.

When data zooming is enabled with the zoom properties/methods or via the user interface, a zoom index will be kept that keeps track of the number of times a graph has been zoomed in and the state of zooming. As an example, assume `ZoomPercentage` is set to 10%, and the user clicks once. The graph is now zoomed to 90% of its original values, and the zoom index is now 1. Now the user clicks a second time. The graph is now zoomed to 80% of its original values, and the zoom index now is 2. Or, actually, the zoom index now has two entries: one for the first, 90% state, and a second for the second, 80% state.

You can keep adding to the zoom index as the user continues to click. Continuing with the example and the same values, assume the user has clicked eight consecutive times. The zoom index now contains eight items:

1. First Level: Zoom Factor 90%
2. Second Level: Zoom Factor 80%
3. Third Level: Zoom Factor 70%
4. Fourth Level: Zoom Factor 60%
5. Fifth Level: Zoom Factor 50%
6. Sixth Level: Zoom Factor 40%
7. Seventh Level: Zoom Factor 30%
8. Eighth Level: Zoom Factor 20%

Your application can go back and forth in this list. For example, assume you want to switch from the current level (the last level the user clicked, the eighth zoom level which currently shows 20% of the original state). `setZoomIndex(4)` can be used to immediately switch to the fourth level which had a zoom factor of 60%. You can use `setZoomIndex` to switch back and forth between any zoom level set.

How Missing Data Is Handled

By default, the following rules govern how missing data points are handled:

1. When a data point is part of a cumulative total (stacked or percent bar, line, area, pie slice), it is interpreted as 0.0.
2. When a data point is used with neighboring points to define a line segment or surface (3D connected, 3D surface, 2D line, 2D area), it is interpreted as 0.0.
3. Otherwise (2D side-by-side bar, 3D riser, 3D floating, 2D and 3D scatter, stock graphs, spectral map, histogram), the (missing) point will not be plotted.

The `FillMissingData` and `NumMissingDataSegments` properties can be used to override these rules and define how missing data points are handled.

Data Requirements for Different Graph Types

Regardless of the data import method, data will be interpreted differently based on the selected graph type. The following table shows the data values that are required for each graph type.

GraphType	Required Data Values per Row (Series)
3D (0...16)	One value per marker.
2D (17...54)	One value per marker.
Pie (55...60)	One value per marker.
XY Scatter (61...62)	Two values per marker, X and Y, in that order.
XY Scatter with Labels (63...64)	Three values per maker, X, Y, and text label, in that order. Each XY point is discreetly labeled.
Polar (65...66)	Two values per marker in the following order: X (degree) and Y (distance from the center).
Radar (67...69)	One value per marker.

GraphType	Required Data Values per Row (Series)
Candle: Open/High/Low/Close (70)	Four values per marker: Open, High, Low, and Close, in that order.
Candle: Open/High/Low/Close/Volume (71)	Five values per marker: Open, High, Low, Close, and Volume.
Candle: Open/Close (72)	Two values per marker: Open and Close.
Stock: High/ Low (73...75)	Two values per marker: High and Low.
Stock: High/Low/Close (76...78)	Three values per marker: High, Low, and Close.
Stock: Open/High/Low/Close (79...81)	Four values per marker: Open, High, Low, and Close.
Stock: High/Low/Volume (82)	Three values per marker: High, Low, and Volume.
Stock: Open/High/Low/Close/Volume (83)	Five values per marker: Open, High, Low, Close, and Volume.
Candle: Open/Close/Volume (84)	Three values per marker: Open, Close, and Volume.
Histogram (85...86)	One value per marker.
Spectral Map (87)	One value per marker.
Stock: High/Low/Close/ Volume (88)	Four values per marker: High, Low, Close, and Volume.
Bubble Graph (89)	Three values per marker: X, Y, and Z, in that order.
Bubble Graph with Labels (90)	Four values per marker: X, Y, Z, and text label, in that order.
Bubble Graph Dual-Axes (91)	Three values per marker: X, Y, and Z, in that order.
Bubble Graph with Labels Dual-Axis (92)	Four values per marker: X, Y, Z, and text label, in that order.
Waterfall (100...101)	One value per marker.

GraphType	Required Data Values per Row (Series)
Pareto (102)	One value per marker.
Multi-Y (103...105)	One value per marker.
Funnel (106)	One value for each series/section of the funnel. Only one group is shown in a funnel graph.
Gauge (110)	One value per gauge.

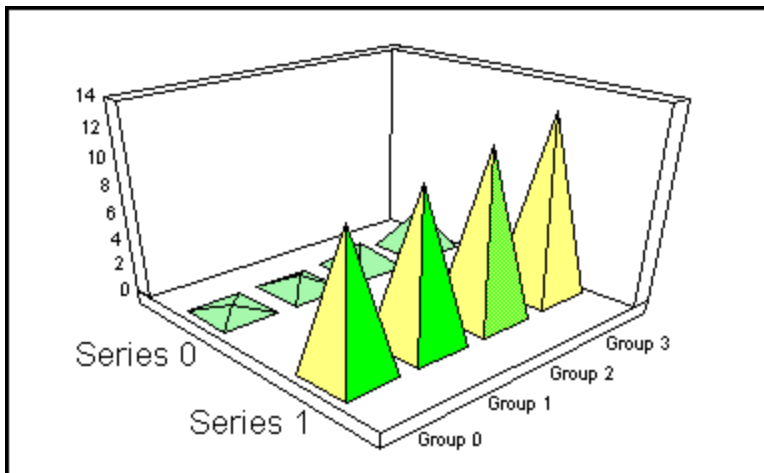
Note:

- ❑ If your application does not supply the correct number of values for each data point in your graph, you may receive the message "Not enough data to draw chart". If you are using setData() to define values, make sure you use setDataRangeToExtent() after the last setData() method to apply all data values to the graph.
- ❑ Clustered Bar, Stacked Bar, Line, and Scatter graphs may use a time scale axis. See [Time Scale Axis Graphs](#) on page 221 for special data requirements for defining a time scale axis.

Data in a 3D Graph

A 3D graph requires one value for each series/group. The following sample program creates a 3D pyramid graph with two series and four groups.

```
setGraphType(1);  
setData(0,0,.5);  
setData(0,1,1);  
setData(0,2,2);  
setData(0,3,3);  
setData(1,0,10);  
setData(1,1,11);  
setData(1,2,12);  
setData(1,3,13);  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");  
setGroupLabel(0,"Group 0");  
setGroupLabel(1,"Group 1");  
setGroupLabel(2,"Group 2");  
setGroupLabel(3,"Group 3");
```

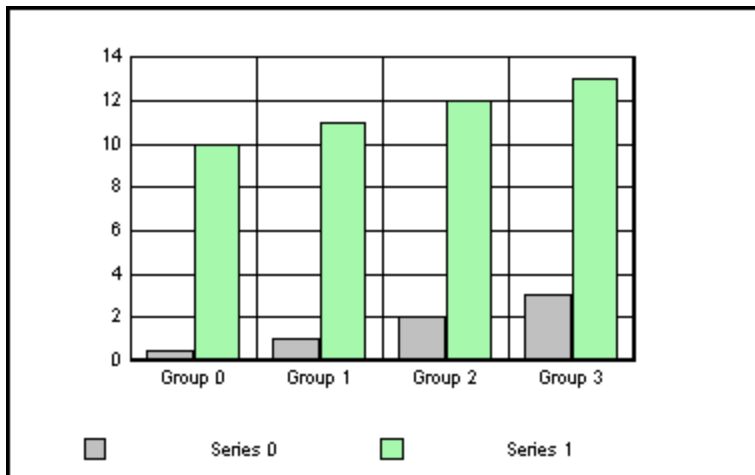


Data in a Bar/Line/Area Graph

The following sample program creates a vertical bar graph with two series and four groups.

```
setDepthRadius(0);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setData(1,0,10);
setData(1,1,11);
setData(1,2,12);
setData(1,3,13);
setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");
```

This data produces a vertical bar graph in the following format:

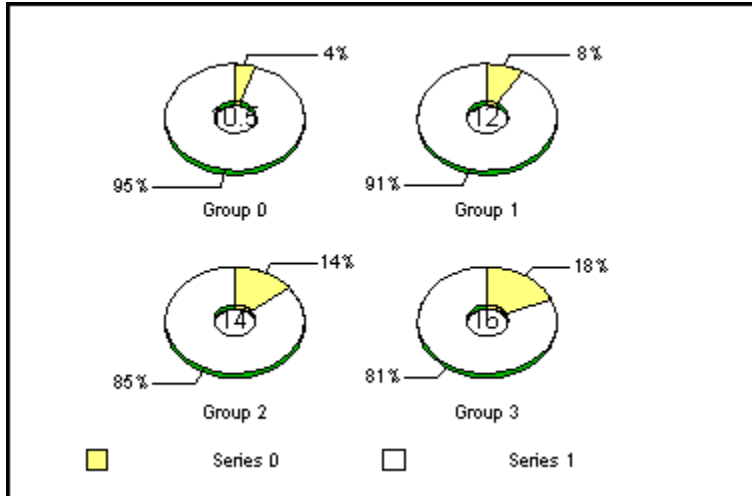


Data in a Pie Graph

The following data is the same as the sample data that was used to define the vertical bar graph in [Data in a Bar/Line/Area Graph](#) on page 197 except a multiple ring pie graph is selected. In pie graphs, each series defines a slice in the pie and each group defines a pie.

```
setGraphType(58);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setData(1,0,10);
setData(1,1,11);
setData(1,2,12);
setData(1,3,13);
setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");
```

When this data is used to define a pie graph, the graph is displayed in the following format:

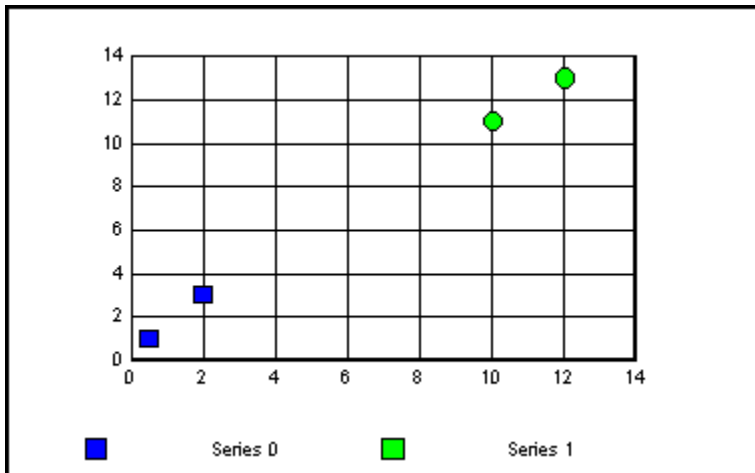


Data in an XY Scatter Graph

An XY scatter graph requires two values per marker, X and Y in that order.

```
setGraphType(61);  
setMarkerSizeDefault(60);  
setData(0,0,.5);  
setData(0,1,1);  
setData(0,2,2);  
setData(0,3,3);  
setData(1,0,10);  
setData(1,1,11);  
setData(1,2,12);  
setData(1,3,13);  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");
```

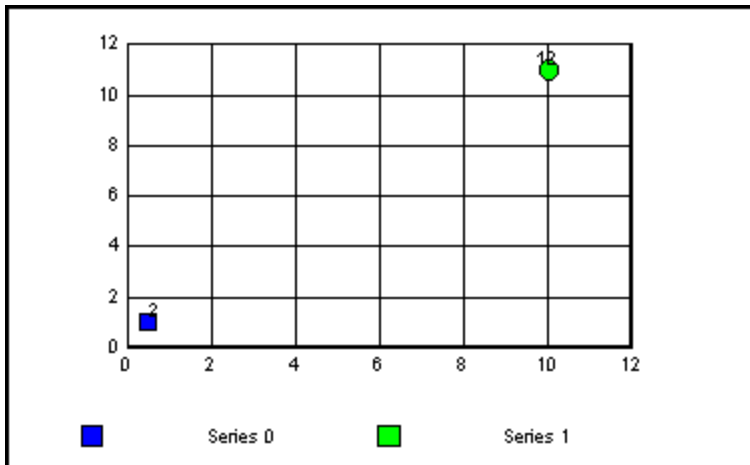
Since there are eight setData() statements in the example code, four markers are drawn in the scatter graph.



Data in an XY Scatter Graph With Labels

An XY scatter graph with labels requires three values per marker, X, Y, and the text label, in that order.

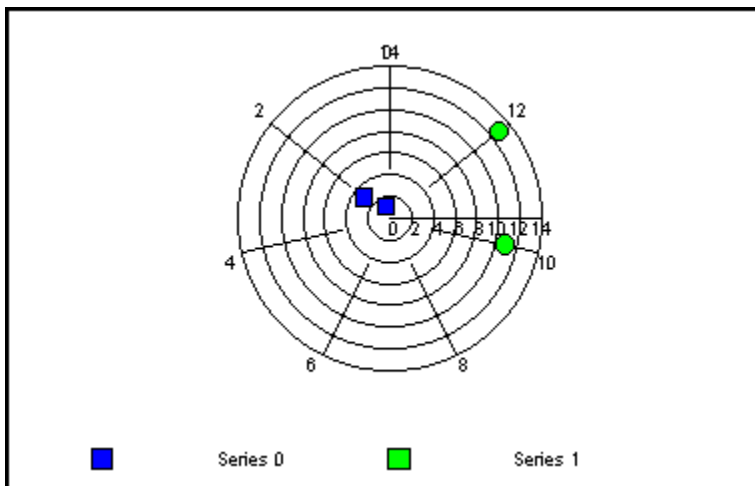
```
setGraphType(63);  
setMarkerSizeDefault(60);  
setData(0,0,.5); /* X Value */  
setData(0,1,1); /* Y Value */  
setData(0,2,2); /* Label */  
setData(1,0,10); /* X Value */  
setData(1,1,11); /* Y Value */  
setData(1,2,12); /* Label */  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");
```



Data in a Polar Graph

A polar graph requires two values per marker in the following order: X (degree) and Y (distance from center).

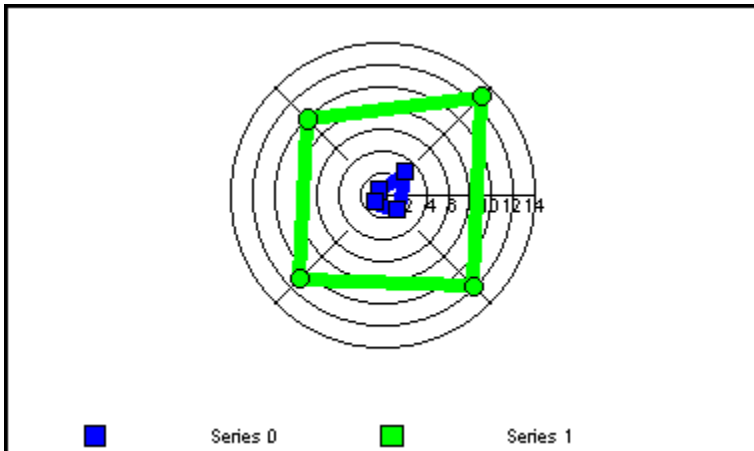
```
setGraphType(65);
setMarkerSizeDefault(60);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setData(1,0,10);
setData(1,1,11);
setData(1,2,12);
setData(1,3,13);
setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
```



Data in a Radar Graph

A radar graph requires one value per marker to be drawn in the graph.

```
setGraphType(67);  
setMarkerSizeDefault(60);  
setData(0,0,.5);  
setData(0,1,1);  
setData(0,2,2);  
setData(0,3,3);  
setData(1,0,10);  
setData(1,1,11);  
setData(1,2,12);  
setData(1,3,13);  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");
```

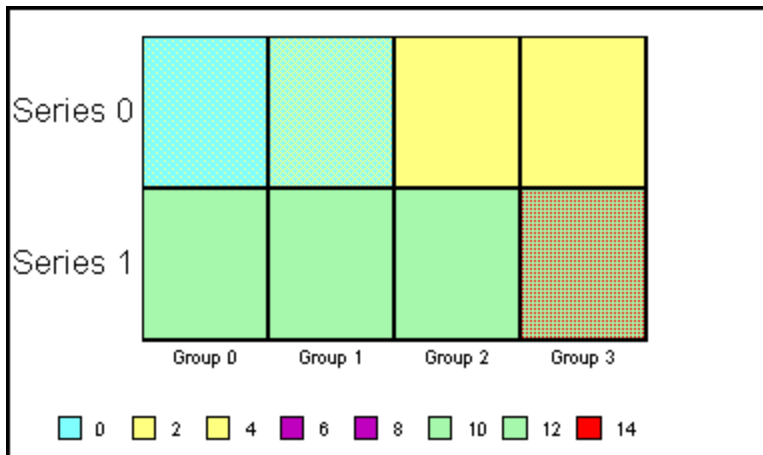


Data in a Spectral Map

A spectral map requires one value for each cell to be drawn in the map.

```
setGraphType(87);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setData(1,0,10);
setData(1,1,11);
setData(1,2,12);
setData(1,3,13);
setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");
```

When this data is used to define a spectral map, the graph is displayed in the following format:

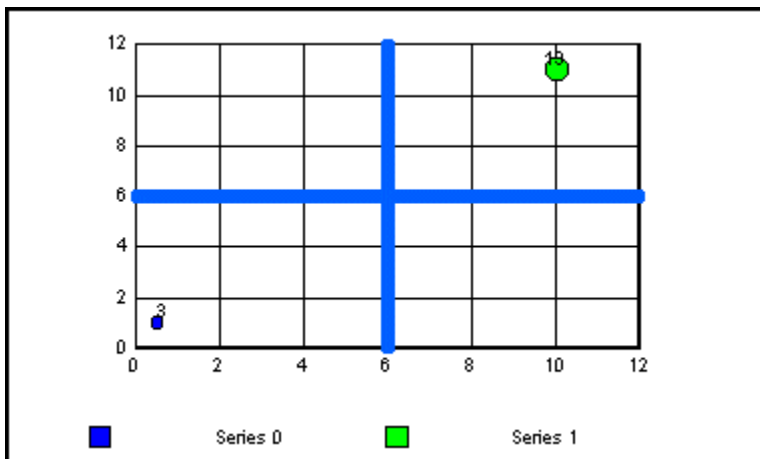


Data in a Bubble Graph

A bubble graph with labels (GraphType 92) requires four values per marker: X, Y, Z, and text label.

```
setGraphType(92);  
setMarkerSizeDefault(60);  
setData(0,0,.5); /* X Value */  
setData(0,1,1); /* Y Value */  
setData(0,2,2); /* Z Value */  
setData(0,3,3); /* Text Label */  
setData(1,0,10); /* X Value */  
setData(1,1,11); /* Y Value */  
setData(1,2,12); /* Z Value */  
setData(1,3,13); /* Text Label */  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");
```

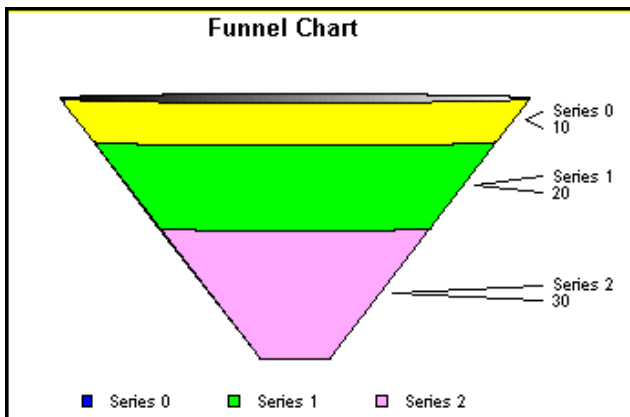
When this data is used to define a bubble graph, the graph is displayed in the following format:



Data in a Funnel Graph

A funnel graph is basically a pie graph 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. Since the funnel graph shows only one group of data at a time, the data that defines the graph should include multiple rows (series) and one column (group). If a data set includes multiple columns (groups), you can use `setDisplayFunnelGroup()` to define which group to display in the funnel graph. The following code defines a simple funnel graph with one group and three series.

```
setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleString("DisplayFunnelGroup = 0");
setTransparentBorderColor(getFrame(), true);
```



Data in a Multi-Y Graph

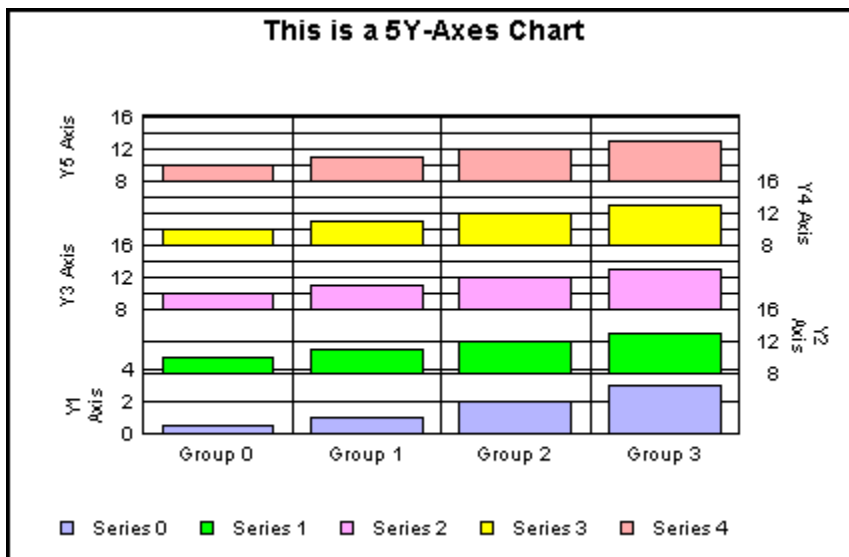
The data that is required for a multi-Y (3Y-, 4Y- or 5Y-axis) graph is the same as for any normal bar graph. By default, axis labels are staggered with odd numbered (1/3/5) axis labels on the left and even numbered (2/4) axis labels on the right. These default settings can be changed with `setAxisSide()`. The `setAxisAssignment()` method must be used to assign each series in the data set to the desired axis (Y1, Y2, Y3, Y4, Y5).

The following example creates a 5Y-axes graph with a single series assigned to each axis.

```
setDepthRadius(0);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setData(1,0,10);
setData(1,1,11);
setData(1,2,12);
setData(1,3,13);
setData(1,0,10);
setData(2,0,10);
setData(2,1,11);
setData(2,2,12);
setData(2,3,13);
setData(3,0,10);
setData(3,1,11);
setData(3,2,12);
setData(3,3,13);
setData(4,0,10);
setData(4,1,11);
setData(4,2,12);
setData(4,3,13);
setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setSeriesLabel(2,"Series 2");
setSeriesLabel(3,"Series 3");
setSeriesLabel(4,"Series 4");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");
setGraphType(105);

setAxisAssignment(1,1);
setAxisAssignment(2,2);
setAxisAssignment(3,3);
setAxisAssignment(4,4);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setAlignMultiYBars(true);
setTitleString("This is a 5Y-Axes Chart");
```

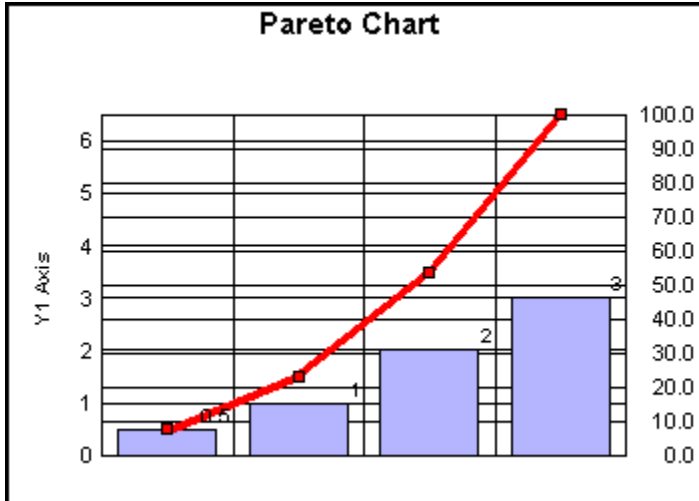
The preceding example creates the following graph output.



Data in a Pareto Graph

In a Pareto graph, the X-axis scale shows group numbers. The Y-axis scale is the percent of the total accumulation for each selected series. Each graph displays a single series. The following methods and properties create a simple Pareto graph.

```
setDepthRadius(0);
setDataTextDisplay(true);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setDataRangeToExtent();
setTitleString("Pareto Chart");
setY1TitleString("Y1 Axis");
setGraphType(102);
```



Data in a Waterfall Graph

A waterfall graph (GraphTypes 100 and 101) requires one data value for each series/group marker to be drawn in the graph. Each setData() assigns a value to a series/group that is drawn similar to a stacked bar graph. The group mode (normal, subtotal, total, or extra) determines how each series/group value is interpreted. The setWaterfallGroupMode() method assigns each group in the data set to be graphed as a normal, subtotal, total, or extra group.

Syntax:

```
void setWaterfallGroupMode(int nGroup, int nGroupMode);
```

where:

nGroup

Is a group number.

nGroupMode

Can be one of the following:

```
int WATERFALLGROUPMODE_NORMAL = 0;
int WATERFALLGROUPMODE_SUBTOTAL = 1;
int WATERFALLGROUPMODE_TOTAL = 2;
int WATERFALLGROUPMODE_EXTRA = 3;
```

For a normal group, series risers are stacked on top of each other and each successive series represents the value of its series/group plus any preceding series/group. For a subtotal or total group, the data for the group is ignored and the cumulative total up to this point will be shown for this group. Only one group in a graph can be designated as a total. An extra group is not used when performing any waterfall cumulative sums. It will appear just as stacked data. When used, it should be defined as the last group in the graph.

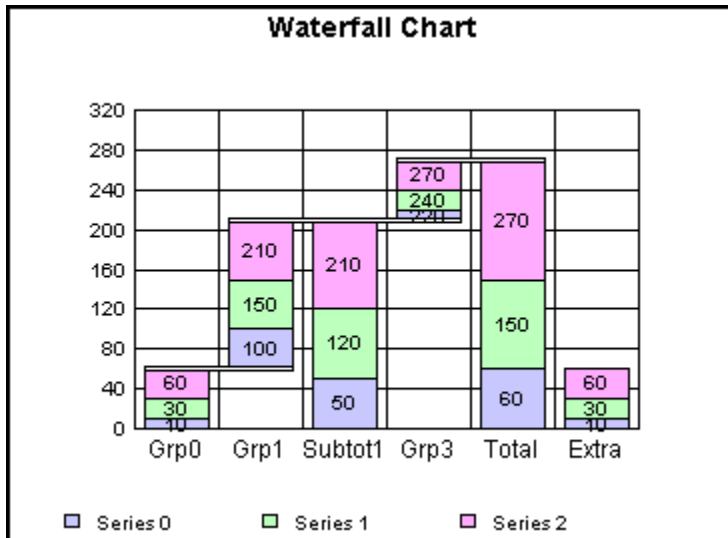
Note: The sign of each group of data can be positive or negative. Mixed signs of data (positive and negative) for a group will force the sign of all data for the group to be the first sign of the data for series zero. Zero is used for any missing data and is considered positive.

```
setGraphType(100);
setDepthRadius(0);
setDepthAngle(0);
setData(0, 0, 10);
setData(1, 0, 20);
setData(2, 0, 30);
setGroupLabel(0, "Grp0");
setData(0, 1, 40);
setData(1, 1, 50);
setData(2, 1, 60);
setGroupLabel(1, "Grp1");
```

```

setData(0, 2, 0);
setData(1, 2, 0);
setData(2, 2, 0);
setWaterfallGroupMode(2, 1);
setGroupLabel(getGroup(2), "Subtot1");
setData(0, 3, 10);
setData(1, 3, 20);
setData(2, 3, 30);
setGroupLabel(3, "Grp3");
setData(0, 4, 0);
setData(1, 4, 0);
setData(2, 4, 0);
setWaterfallGroupMode(4, 2);
setGroupLabel(getGroup(4), "Total");
setData(0, 5, 10);
setData(1, 5, 20);
setData(2, 5, 30);
setWaterfallGroupMode(5, 3);
setGroupLabel(getGroup(5), "Extra");
setStackedDataValueSum(true);
setDataTextDisplay(true);
setTextJustHoriz(getDataText(),1);
setFontSizeAbsolute(getDataText(),true);
setFontSize(getDataText(),10);
setFontSizeAbsolute(getO1Label(0),true);
setFontSize(getO1Label(0),12);
setDataRangeToExtent();

```



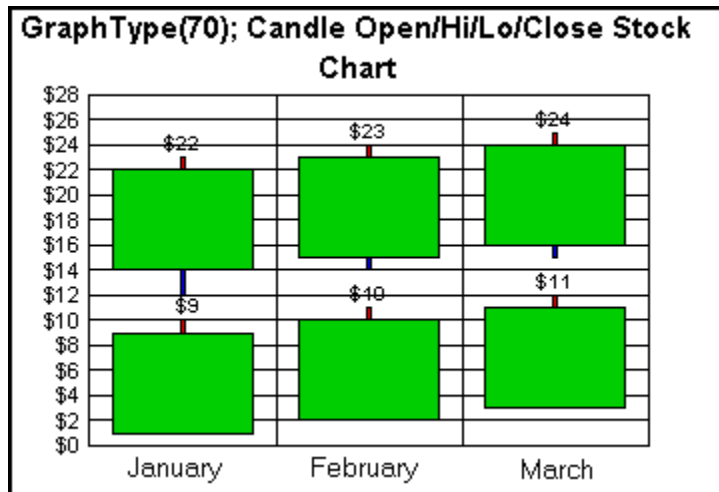
Data in a Stock Graph

The following topics describe how data is used in the stock graph types (70, 72...82, and 84).

GraphType 70

This data set can be used to produce a Candle Open/High/Low/Close stock graph (GraphType 70). This graph type requires four values per marker: Open, High, Low, and Close. The following methods were added to the data set used in [GraphTypes 79...81](#) on page 215.

```
setTitleString("GraphType(70); Candle Open/Hi/Lo/Close Stock Chart");
setGraphType(70);
setLineWidth(getStockHighLine(), 2);
setFillColor(getStockHighLine(), new Color (255,0,0));
setLineWidth(getStockLowLine(), 2);
setFillColor(getStockLowLine(), new Color(0,0,255));
```

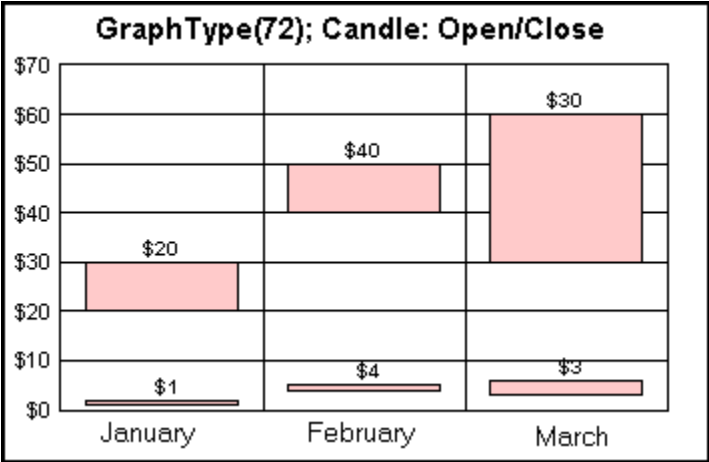


In this graph type, the size/height of each riser illustrates Open and Close values. The data text shows each Close value. The length of the red tick at the top of each riser extends the riser to the stock's High value. If a stock's Low value is less than its Open value, a tick mark will be drawn at the bottom of the riser to show how much less the low value is from the open. These tick marks are shown in blue in the top risers in the example graph above.

GraphType 72

The Candle Open/Close stock graph (GraphType 72) requires two values per marker: Open and Close.

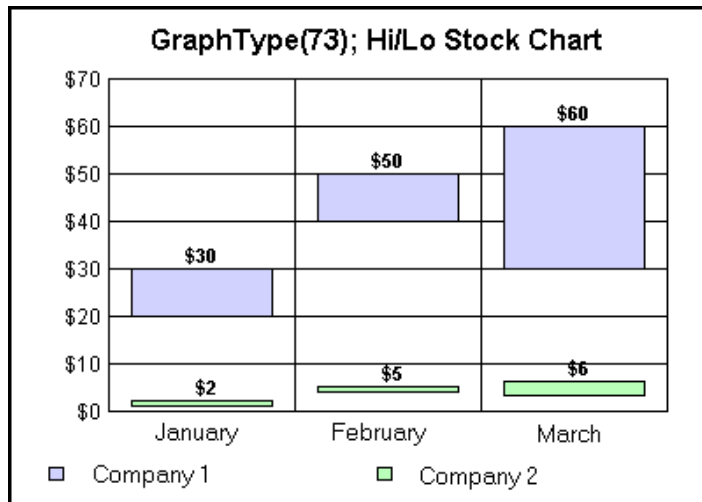
```
setData(0,0,30); // Open
setData(0,1,20); // Close
setData(0,2,50); // Open
setData(0,3,40); // Close
setData(0,4,60); // Open
setData(0,5,30); // Close
setData(1,0,2);
setData(1,1,1);
setData(1,2,5);
setData(1,3,4);
setData(1,4,6);
setData(1,5,3);
setGraphType(72);
setDataRange(0,0,1,6);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setTitleString("GraphType(72); Candle: Open/Close");
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
```



GraphTypes 73...75

For High/Low stock graphs (GraphType 73...75), the graph requires two values (high and low) for each riser to be drawn in the graph. The following example data defines two series (rows 0 and 1) and three high/low values (0...5 columns) for each series.

```
setData(0,0,30); // Company 1 - Stock 1/January High
setData(0,1,20); // Company 1 - Stock 1/January Low
setData(0,2,50); // Company 1 - Stock 1/February High
setData(0,3,40); // Company 1 - Stock 1/February Low
setData(0,4,60); // Company 1 - Stock 1/March High
setData(0,5,30); // Company 1 - Stock 1/March Low
setData(1,0,2); // Company 2 - Stock 2/January High
setData(1,1,1); // Company 2 - Stock 2/January Low
setData(1,2,5); // Company 2 - Stock 2/February High
setData(1,3,4); // Company 2 - Stock 2/February Low
setData(1,4,6); // Company 2 - Stock 2/March High
setData(1,5,3); // Company 2 - Stock 2/March Low
setGraphType(73);
setDataRange(0,0,1,6);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setTitleString("GraphType(73); Hi/Lo Stock Chart");
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
```

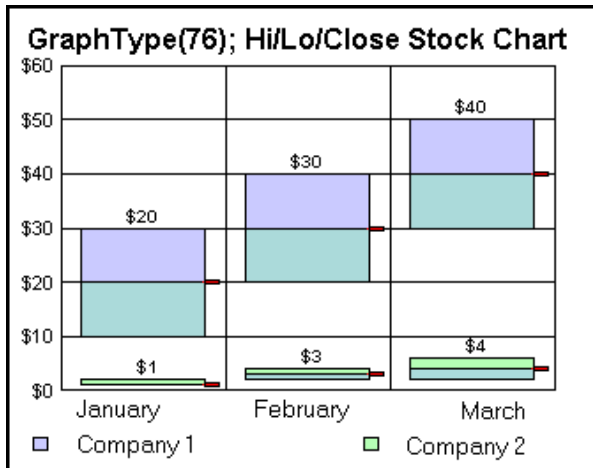


GraphTypes 76...78

The High/Low/Close stock graph requires 3 values per marker in the following order: High, Low, and Close:

```
setData(0,0,30);
setData(0,1,10);
setData(0,2,20);
setData(0,3,40);
setData(0,4,20);
setData(0,5,30);
setData(0,6,50);
setData(0,7,30);
setData(0,8,40);
setData(1,0,2);
setData(1,1,1);
setData(1,2,1);
setData(1,3,4);
setData(1,4,2);
setData(1,5,3);
setData(1,6,6);
setData(1,7,2);
setData(1,8,4);
setGraphType(76);
setDataRange(0,0,1,9);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setTitleString("GraphType(76); Hi/Lo/Close Stock Chart");
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
setFillColor(getStockCloseTick(), new Color (255,0,0));
setStockTickLength(100);
```

The preceding example creates the following graph output.



GraphTypes 79...81

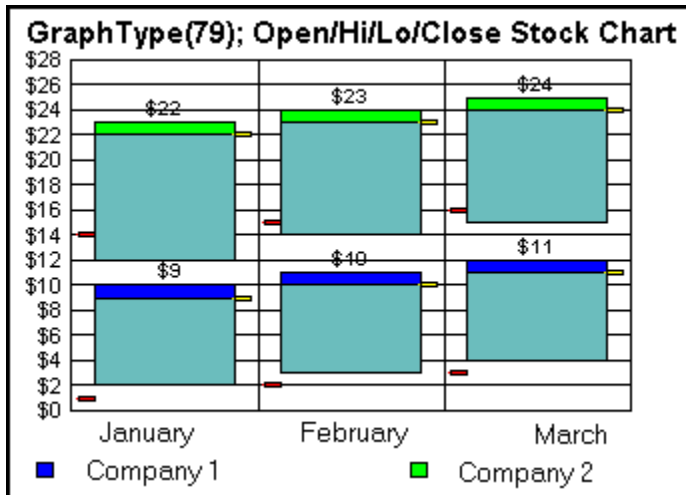
For normal stock graphs that show high, low, open, and close values (GraphType 79...81), four values are required for each data point. The four values must be supplied in the following order: Open Value, High Value, Low Value, and Close Value. The setData() methods used in the example code that follows, plot the data for three stocks/groups using the Open, High, Low, and Close values shown in the following table.

	Open	High	Low	Close
Series 1/ Stock 1 (setData(0,0...3))	1	10	2	9
Series 1/Stock 2 (setData(0,4...7))	2	11	3	10
Series 1/Stock 3 (setData(0,8...11))	3	12	4	11
Series 2/Stock 1 (setData(1,0...3))	14	23	12	22
Series 2/Stock 2 (setData(1,4...7))	15	24	14	23
Series 2/Stock 3 (setData(1,8...11))	16	25	15	24

The following example code uses setData() methods.

```
setData(0,0,1);
setData(0,1,10);
setData(0,2,2);
setData(0,3,9);
setData(0,4,2);
setData(0,5,11);
setData(0,6,3);
setData(0,7,10);
setData(0,8,3);
setData(0,9,12);
setData(0,10,4);
setData(0,11,11);
setData(1,0,14);
setData(1,1,23);
setData(1,2,12);
setData(1,3,22);
setData(1,4,15);
setData(1,5,24);
setData(1,6,14);
setData(1,7,23);
setData(1,8,16);
setData(1,9,25);
setData(1,10,15);
setData(1,11,24);
setDataRange(0,0,1,12);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
setStockTickLength(100);
setFillColor(getStockOpenTick(), new Color (255,0,0));
setFillColor(getStockCloseTick(), new Color(255,255,0));
setY1MajorGridStep(2);
setTitleString("GraphType(79); Open/Hi/Lo/Close Stock Chart");
setGraphType(79);
```

The preceding code generates the following graph output.



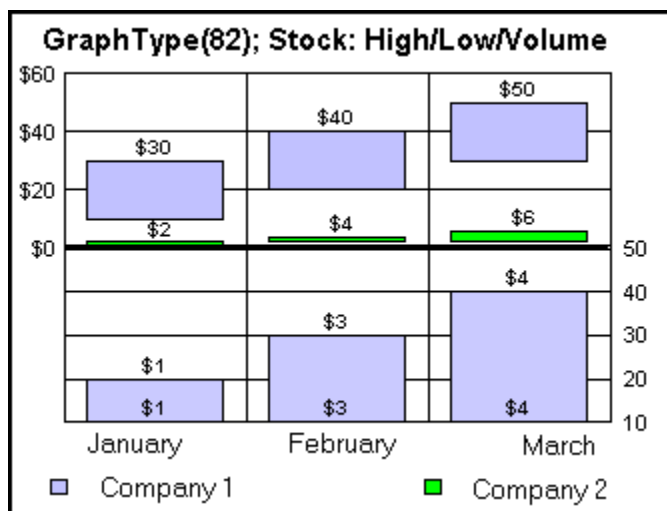
The size/height of each riser illustrates low/high values. The red tick marks identify open values. The yellow tick marks and data text show close values. The bar at the top of each riser illustrates the difference between close and high values.

GraphType 82

The stock High/Low/Volume graph requires three values per marker: High, Low, and Volume.

```
setData(0,0,30);
setData(0,1,10);
setData(0,2,20);
setData(0,3,40);
setData(0,4,20);
setData(0,5,30);
setData(0,6,50);
setData(0,7,30);
setData(0,8,40);
setData(1,0,2);
setData(1,1,1);
setData(1,2,1);
setData(1,3,4);
setData(1,4,2);
setData(1,5,3);
setData(1,6,6);
setData(1,7,2);
setData(1,8,4);
setGraphType(82);
setDataRange(0,0,1,9);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setTitleString("GraphType(82); Stock: High/Low/Volume");
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
setFillColor(getStockCloseTick(), new Color (255,0,0));
setStockTickLength(100);
```

The preceding code generates the following graph output.

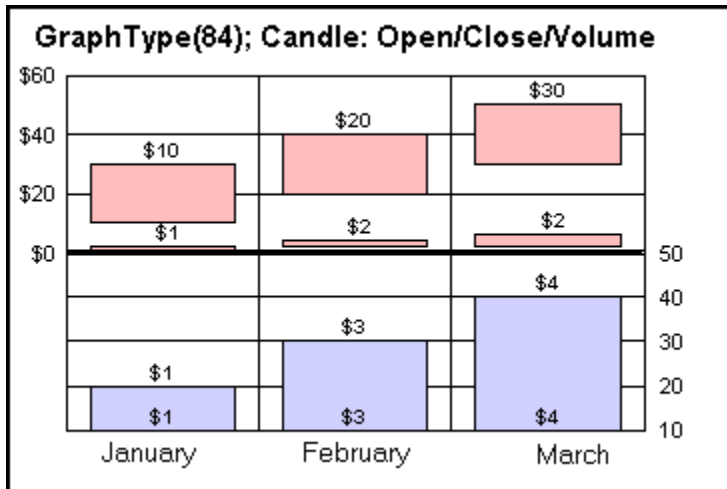


GraphType 84

The Candle Open/Close/Volume graph requires three values per marker: Open, Close, and Volume.

```
setData(0,0,30);
setData(0,1,10);
setData(0,2,20);
setData(0,3,40);
setData(0,4,20);
setData(0,5,30);
setData(0,6,50);
setData(0,7,30);
setData(0,8,40);
setData(1,0,2);
setData(1,1,1);
setData(1,2,1);
setData(1,3,4);
setData(1,4,2);
setData(1,5,3);
setData(1,6,6);
setData(1,7,2);
setData(1,8,4);
setGraphType(84);
setDataRange(0,0,1,9);
setDataTextDisplay(true);
setDataTextFormat(6);
setY1LabelFormat(6);
setTitleString("GraphType(84); Candle: Open/Close/Volume");
setLegendDisplay(true);
setSeriesLabel(0, "Company 1");
setSeriesLabel(1, "Company 2");
setGroupLabel(0, "January");
setGroupLabel(1, "February");
setGroupLabel(2, "March");
setFillColor(getStockCloseTick(), new Color (255,0,0));
setStockTickLength(100);
```


The preceding code generates the following graph output.



Time Scale Axis Graphs

A time scale axis uses time to display data. Clustered Bar, Stacked Bar, Line, and Scatter graphs may use a time scale axis. The following time intervals can be specified in each of these graph types:

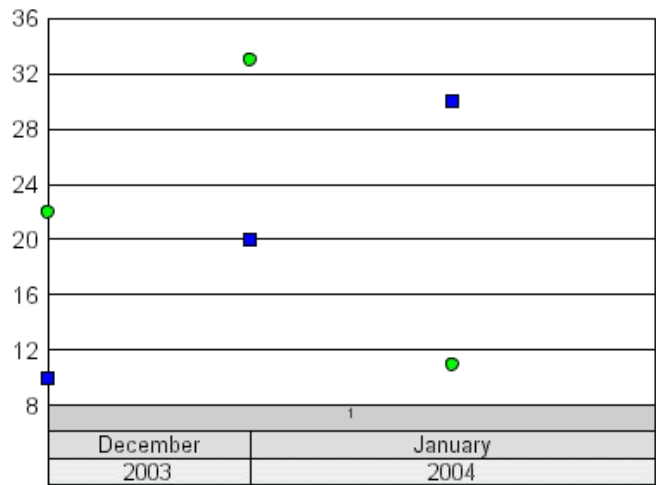
```

TIME_INTERVAL_UNDEFINED = 0
TIME_INTERVAL_SECONDS = 1
TIME_INTERVAL_MINUTES = 2
TIME_INTERVAL_HOURS = 3
TIME_INTERVAL_DAYS = 4
TIME_INTERVAL_WEEKS = 5
TIME_INTERVAL_MONTHS = 6
TIME_INTERVAL_QUARTERS = 7
TIME_INTERVAL_YEARS = 8

```

It is often useful to represent data with several intervals shown simultaneously on a time axis scale.

The following example shows three intervals, which are weeks, months, and years.



The level intervals must be set in decreasing order – the level zero interval must be less than level one, level one must be less than level two. In this example, the following methods were used to set the level intervals:

```
setLevelInterval(0, Interval.TIME_INTERVAL_WEEKS);
setLevelInterval(1, Interval.TIME_INTERVAL_MONTHS);
setLevelInterval(2, Interval.TIME_INTERVAL_YEARS);
```

Each level can be formatted independently using properties and methods you can apply to any other label object (for example, font size, font color, word wrap, anti-alias, and so on). The following properties and methods handle functionality specific to the Time Scale Axis:

- ❑ UseTimeScaleAxis; Property to enable/disable time scale axis.
- ❑ get/setLevelDateFormat(); Get/set the date format of a level.
- ❑ get/setLevelFillColor(); Get/set the fill color of a level.
- ❑ get/setLevelHeight(); Get/set the height of a level in virtual coordinates.
- ❑ get/setLevelInterval(); Get/set the time interval of a level.
- ❑ setLevelBorderColor(); Set the border color of a level.
- ❑ setLevelTextColor(); Set the text color of a level.
- ❑ get/setLevelTextFormat(); Get/set the text format of a level.

The `setLevelTextFormat()` method defines how time axis scale labels are drawn.

```
IIntervalScale.FORMAT_SHORT = 0
IIntervalScale.FORMAT_MEDIUM = 1
IIntervalScale.FORMAT_LONG = 2
IIntervalScale.FORMAT_RAW = 3
```

The format in which text is drawn depends on the level interval set by `setLevelInterval()`. The following table shows how each text format is represented for each level:

Interval Level	FORMAT_SHORT	FORMAT_MEDIUM	FORMAT_LONG
TIME_INTERVAL_SECONDS	1	Sec 1	Second 1
TIME_INTERVAL_MINUTES	1	Min 1	Minute 1
TIME_INTERVAL_HOURS	1	Hour 1	Hour 1
TIME_INTERVAL_DAYS	1	Day 1	Day 1
TIME_INTERVAL_WEEKS	1	Week 1	Week 1
TIME_INTERVAL_MONTHS	1	Mon 1	Month 1
TIME_INTERVAL_QUARTERS	1	Q 1	Quarter 1
TIME_INTERVAL_YEARS	1	01	2001

If `setLevelTextFormat()` selects `FORMAT_RAW`, you can specifically define how date and time text is formatted using `setLevelDateFormat()` and standard Java `SimpleDateFormat` codes, which are listed and described in the following table.

Code	Code Description
G	Era Designation (for example, AD)
y	Year (for example, 2004, 04)
M	Month in year (e.g, March, Mar., 03)
w	Week in year (for example, 1...52)
W	Week in month (for example, 2)

Code	Code Description
D	Day in year (for example, 1...365)
d	Day in month (for example, 10)
F	Day of week in month (for example, 2)
E	Day in week (for example, Tuesday, Tue)
a	AM/PM Marker (for example, PM)
H	Zero-Based Hour in day (for example, 0...23)
k	Hour in day (for example, 1...24)
K	Zero-Based Hour in AM/PM (for example, 0...11)
h	Hour in AM/PM (for example, 1...11)
m	Minute in hour (for example, 30)
s	Second in minute (for example, 55)
S	Millisecond (for example, 978)
z	General Time Zone (for example, Pacific Standard Time, PST, GMT-08:00)
Z	RFC 822 Time Zone (for example, -0800)

Examples:

If the date/time is set to July 4, 2001 12:08:56 local time in the United States Pacific Time zone, the following is an example of the necessary code:

```
setLevelDateFormat(FORMAT_RAW);
setLevelDateFormat("yyyy.MM.dd G 'at' HH:mm:ss z");
```

2001.07.04 AD at 12:08:56 PDT

```
setLevelDateFormat(FORMAT_RAW);
setLevelDateFormat("EEE, MMM d, 'yy");
```

Wed, Jul 4, '01

```
setLevelTextFormat(FORMAT_RAW);
setLevelDateFormat("hh 'o'clock' a, zzzz");
```

12 o'clock PM, Pacific Daylight Time

```
setLevelTextFormat(FORMAT_RAW);
setLevelDateFormat("yyyyy.MMMMM.dd GGG hh:mm aaa");
```

02001.July.04 AD 12:08 PM

The following methods must be used to set the starting and ending date boundaries of the graph:

- ❑ `setStartDate();`
- ❑ `setEndDate();`

The following properties can be used to define major and minor grid ticks on a time axis scale.

- ❑ `MajorGridTickInterval`
- ❑ `MinorGridTickInterval`

You must use the `setData()` methods to define dates/times and data to be graphed. Each data point to be graphed requires two values: a date/time specification and a value.

Example:

```
restoreDefaults();
setUseTimeScaleAxis(true);
clearDataLabels();
clearGroupLabels();
clearSeriesLabels();
setLegendDisplay(false);
setDepthRadius(0);
clearDataStorage();
setData(0,0,12,28,2003); // date
setData(0,1,10.0);      // value
setData(0,2,12,29,2003);
setData(0,3,20.0);
setData(0,4,12,30,2003);
setData(0,5,30.0);
setData(0,6,12,31,2003);
setData(0,7,25.0);
setData(0,8,01,01,2004);
setData(0,9,20.0);
setData(0,10,01,2,2004);
setData(0,11,25.0);
setData(0,12,01,3,2004);
setData(0,13,22.0);
setData(0,14,01,4,2004);
setData(0,15,23.0);
setData(1,0,12,28,2003);
setData(1,1,12.0);
setData(1,2,12,29,2003);
setData(1,3,12.0);
setData(1,4,12,30,2003);
setData(1,5,21.0);
setData(1,6,12,31,2003);
setData(1,7,15.0);
setData(1,8,01,01,2004);
setData(1,9,23.0);
setData(1,10,1,2,2004);

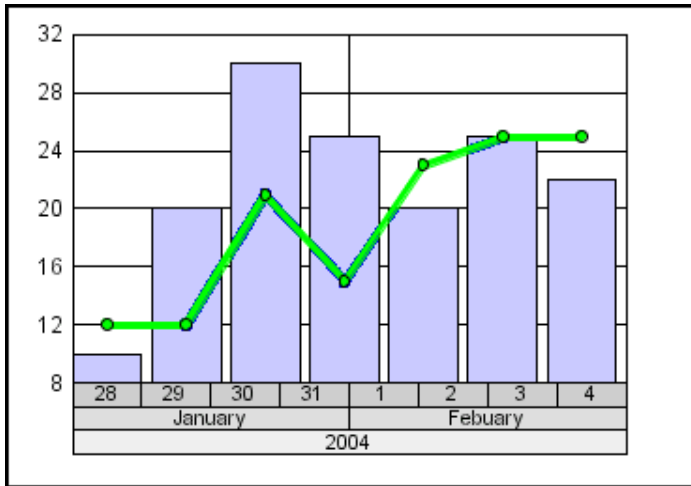
setData(1,11,25.0);
setData(1,12,1,3,2004);
setData(1,13,25.0);
setData(1,14,1,4,2004);

setStartDate(12,28,2003);
setEndDate(1,4,2004);
setDataRangeToExtent();
setGraphType(17);
setSeriesType(getSeries(0), 1);
setSeriesType(getSeries(1), 2);
```

```

setUseTimeScaleAxis(true);
setLevelInterval(0,4);
setLevelInterval(1,6);
setLevelInterval(2,8);
setLevelTextFormat(0,0);
setLevelTextFormat(1,2);
setLevelTextFormat(2,2);

```



The `DateUtils.getDateTime()` method is optional and can be used to return a date as a Java Calendar Object.

Example:

```

Calendar start = DateUtils.getDateTime ();
Calendar end = DateUtils.getDateTime ();
setStartDate(start);
setEndDate(end);

```

`TimeScaleAxis.setTimeScaleAxisDefaults()` is an optional method that can be called to do clean up.

Properties

The following topics describe the properties you can set for each graph type. The properties are listed in alphabetical order.

Properties for Graphs

The following properties are available for graphs:

A

AdjustAutoScaleForRefLines

AltFmtFrameNumColors

Autoshading

AlignMultiYBars

AltFmtMatchAxisLabelColor

AutoshadingWalls

AltFmtFrameSeparator

AutoHideDataTextOnSmallRisers

B

BiDirectional

BoxPlotType

C

<i>Cascade</i>	<i>CubePanX</i>	<i>CubeViewerX</i>
<i>ColorMode</i>	<i>CubePanY</i>	<i>CubeViewerY</i>
<i>ConnectLineMarkers</i>	<i>CubeProportional</i>	<i>CubeViewerZ</i>
<i>ConnectParetoBars</i>	<i>CubeRiserInterpolation</i>	<i>CubeWallThickX</i>
<i>ConnectScatterMarkers</i>	<i>CubeSizeX</i>	<i>CubeWallThickY</i>
<i>CubeFocusFactor</i>	<i>CubeSizeY</i>	<i>CubeWallThickZ</i>
<i>CubeIsometricProjection</i>	<i>CubeSizeZ</i>	<i>CubeZoomFactor</i>
<i>CubeLightSourceX</i>	<i>CubeSquareRisers</i>	<i>CurveFitEquationDisplay</i>
<i>CubeLightSourceY</i>	<i>CubeTranslationX</i>	<i>CurveFitHighOrderFirst</i>
<i>CubeLightSourceZ</i>	<i>CubeTranslationY</i>	<i>CurveFitNumSegments</i>
	<i>CubeTranslationZ</i>	

D

<i>DataItemsAlongSeries</i>	<i>DataTextPosition</i>	<i>Display3DRightWall</i>
<i>DataLineThickness</i>	<i>DataTextRadiusDefault</i>	<i>DisplayBarAsPictograph</i>
<i>DataScrollerPresenceGroups</i>	<i>DataTextStackedTotalOnTop()</i>	<i>DisplayFunnelGroup</i>
<i>DataScrollerPresenceSeries</i>	<i>DataTextTemplateBubble</i>	<i>DisplayParetoSeries</i>
<i>DataSortType</i>	<i>DataTextTemplateHiLo</i>	<i>DocumentHeightInTwips</i>
<i>DataTextAngleDefault</i>	<i>DataTextTemplateScatter</i>	<i>DocumentWidthInTwips</i>
<i>DataTextDisplay</i>	<i>DepthAngle</i>	<i>DrawAreaMarkers</i>
<i>DataTextFormat</i>	<i>DepthRadius</i>	<i>DrawCompleteAreaRiser</i>
<i>DataTextFormatPattern</i>	<i>Display3DFloor</i>	<i>DualAxisLineDisplay</i>
	<i>Display3DLeftWall</i>	<i>DualAxisSplitPosition</i>

E*ExactColorByHeight***F***FillMissingData**FrameAutoShade**FootnoteDisplay**FrameDisplay**FootnoteString**FunnelLabelValueLocation**ForceSeriesAbsolute***G***GaugeBackgroundStyle**GaugeRelativeInnerRadius**Grid3DLeftWallDisplayY**GaugeColorCenterByQuality**GaugeRelativeThickness**Grid3DLeftWallDisplayZ**GaugeDescending**GaugeSizeMode**Grid3DRightWallDisplayX**GaugeIgnoreBackgroundStyle**GaugeStartAngle**Grid3DRightWallDisplayY**GaugeMajorAutoStep**GaugeStopAngle**Grid3DRiserDisplayX**GaugeMajorTickRelativeTickLength* *GraphType**Grid3DRiserDisplayY**GaugeMinorAutoStep**Grid3DFloorDisplayX**Grid3DRiserDisplayZ**GaugeMinorTickRelativeTickLength* *Grid3DFloorDisplayZ**GaugeNeedleStyle***H***HiliteBarResize*

I

IgnoreTickSkip

InterpretAsHLOC

L

*LegendAutomatic**LegendMarkersPerRow**LegendScroll*

*LegendDisplay**LegendOrient**LegendSeriesCount*

*LegendMarkerPosition**LegendReverse**LegendSeriesStart*

M

*MajorGridDrawEveryCount**MarkerDisplay*

*MajorGridTickInterval**MarkerSizeDefault*

*MinorGridTickInterval**MoveAxisTitles*

N

*NestedLabels**NumMissingDataSegments*

*NullLegendMarkerShapesAsSquares**NumPieArcSegmentsPerPixel*

O

*OtherPercentage**OutputSVG*

*OtherSeries**OverDrawSkip*

O1

<i>O1AxisLineDisplay</i>	<i>O1LabelRotate</i>	<i>O1MinorGridCount</i>
<i>O1AxisSide</i>	<i>O1LabelSkipBegin</i>	<i>O1MinorGridDisplay</i>
<i>O1ExcludeMaxLabel</i>	<i>O1LabelSkipCount</i>	<i>O1MinorGridStyle</i>
<i>O1ExcludeMinLabel</i>	<i>O1LabelStagger</i>	<i>O1MinorTickCount</i>
<i>O1LabelAutoAngle</i>	<i>O1LabelWrap</i>	<i>O1MinorTickDisplay</i>
<i>O1LabelAutoSkip</i>	<i>O1MajorGridDisplay</i>	<i>O1MinorTickStyle</i>
<i>O1LabelDisplay</i>	<i>O1MajorGridStyle</i>	<i>O1TitleDisplay</i>
<i>O1LabelMargin</i>	<i>O1MajorTickDisplay</i>	<i>O1TitleString</i>
	<i>O1MajorTickStyle</i>	

O2

<i>O2ExcludeMaxLabel</i>	<i>O2LabelDisplay</i>	<i>O2LabelWrap</i>
<i>O2ExcludeMinLabel</i>	<i>O2LabelRotate</i>	<i>O2TitleDisplay</i>
<i>O2LabelAutoSkip</i>	<i>O2LabelSkipBegin</i>	<i>O2TitleString</i>
	<i>O2LabelSkipCount</i>	

P

<i>ParetoDisplayThreshold</i>	<i>PieFeelerTextDisplay</i>	<i>PieRingTotalFormatPattern</i>
<i>ParetoY2MajorGridStep</i>	<i>PieFeelerTextFormat</i>	<i>PieRotate</i>
<i>ParetoY2MajorTickStep</i>	<i>PieFeelerTextFormatPattern</i>	<i>PieSorting</i>
<i>PieBarLabelDisplay</i>	<i>PieLabelDisplay</i>	<i>PiesPerRow</i>
<i>PieBarLabelString</i>	<i>PieRingSize</i>	<i>PieTilt</i>
<i>PieBarSeries</i>	<i>PieRingTotalDisplay</i>	<i>Place</i>
<i>PieDepth</i>	<i>PieRingTotalFormat</i>	<i>PrimitiveAntialiasing</i>

Q

<i>QuadrantLineCountX</i>	<i>QuadrantLineCountY</i>

R

<i>ReferencePercentageMode</i>	<i>ReshapeEnable</i>
<i>ReferenceTextPosition</i>	<i>ResizeBarMode</i>
<i>RemoveDuplicateDataText</i>	<i>ReverseGroups</i>
<i>RepositionDataTextCollision</i>	<i>ReverseSeries</i>
<i>RepositionStackedDataTextCollision</i>	<i>Riser3DThicknessY</i>
	<i>RiserBarGroupSpacing</i>
	<i>RiserBorderMode</i>
	<i>RiserWidth</i>

S

<i>ScaleFromZero</i>	<i>SeriesLooping</i>	<i>Stock52WeekLowValue</i>
<i>ScrollOffsetGroup</i>	<i>ShadowXOffsetDefault</i>	<i>StockCandleForVolumeColor</i>
<i>ScrollOffsetSeries</i>	<i>ShadowYOffsetDefault</i>	<i>StockCloseSplitDisplay</i>
<i>SelectionEnable</i>	<i>SmoothLines</i>	<i>StockCloseTicksDisplay</i>
<i>SelectionEnableMove</i>	<i>SquareMarkers</i>	<i>StockMovingAverageDisplay</i>
<i>SeriesAreRows</i>	<i>StackedDataValueSum</i>	<i>StockOpenTicksDisplay</i>
<i>SeriesDefaultBorderColor</i>	<i>Stock52WeekHighDisplay</i>	<i>StockTickLength</i>
<i>SeriesDefaultTransparentBorderColor</i>	<i>Stock52WeekHighValue</i>	<i>SubtitleDisplay</i>
<i>SeriesLineWidthDefault</i>	<i>Stock52WeekLowDisplay</i>	<i>SubtitleString</i>

T[*TextAntialiasing*](#)[*ToolTipDelay*](#)[*ToolTipTimeoutEnabled*](#)[*TitleDisplay*](#)[*ToolTipDisplay*](#)[*TransparentMarkers*](#)[*TitleString*](#)[*ToolTipMode*](#)[*TransparentMarkerSize*](#)**U**[*UseDefaultBubbleMarker*](#)[*UseSeriesShapes*](#)[*UseOffScreen*](#)[*UseTimeScaleAxis*](#)[*UseSeriesBorderDefaults*](#)**V**[*ViewableGroups*](#)[*ViewableSeries*](#)[*ViewableGroupsStock*](#)[*ViewableSeriesStock*](#)[*Viewing3DAnglePreset*](#)**W**[*WaterfallLastGroupTotal*](#)[*WaterfallStackColorMode*](#)

X1

<i>X1AxisDescending</i>	<i>X1MajorGridDisplay</i>	<i>X1MinorTickStyle</i>
<i>X1AxisLineDisplay</i>	<i>X1MajorGridStep</i>	<i>X1MustIncludeZero</i>
<i>X1AxisSide</i>	<i>X1MajorGridStepAuto</i>	<i>X1OffScaleDisplay</i>
<i>X1ExcludeMaxLabel</i>	<i>X1MajorGridStyle</i>	<i>X1ScaleMax</i>
<i>X1ExcludeMinLabel</i>	<i>X1MajorTickDisplay</i>	<i>X1ScaleMaxAuto</i>
<i>X1LabelDisplay</i>	<i>X1MajorTickStyle</i>	<i>X1ScaleMin</i>
<i>X1LabelFormat</i>	<i>X1MinorGridDisplay</i>	<i>X1ScaleMinAuto</i>
<i>X1LabelFormatPattern</i>	<i>X1MinorGridStep</i>	<i>X1TitleDisplay</i>
<i>X1LabelRotate</i>	<i>X1MinorGridStepAuto</i>	<i>X1TitleString</i>
<i>X1LabelStagger</i>	<i>X1MinorGridStyle</i>	<i>X1ZeroLineDisplay</i>
<i>X1LogScale</i>	<i>X1MinorTickDisplay</i>	

Y

<i>Y1AxisDescending</i>	<i>Y1MajorGridStep</i>	<i>Y1MinorTickStepAuto</i>
<i>Y1AxisLineDisplay</i>	<i>Y1MajorGridStepAuto</i>	<i>Y1MinorTickStyle</i>
<i>Y1AxisSide</i>	<i>Y1MajorGridStyle</i>	<i>Y1MustIncludeZero</i>
<i>Y1ExcludeMaxLabel</i>	<i>Y1MajorTickDisplay</i>	<i>Y1OffScaleDisplay</i>
<i>Y1ExcludeMinLabel</i>	<i>Y1MajorTickStyle</i>	<i>Y1ScaleMax</i>
<i>Y1LabelDisplay</i>	<i>Y1MinorGridDisplay</i>	<i>Y1ScaleMaxAuto</i>
<i>Y1LabelFormat</i>	<i>Y1MinorGridStepAuto</i>	<i>Y1ScaleMin</i>
<i>Y1LabelFormatPattern</i>	<i>Y1MinorGridStyle</i>	<i>Y1ScaleMinAuto</i>
<i>Y1LabelRotate</i>	<i>Y1MinorTickDisplay</i>	<i>Y1TitleDisplay</i>
<i>Y1LabelStagger</i>		<i>Y1TitleString</i>
<i>Y1LogScale</i>		<i>Y1ZeroLineDisplay</i>
<i>Y1MajorGridDisplay</i>		

Y2

<i>Y2AxisDescending</i>	<i>Y2MajorGridStep</i>	<i>Y2MustIncludeZero</i>
<i>Y2AxisLineDisplay</i>	<i>Y2MajorGridStepAuto</i>	<i>Y2OffScaleDisplay</i>
<i>Y2AxisSide</i>	<i>Y2MajorGridStyle</i>	<i>Y2ScaleMax</i>
<i>Y2ExcludeMaxLabel</i>	<i>Y2MinorGridDisplay</i>	<i>Y2ScaleMaxAuto</i>
<i>Y2ExcludeMinLabel</i>	<i>Y2MinorGridStep</i>	<i>Y2ScaleMin</i>
<i>Y2LabelDisplay</i>	<i>Y2MinorGridStepAuto</i>	<i>Y2ScaleMinAuto</i>
<i>Y2LabelFormat</i>	<i>Y2MinorGridStyle</i>	<i>Y2TitleDisplay</i>
<i>Y2LabelFormatPattern</i>		<i>Y2TitleString</i>
<i>Y2LabelRotate</i>		<i>Y2ZeroLineDisplay</i>
<i>Y2LabelStagger</i>		
<i>Y2LogScale</i>		
<i>Y2MajorGridDisplay</i>		

Y3, Y4, Y5

<i>Y3AxisDescending</i>	<i>Y4AxisDescending</i>	<i>Y5AxisDescending</i>
<i>Y3LogScale</i>	<i>Y4LogScale</i>	<i>Y5LogScale</i>
<i>Y3ZeroLineDisplay</i>	<i>Y4ZeroLineDisplay</i>	<i>Y5ZeroLineDisplay</i>

Z

<i>ZeroValueDataTextDisplay</i>	<i>ZoomDirection</i>
<i>ZeroValueDataTextStackedTotalOnTopDisplay</i>	<i>ZoomPercentage</i>

AdjustAutoScaleForRefLines

When set to true, this property assures that the maximum or minimum Axis scale value will be sufficient to show the reference line. If the current scale is not sufficient, the maximum or minimum value is automatically adjusted to ensure that the reference line displays on the graph. False is the default value.

Syntax:

```
setAdjustAutoScaleForRefLines(boolean);
```

```
boolean = getAdjustAutoScaleForRefLines();
```

where:

boolean

Can be one of the following:

[true](#) enables automatic adjustment of the scale if necessary to display the reference line on the graph output.

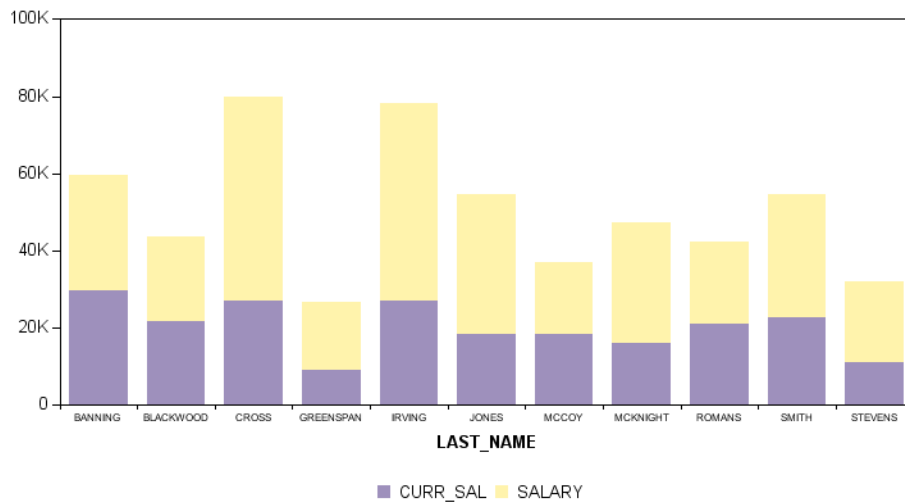
Note: Automatic scaling for the maximum and/or minimum scale values must be enabled in order for automatic scale adjustment to work. See [getScaleMaxAuto\(\)](#) on page 804 and [getScaleMinAuto\(\)](#) on page 806.

[false](#) disables automatic adjustment of the scale. This value is the default.

Example:

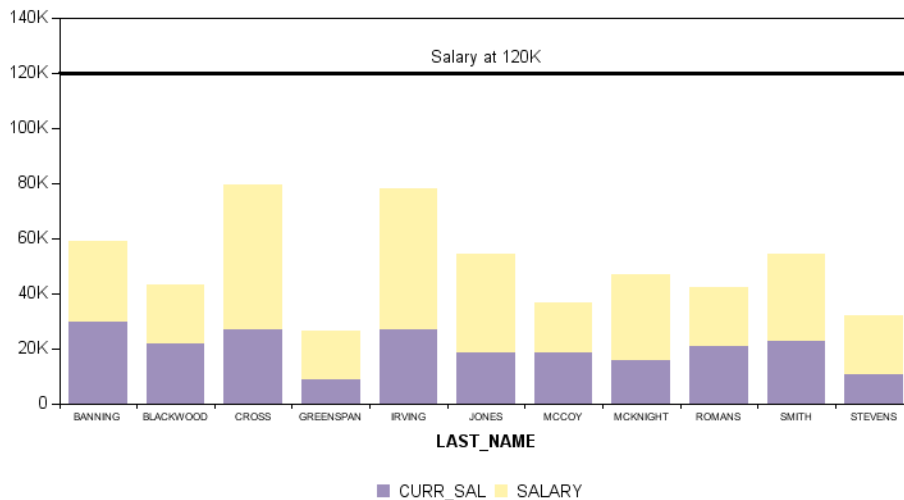
With `setAdjustAutoScaleForRefLines(off)`, the reference line does not display on the graph output:

```
setAdjustAutoScaleForRefLines(false);  
setDisplay(getReferenceLineY1(0), true);  
setLineWidth(getReferenceLineY1(0), 2);  
setReferenceLineValueY1(0, 120000.0);  
setTextString(getReferenceLineTextY1(0), "Salary at 120K");  
setDisplay(getReferenceLineTextY1(0), true);
```



With `setAdjustAutoScaleForRefLines(on)`, the graph scale adjusts so the reference line displays on the graph output:

```
setAdjustAutoScaleForRefLines(true);
setDisplay(getReferenceLineY1(0), true);
setLineWidth(getReferenceLineY1(0), 2);
setReferenceLineValueY1(0, 120000.0);
setTextString(getReferenceLineTextY1(0), "Salary at 120K");
setDisplay(getReferenceLineTextY1(0), true);
```



See [Reference Lines](#) on page 183, [getReferenceLineY1\(\)](#) on page 1118.

AlignMultiYBars

In Y3-axis, Y4-axis, and Y5-axis graphs, this property enables or disables the alignment of the bar risers.

Syntax:

```
setAlignMultiYBars(boolean);
boolean = getAlignMultiYBars();
```

where:

boolean

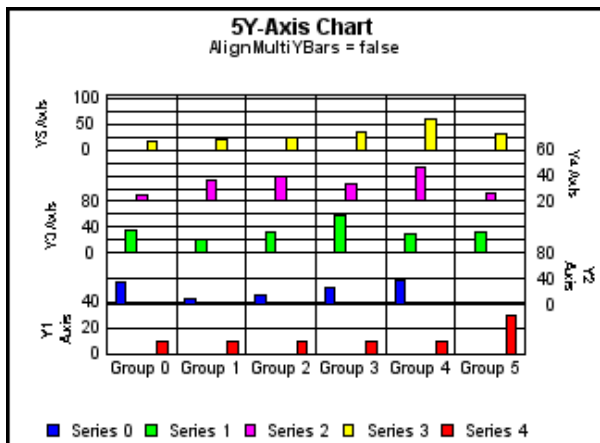
Can be one of the following:

true enables bar alignment.

`false` disables bar alignment. This value is the default.

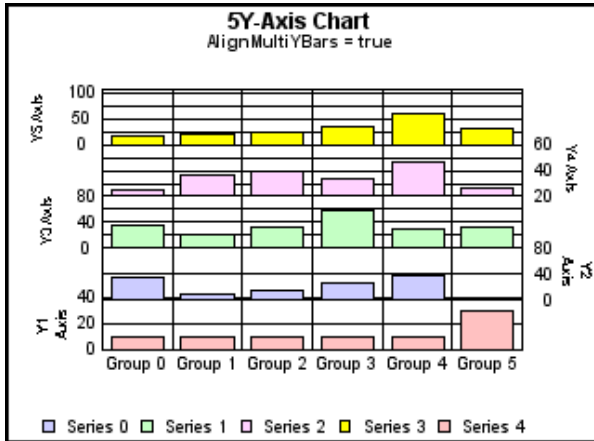
Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setSubtitleString("AlignMultiYBars = false"); setAlignMultiYBars(false);
```



```
setSubtitleString("AlignMultiYBars = true");
```

```
setAlignMultiYBars(true);
```



AltFmtFrameSeparator

Alternate formatting objects and methods can be used to apply different colors to sections of an axis and axis labels. You may also define a text label to show with an alternate formatted section.

These methods get/set the location on an axis where alternate formatting is applied.

Syntax:

```
double getAltFmtFrameSeparator (IdentObj idObj);
```

```
void setAltFmtFrameSeparator(IdentObj idObj,double newValue);
```

where:

idObj

Is the object ID of an alternate formatting axis object (for example, getY1AltFmtFrame(), getO1AltFmtFrame(), and others).

newValue

For setAltFmtFrameSeparator(), the location on the axis where alternate formatting will be applied. For a numeric axis, specify a value that is within the range of values on the specified axis. For the ordinal (O1) axis, specify a group number that is within the range of groups on the ordinal axis.

Return:

double

For getAltFmtFrameSeparator(), the axis location where alternate formatting is applied.

The following objects support alternate formatting of an axis and axis labels.

- ❑ IdentObj getY1AltFmtFrame()
- ❑ IdentObj getY1AltFmtFrame(int nIndex)
- ❑ IdentObj getY2AltFmtFrame()
- ❑ IdentObj getY2AltFmtFrame(int nIndex)
- ❑ IdentObj getY3AltFmtFrame()
- ❑ IdentObj getY3AltFmtFrame(int nIndex)
- ❑ IdentObj getY4AltFmtFrame()
- ❑ IdentObj getY4AltFmtFrame(int nIndex)
- ❑ IdentObj getY5AltFmtFrame()
- ❑ IdentObj getY5AltFmtFrame(int nIndex)
- ❑ IdentObj getO1AltFmtFrame()
- ❑ IdentObj getO1AltFmtFrame(int nIndex)
- ❑ IdentObj getX1AltFmtFrame()
- ❑ IdentObj getX1AltFmtFrame(int nIndex)
- ❑ IdentObj getY1AltFmtLabel()
- ❑ IdentObj getY1AltFmtLabel(int nIndex)
- ❑ IdentObj getY2AltFmtLabel()
- ❑ IdentObj getY2AltFmtLabel(int nIndex)
- ❑ IdentObj getY3AltFmtLabel()
- ❑ IdentObj getY3AltFmtLabel(int nIndex)
- ❑ IdentObj getY4AltFmtLabel()
- ❑ IdentObj getY4AltFmtLabel(int nIndex)
- ❑ IdentObj getY5AltFmtLabel()
- ❑ IdentObj getY5AltFmtLabel(int nIndex)
- ❑ IdentObj getO1AltFmtLabel()

- ❑ IdentObj getO1AltFmtLabel(int nIndex)
- ❑ IdentObj getX1AltFmtLabel()
- ❑ IdentObj getX1AltFmtLabel(int nIndex)

Use setDisplay() and omit the nIndex parameter to enable alternate formatting of the axis and/or axis labels. Example:

- ❑ setDisplay(getY1AltFmtFrame(),true); //Enable alt format of Y1Axis
- ❑ setDisplay(getY1AltFmtLabel(),true); //Enable alt format of labels

When alternate formatting is enabled for an axis, use the alternate formatting methods to define how many sections/colors will be defined and the location on the axis to separate the sections.

See [Alternate Formatting](#) on page 174.

AltFmtFrameNumColors

These methods get/set the number of colors/sections to be used for alternate formatting of an axis. Alternate formatting can be used to assign different colors to individual sections of an axis and axis labels. You may also define a text string label that is shown with the alternate formatted section.

Syntax:

```
int getAltFmtFrameNumColors (IdentObj idObj);
void setAltFmtFrameNumColors (IdentObj idObj, int newValue);
```

where:

idObj

Is the object ID of an alternate formatting axis object (for example, getY1AltFmtFrame(), getO1AltFmtFrame(), and others).

newValue

For setAltFmtFrameNumColors(), number of colors/sections to use for alternate formatting of the specified axis (idObj). It can be a value in the range 1...100. The default value is 2.

Return:

int

For getAltFmtFrameNumColors(), number of colors/sections that are used for alternate formatting of the specified axis (idObj).

See [Alternate Formatting](#) on page 174.

AltFmtMatchAxisLabelColor

This property controls whether Y-Axis Labels associated with the alternate frame region inherit the color specified for that region. When enabled, the labels are colored with the color specified for the region. When disabled, labels are displayed using the default color.

Syntax:

```
void setAltFmtMatchAxisLabelColor(boolean);  
boolean = getAltFmtMatchAxisLabelColor();
```

where:

boolean

Can be one of the following:

true

Causes Y axis labels to inherit the color of alternate frame regions.

false

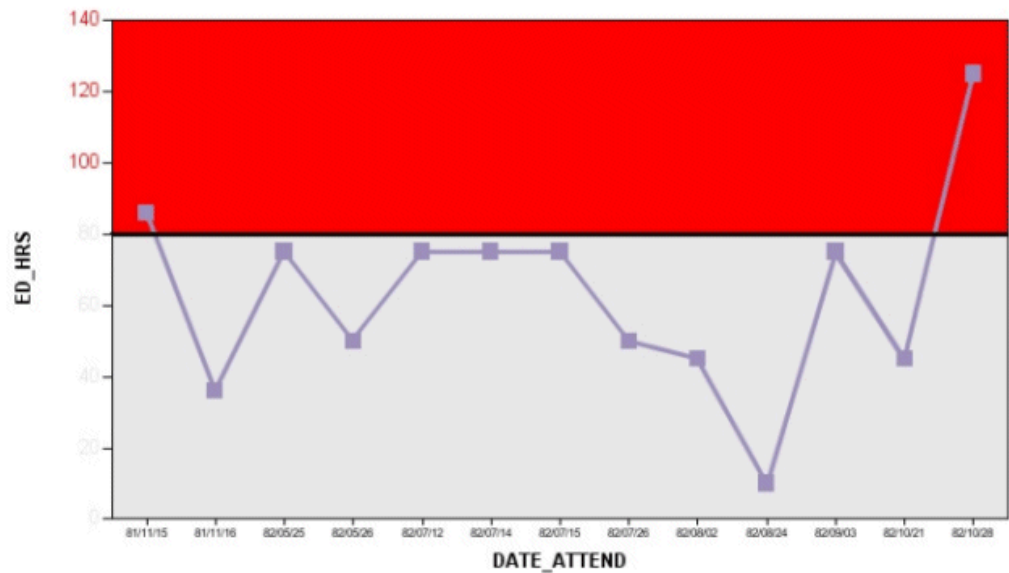
Causes Y axis labels to use the default color. This value is the default.

Example:

The following request sets the setAltFmtMatchAxisLabelColor property to true:

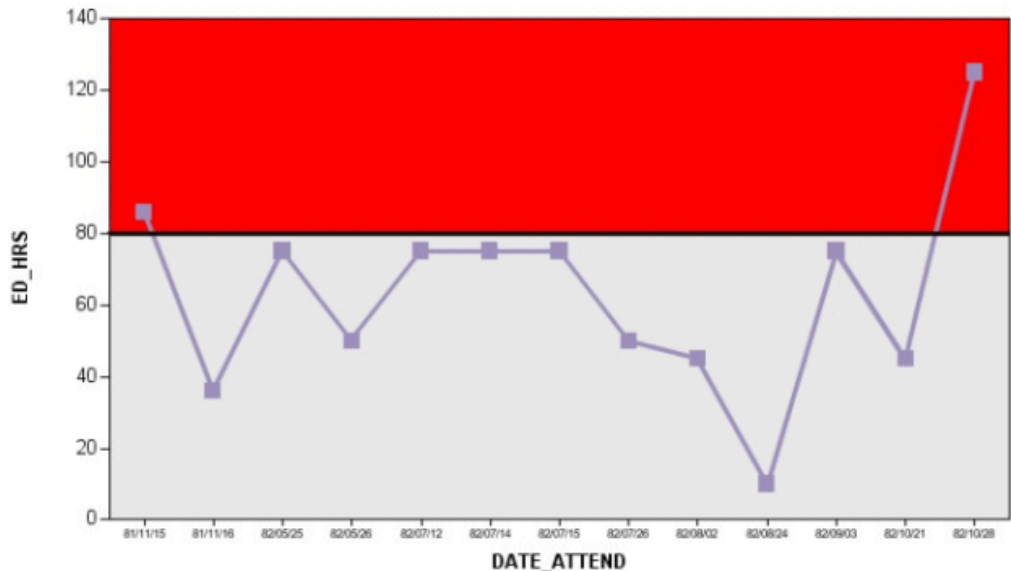
```
setDisplay(getY1AltFmtFrame(), true);  
setAltFmtFrameNumColors(getY1AltFmtFrame(), 2);  
setAltFmtFrameSeparator(getY1AltFmtFrame(0), 80.0);  
setFillColor(getY1AltFmtLabel(0), new Color(0, 0, 0));  
setFillColor(getY1AltFmtFrame(1), new Color(255, 0, 0));  
setReferenceLineValueY1(0, 80.0);  
setLineWidth(getReferenceLineY1(0), 2);  
setFillColor(getY1AltFmtFrame(0), new Color(232, 232, 232));  
setAltFmtMatchAxisLabelColor(true);
```

With `setAltFmtMatchAxisLabelColor(true)`, the Y axis labels inherit the color red when the alternate frame region is red and inherit the color grey when the alternate frame region is grey.



Changing `setAltFmtMatchAxisLabelColor()` to `false`, causes the Y axis labels to use the default color, not the colors of the alternate frame regions:

```
setAltFmtMatchAxisLabelColor(false);
```



AutoHideDataTextOnSmallRisers

The `setAutoHideDataTextOnSmallRisers` property, when enabled, suppresses the display of data text if the height of the riser is smaller than the height of the data text.

Syntax:

```
void setAutoHideDataTextOnSmallRisers(boolean);
```

```
boolean = getAutoHideDataTextOnSmallRisers();
```

where:

boolean

Can be one of the following:

`true`

Suppresses the display of data text that is larger than its riser.

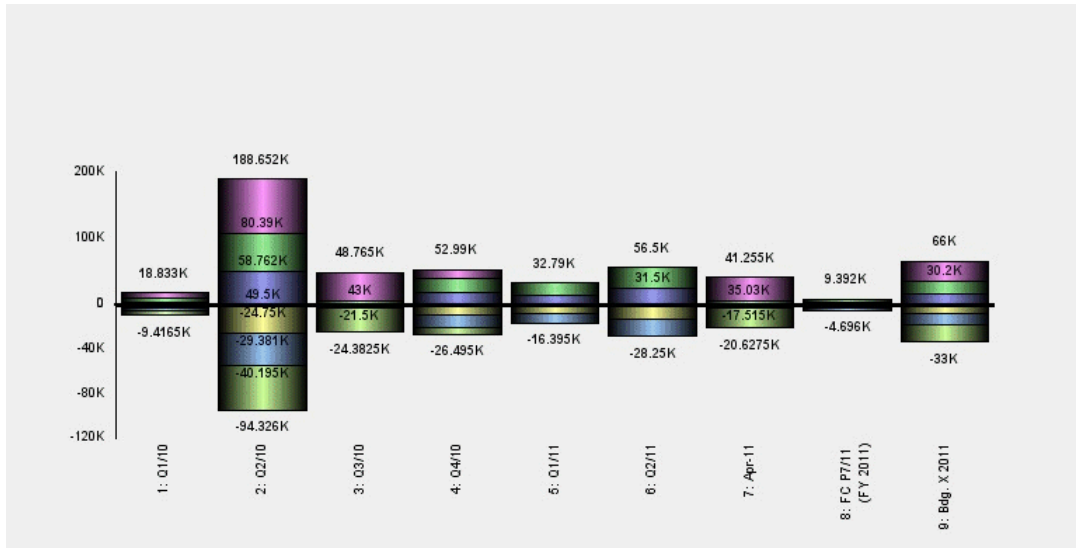
`false`

Displays all data text regardless of the height of the riser. This value is the default.

Example:

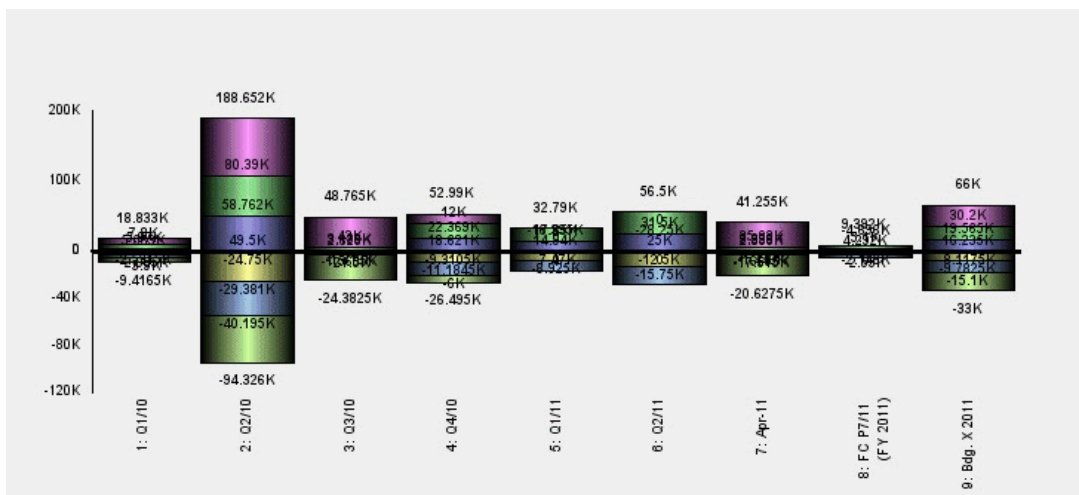
```
setAutoHideDataTextOnSmallRisers(true);
```

With `setAutoHideDataTextOnSmallRisers(true)`, the risers that are too small for data text do not have any text in them.



```
setAutoHideDataTextOnSmallRisers(false);
```

With `setAutoHideDataTextOnSmallRisers(false)`, the risers that are too small for data text still have the text in them. The text in the small risers overwrites text from other risers.



Autoshading

In 3D graphs and graphs where 2.5D depth is applied with DepthRadius, this property enables or disables automatic shading of risers in the graph. If a 3D-graph type is selected, it also enables and disables shading of the 3D cube that surrounds the graph. If this property is set to true, light source settings are used to automatically shade the faces of a 3D graph, simulating a cast light beam. If this property is set to false, all sides of 3D risers or the 3D cube is the same color. It is not possible to color each face separately.

Syntax:

```
setAutoshading(boolean);
```

```
boolean = getAutoshading();
```

where:

boolean

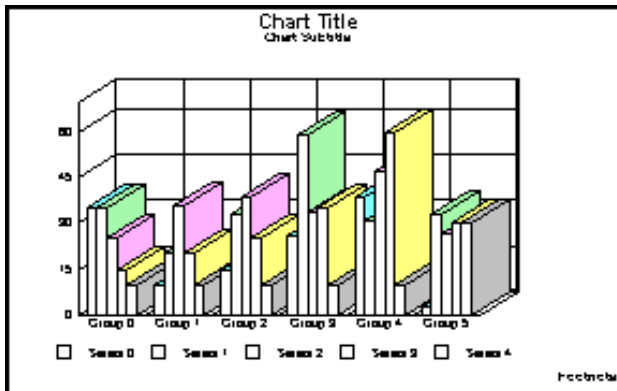
Can be one of the following:

[true](#) enables autoshading. This value is the default.

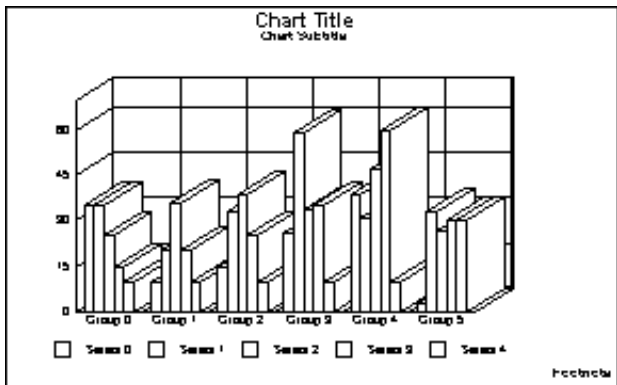
[false](#) disables autoshading.

Example:

```
setAutoshading(true);
```

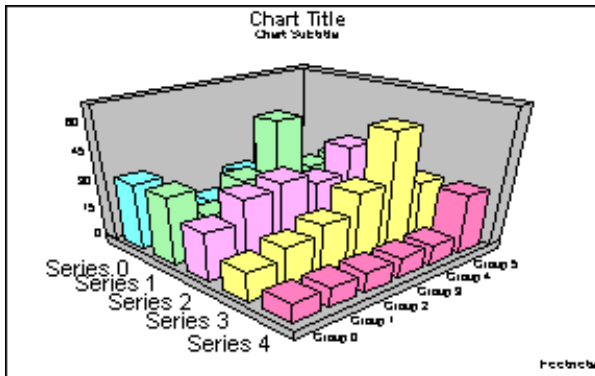


```
setAutoshading(false);
```

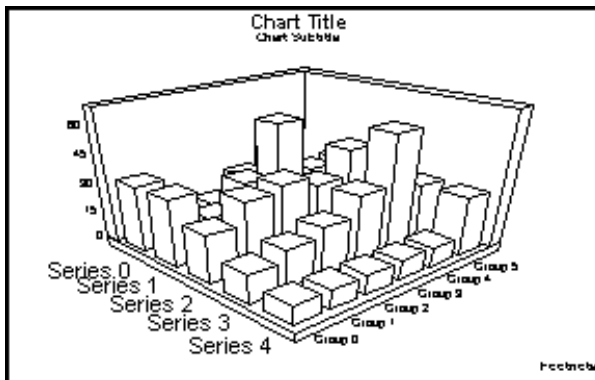


```
setGraphType(0);
```

```
setAutoshading(true);
```



```
setAutoshading(false);
```



Note:

- ❑ When a 3D graph is used and Autoshading is disabled, the `CubeLightSourceX`, `CubeLightSourceY`, and `CubeLightSourceZ` properties are also disabled.
- ❑ The `AutoshadingWalls` property can be used to enable or disable shading of the 3D cube walls independent of risers.

See [AutoshadingWalls](#) on page 253, [CubeLightSourceX](#) on page 271, [CubeLightSourceY](#) on page 272, [CubeLightSourceZ](#) on page 274.

AutoshadingWalls

This property enables or disables autoshading of the cube walls in a 3D graph. It can be used to enable or disable shading of the 3D cube walls independent of riser shading set by the Autoshading property.

Syntax:

```
setAutoshadingWalls (boolean);  
boolean = getAutoshadingWalls();
```

where:

boolean

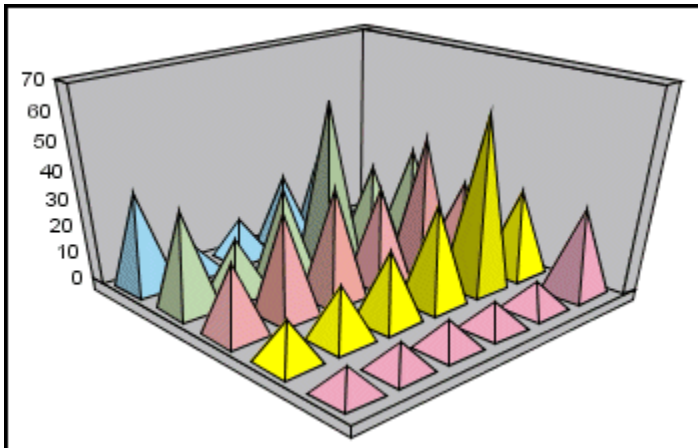
Can be one of the following:

`true` enables autoshading.

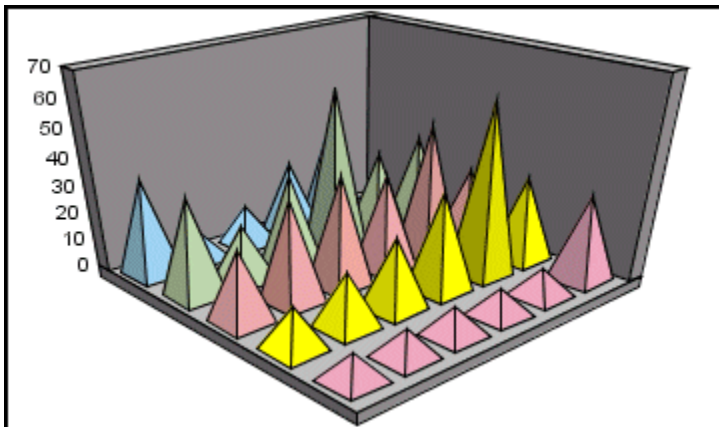
`false` disables autoshading. This value is the default.

Example:

```
setAutoshading(true);  
setAutoshadingWalls(false);
```



```
setAutoshading(true);  
setAutoshadingWalls(true);
```

**BiDirectional**

This property reverses the text alignment of all objects and the location of legend text next to label markers. This property is provided for international support only.

Note: `BiDirectional(true)` does not reverse the values of the associated properties and methods. They retain their original values. For example, `getTextJustHoriz()` would return zero (left) even though visually the text is aligned right by `BiDirectional(true)`.

Syntax:

```
setBiDirectional(boolean);
```

```
boolean = getBiDirectional();
```

where:

boolean

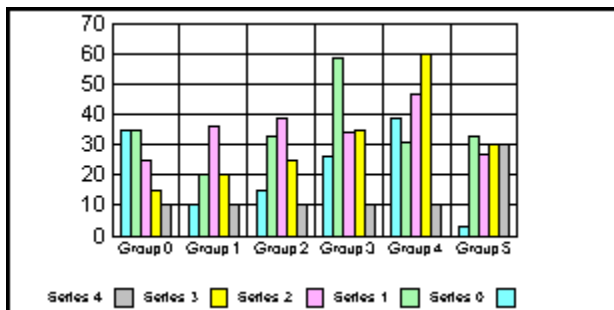
Can be one of the following:

`true` enables right-to-left alignment.

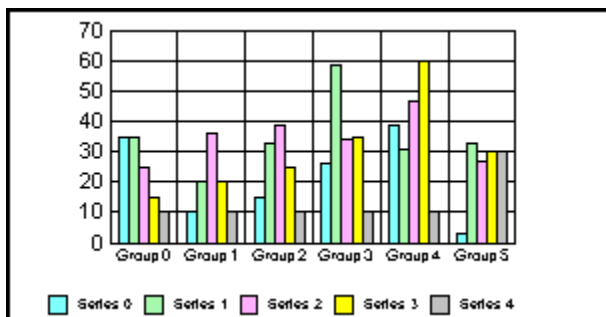
`false` enables normal left-to-right alignment. This value is the default.

Example:

```
setDepthRadius(0); setBiDirectional(true);
```



```
setBiDirectional(false);
```



Note: Use this property with great care. Your application must keep track of the BiDirectional setting and handle the reverse values supplied by associated properties and methods.

BoxPlotType

When the GraphType property is set to 124 (Box Plot) or 130 (Horizontal Box Plot), this property sets the graph to be displayed as either a box, or whisker graph.

Syntax:

`setBoxPlotType(value)`

where:

value

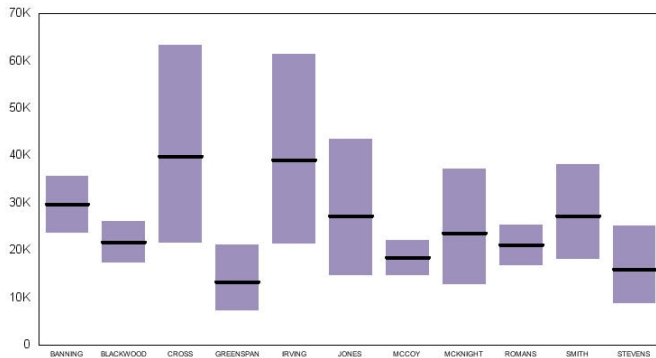
Can be one of the following:

`0` for a box graph. This is the default value.

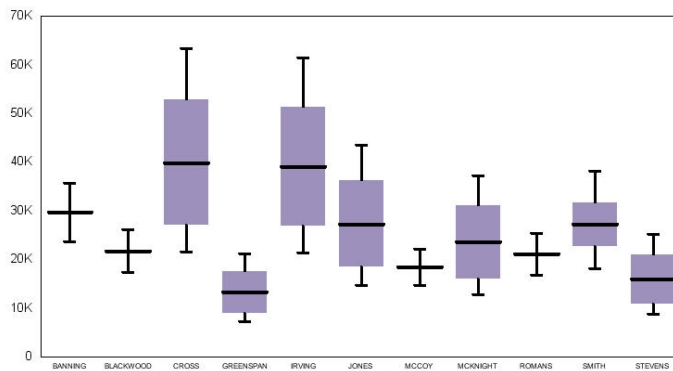
`1` for a whisker graph.

Example:

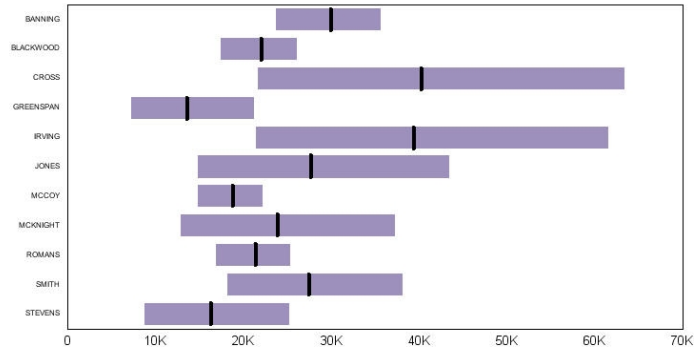
```
setGraphType(124); setBoxPlotType(0);
```



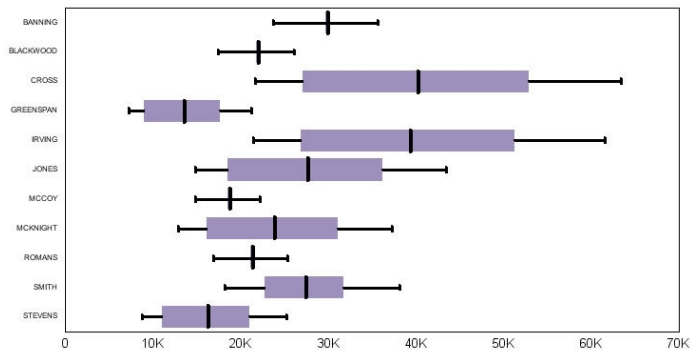
```
setBoxPlotType(1);
```



```
setGraphType (130);  
setBoxPlotType(0);
```



`setBoxPlotType(1);`



Cascade

This property is not intended for use in a normal graphing environment. It is strictly designed for a programmer to determine how setting a method or property may affect other properties and methods.

Syntax:

`setCascade(boolean);`

`boolean = getCascade();`

where:

boolean

Can be one of the following:

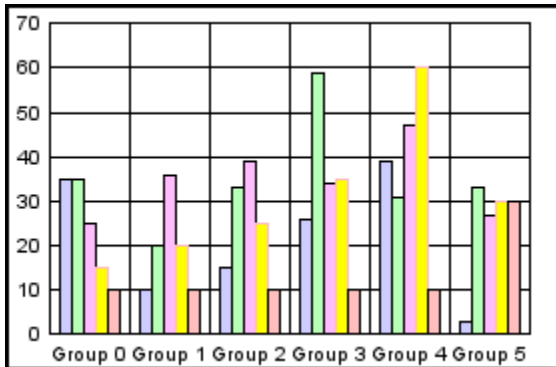
`true` enables cascade mode. The normal relationship between properties and methods is observed. This value is the default.

`false` disables cascade mode. Setting graph characteristics with a method or property will not affect settings by other properties and methods. When Cascade is set to false, properties or methods will never call other properties or methods.

Example:

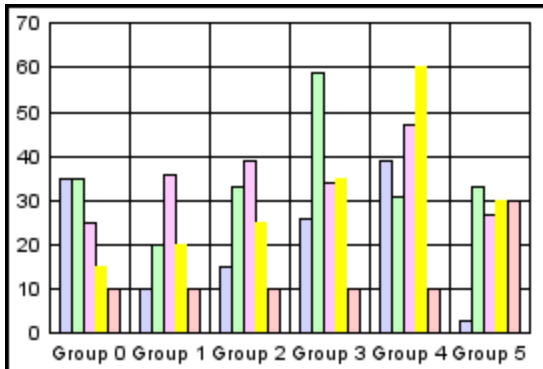
With `Cascade(true)`, `setBorderColor()` will change the border color of series 3 and override the transparent setting.

```
setDepthAngle(0);
setDepthRadius(0);
    setCascade(true);
setTransparentBorderColor(getSeries(3), true);
setBorderColor(getSeries(3), new Color(255,0,0));
```



With Cascade(false), setBorderColor() will not change the border color of series 3 because the border has been set to transparent.

```
setDepthAngle(0);
setDepthRadius(0);
    setCascade(false);
setTransparentBorderColor(getSeries(3),true);
setBorderColor(getSeries(3),new Color(255,0,0));
```



ColorMode

This property sets the color mode to be used in the graph (automatic, color-by-series, color-by-group, automatic riser shading, automatic riser gradient color). For 3D Surface graphs, automatic mode (0) will cause the graph to be colored by height. For bar, line, and area graphs, automatic mode (0) will color the graph by series if there is more than one series or by groups if there is only one series.

Syntax:

```
setColorMode(value);
value = getColorMode();
```

where:

value

Can be one of the following:

0 for Automatic. Use the group as the series if there is only one series. 3D Connected Series Ribbon and Areas are colored by series. 3D Group Ribbon and Areas are colored by group. 0 is the default value.

1 for Color by Series.

2 for Color by Group.

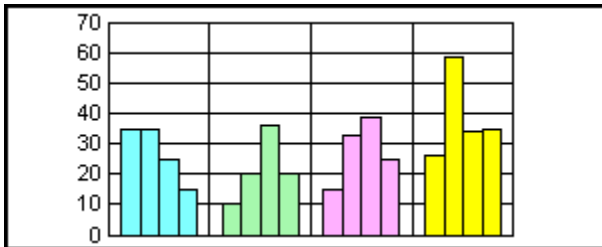
3 for Color by Height.

4 for automatic riser shading. Makes each group of risers a lighter or darker shade of the previous group. Use in conjunction with `setSeriesLooping()` and `setAutoColorLerpFactor()`. For more information, see [Risers and Markers](#) on page 92.

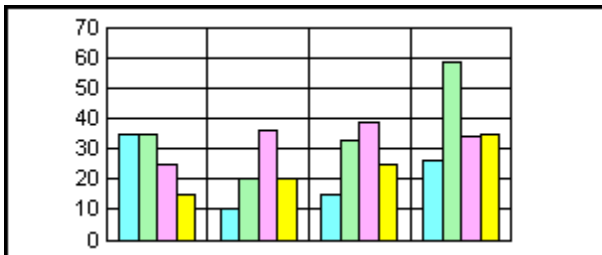
5 for automatic riser gradient coloring. Creates a gradient color effect between the risers. Use in conjunction with `setAutoColorGradientStartColor()` and `setAutoColorGradientEndColor()`. For more information, see [Risers and Markers](#) on page 92.

Example:

```
setDepthRadius(0);
setViewableGroups(4);
setViewableSeries(4);
setDataScrollerPresenceGroups(0);
setLegendDisplay(false);
setO1LabelDisplay(false);
setColorMode(2);
```



```
setColorMode(1);
```



Note: When color by group (`setColorMode(2);`) is selected, the legend is always automatically turned off.

See [ExactColorByHeight](#) on page 352.

ConnectLineMarkers

For horizontal or vertical line graphs where 2.5D depth effect is not applied (for example, DepthRadius=0), this property enables or disables the display of lines that connect markers in the graph. Use the MarkerSizeDefault property to set the size of the markers.

Syntax:

```
setConnectLineMarkers(boolean);  
boolean = getConnectLineMarkers();
```

where:

boolean

Can be one of the following:

[true](#) draws connecting lines. This value is the default.

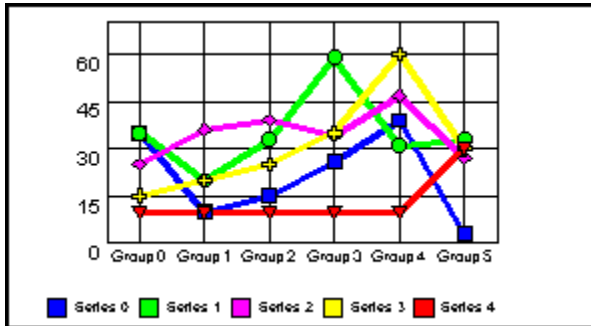
[false](#) produces a "floating marker" effect with only markers and no connecting lines.

Example:

```

setDepthRadius(0);
setMarkerSizeDefault(60);
  setConnectLineMarkers(true);
setGraphType(41);

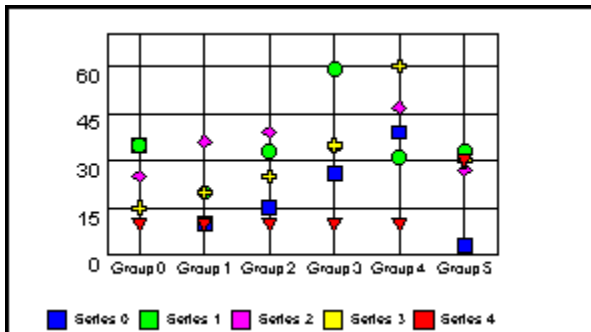
```



```

setConnectLineMarkers(false);

```



Note: A vertical or horizontal line graph must be selected with the GraphType property and the DepthRadius property must be set to zero.

ConnectParetoBars

In a Pareto Graph (GraphType=102), this property affects how the Pareto series bars are drawn.

Syntax:

```

setConnectParetoBars(boolean);
boolean = getParetoBars();

```

where:

boolean

Can be one of the following:

[true](#) draws connected Pareto bars with no space between them. This value is the default.

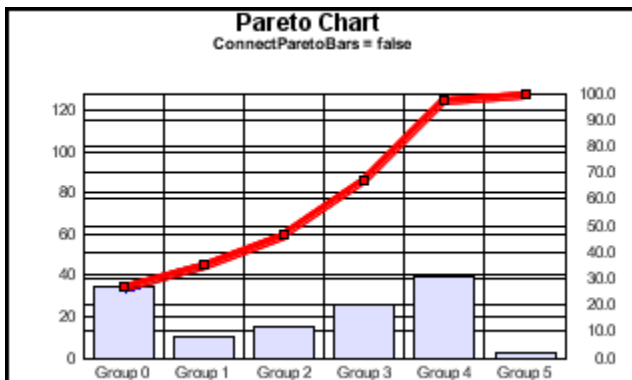
[false](#) draws Pareto bars with space between them.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setDisplayParetoSeries(0);
setGraphType(102);
setTitleString("Pareto Chart");
setSubtitleString("ConnectParetoBars = false");
setConnectParetoBars(false);

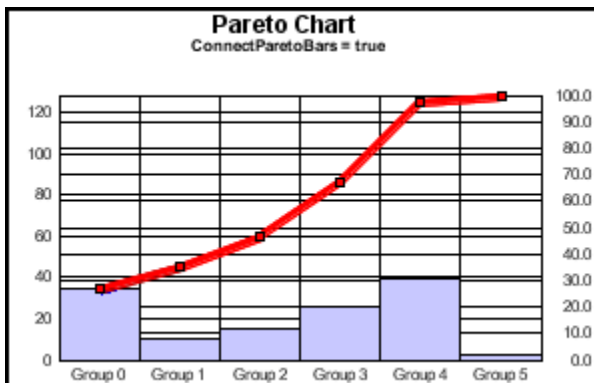
```



```

setSubtitleString("ConnectParetoBars = true");
setConnectParetoBars(true);

```

**ConnectScatterMarkers**

This property enables or disables the display of connecting lines between markers in a scatter graph. The color of each line matches the color of the series marker. You can use the MarkerSizeDefault property to set the size of the markers.

Syntax:

```
setConnectScatterMarkers(boolean);
boolean = getConnectScatterMarkers();
```

where:

boolean

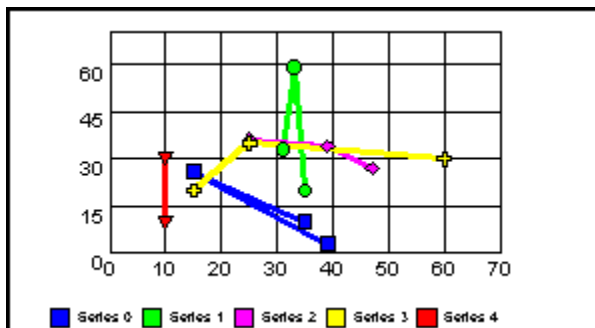
Can be one of the following:

true draws a line connecting all markers in the same series, in the order the points are in the data matrix.

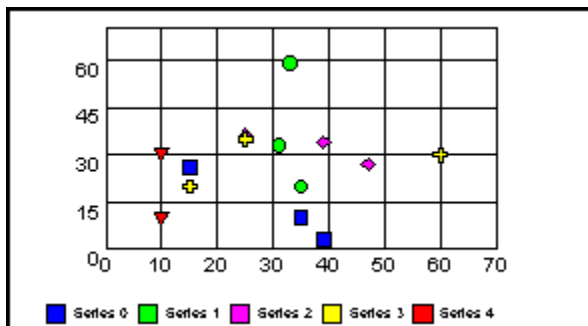
false does not draw connecting lines. This value is the default.

Example:

```
setMarkerSizeDefault(60);
setGraphType(61);
setConnectScatterMarkers(true);
```



```
setConnectScatterMarkers(false);
```



Note: A scatter graph must be selected with the `GraphType` property.

See [ConnectLineMarkers](#) on page 262, [MarkerSizeDefault](#) on page 412.

CubeFocusFactor

This property sets the focus factor for setting perspective distortion in a 3D graph. It is used in conjunction with the `CubeViewer` properties to control the field of view of the cube within the graph frame.

Syntax:

```
setCubeFocusFactor(value);
```

```
value = getCubeFocusFactor();
```

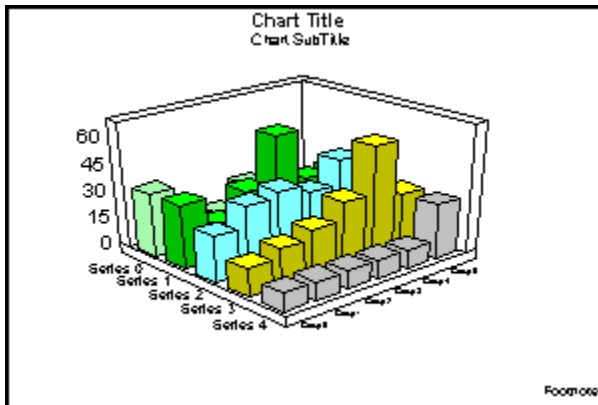
where:

value

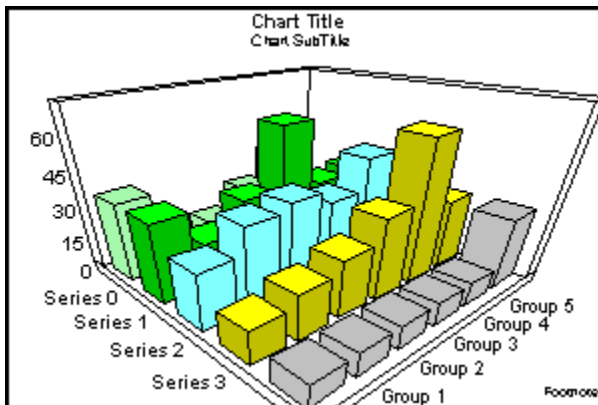
Is a value between 0 and 100. This value is a Java double-precision number. 50.0 is the default value.

Example:

```
setGraphType(0);
setCubeViewerZ(93.0);
setCubeFocusFactor(95.0);
```



```
setCubeFocusFactor(20.0);
setCubeViewerZ(13);
```



Note: If CubelsometricProjection is set to true, this property will be ignored.

CubelsometricProjection

This property enables or disables isometric projection.

Syntax:

```
setCubeIsometricProjection(boolean);  
boolean = getCubeIsometricProjection();
```

where:

boolean

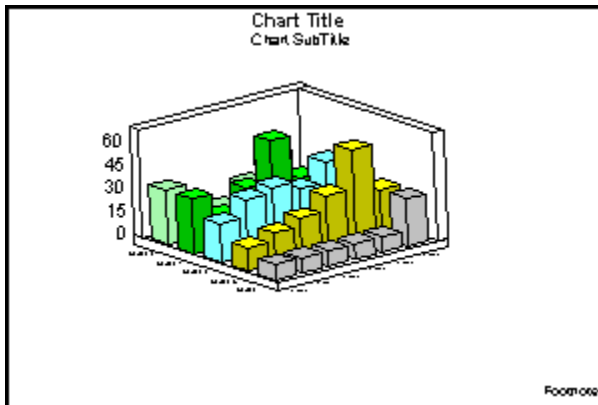
Can be one of the following:

[true](#) enables isometric projection. Perspective distortion (focus) factor is ignored and the graph cube is projected with no line of sight perspective distortion, but perfectly parallel lines, like a mechanical drawing.

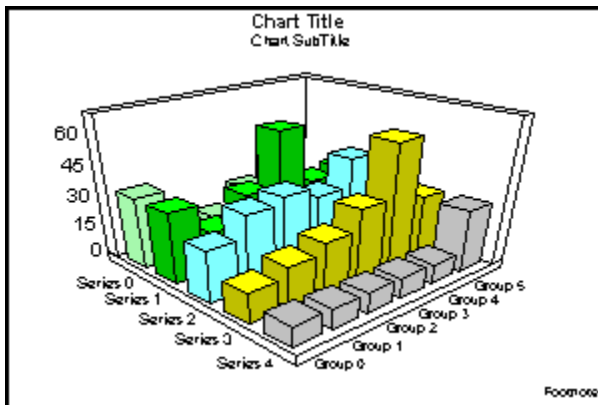
[false](#) disables isometric projection. This value is the default.

Example:

```
setGraphType(0); setCubeIsometricProjection(true);
```



```
setCubeIsometricProjection(false);
```



See [CubeFocusFactor](#) on page 267.

CubeLightSourceX

This property sets the cube light source for the X-coordinate (unit space). In the typical configuration or view, larger values make the face of the left wall cube a lighter color. This property is ignored if Autos shading is set to false.

Syntax:

```
setCubeLightSourceX(value);  
value = getCubeLightSourceX();
```

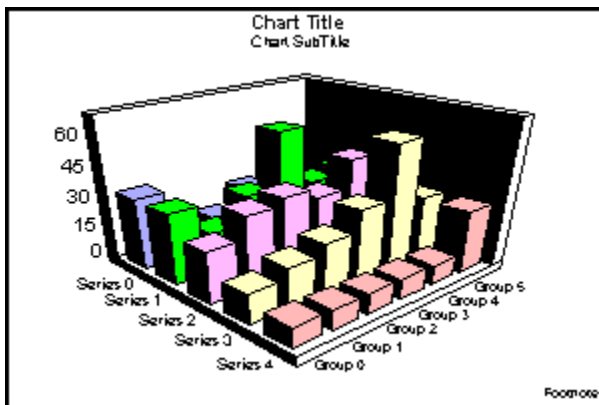
where:

value

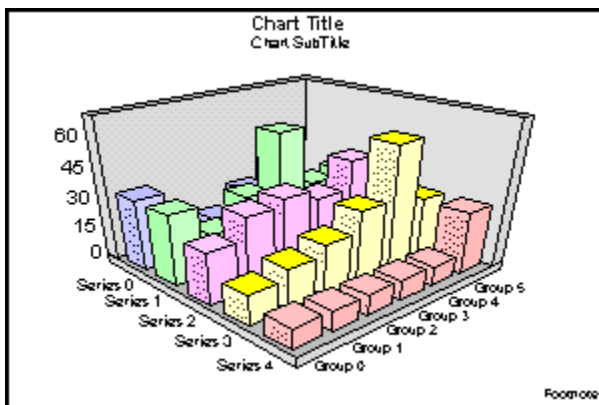
Is a value between 1 and 100. This value is a Java double-precision number. 0.0 is the default value.

Example:

```
setGraphType(0);  setCubeLightSourceX(50.0);
```



```
setCubeLightSourceX(10.0);
```



See [CubeLightSourceY](#) on page 272, [CubeLightSourceZ](#) on page 274.

CubeLightSourceY

This property sets the cube light source for the Y-coordinate (unit space).

Syntax:

```
setCubeLightSourceY(value);  
value = getCubeLightSourceY();
```

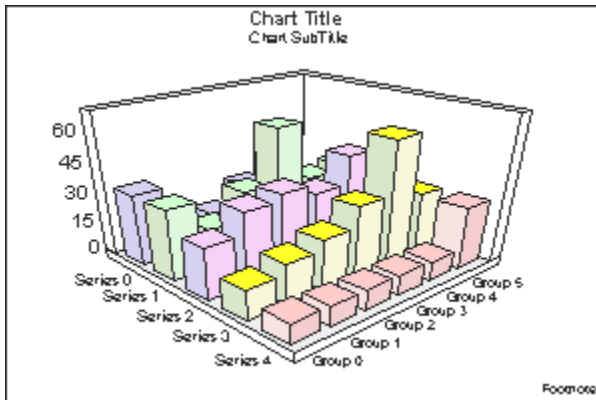
where:

value

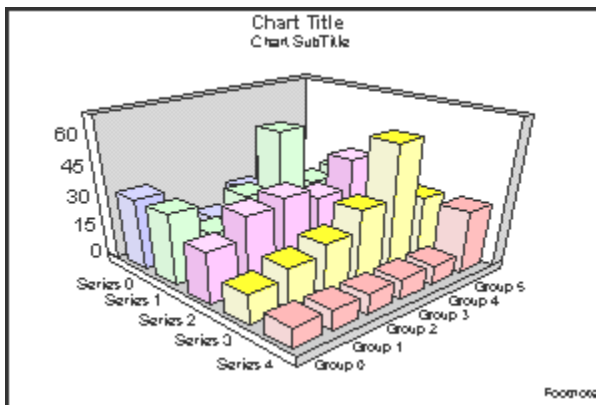
Is a value between 1 and 100. This value is a Java double-precision number. 40.0 is the default value.

Example:

```
setGraphType(0);  setCubeLightSourceY(60.0);
```



```
setCubeLightSourceY(20.0);
```



See [CubeLightSourceX](#) on page 271, [CubeLightSourceZ](#) on page 274.

CubeLightSourceZ

This property sets the cube light source for the Z-coordinate (unit space).

Syntax:

```
setCubeLightSourceZ(value);
```

```
value = getCubeLightSourceZ();
```

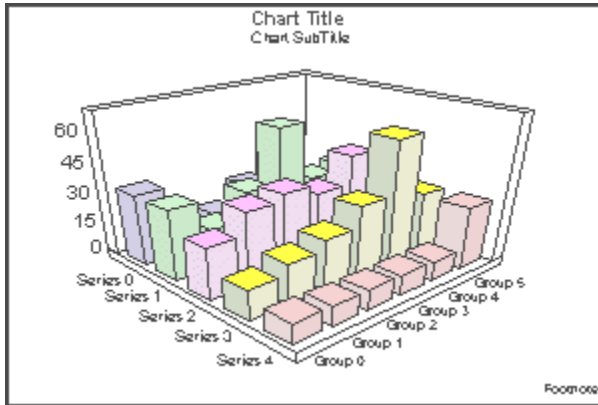
where:

value

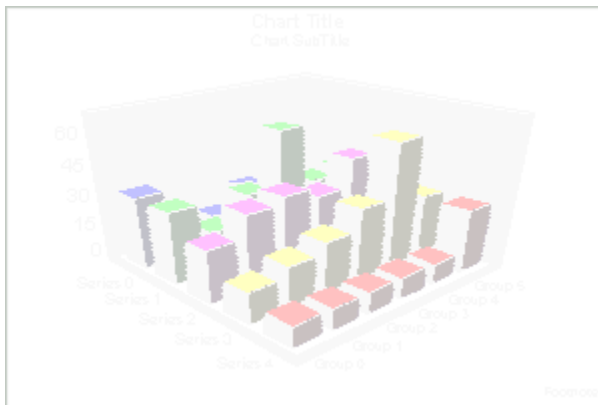
Is a value between 1 and 100. This value is a Java double-precision number. 100.0 is the default value.

Example:

```
setGraphType(0);  setCubeLightSourceZ(80.0);
```



```
setCubeLightSourceZ(40.0);
```



See [CubeLightSourceX](#) on page 271, [CubeLightSourceY](#) on page 272.

CubePanX

This property pans the graph (in 2D virtual coordinates) in the X direction. Before you use this property, the `GraphType` property must select a 3D graph type (0...7, 9...10, 12...14).

Syntax:

```
setCubePanX(value);
```

```
value = getCubePanX();
```

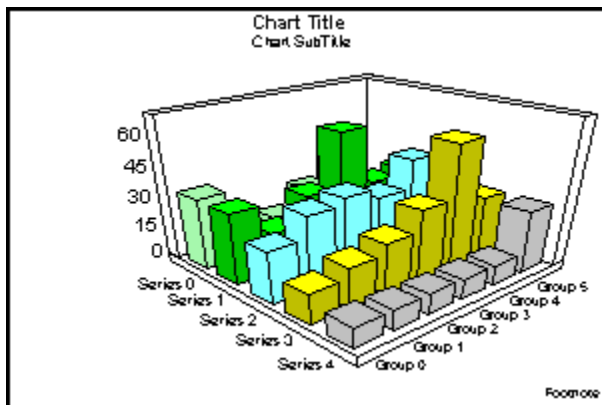
where:

value

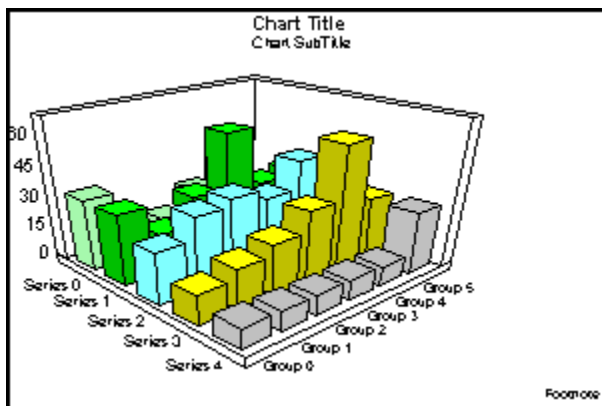
Is a value between 0 and 100. This value is a Java double-precision number. 39.3 is the default value.

Example:

```
setGraphType(0);
setCubePanX(50.0);
```



```
setCubePanX(30.0);
```



See [CubePanY](#) on page 277.

CubePanY

This property pans the graph (in 2D virtual coordinates) in the Y direction. Before you use this property, the GraphType property must select one of the 3D graph types (0...7, 9...10, 12...14).

Syntax:

```
setCubePanY(value);
```

```
value = getCubePanY();
```

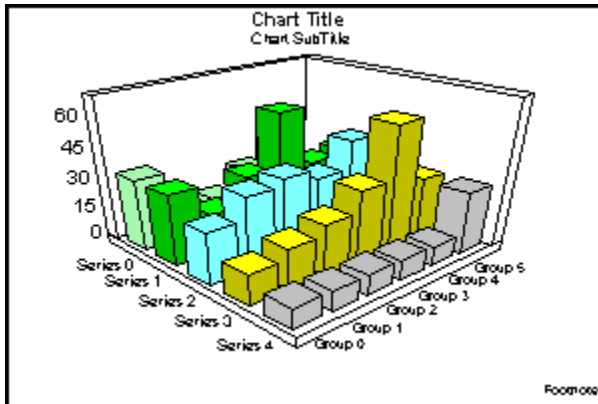
where:

value

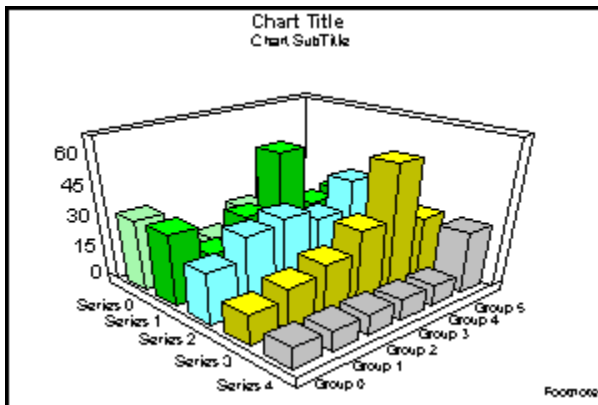
Is a value between 0 and 100. This value is a Java double-precision number. 60.7 is the default value.

Example:

```
setGraphType(0);
setCubePanY(65.0);
```



```
setCubePanY(50.0);
```



See [CubePanX](#) on page 275.

CubeProportional

When a 3D graph type is selected, this property tells the chart to define the axis size proportional to the number of series or groups.

Syntax:

```
setCubeProportional(boolean);
boolean = getCubeProportional();
```

where:

boolean

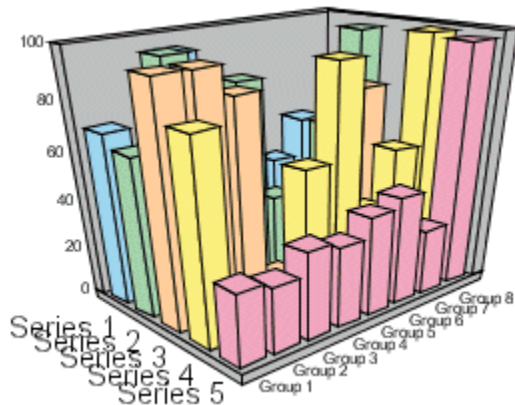
Can be one of the following:

true draws cube proportional to number of groups. This value is the default.

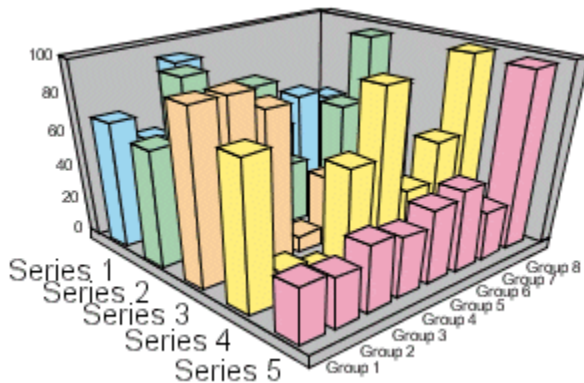
false draws cube proportional to number of series.

Example:

`setGraphType(0); setCubeProportional(true);`



`setGraphType(0); setCubeProportional(false);`



CubeRiserInterpolation

This property controls whether risers in 3D graphs are interpolated (faster) or explicitly calculated (more accurate). For isometric projections (`CubelsometricProjection` is true), the results are the same, so it is always OK to use interpolation. When perspective projection is used (`CubelsometricProjection` is false), users with very large data sets may want to turn this on for faster calculations.

Syntax:

```
setCubeRiserInterpolation(value);
```

```
value = getCubeRiserInterpolation();
```

where:

value

Can be one of the following:

[0](#) for use only with isometric projection for large data sets. 0 is the default value.

[1](#) for isometric only.

[2](#) for never.

See [CubelsometricProjection](#) on page 269.

CubeSizeX

This property gets and sets the size of the 3D cube in the (user) X-direction. Before you use this property, the `GraphType` property must select one of the 3D graph types (0...7, 9...10, 12...14). The dimensions are expressed in virtual coordinates. For details, see [Virtual Coordinate System](#) on page 47.

Syntax:

```
setCubeSizeX(value);
```

```
value = getCubeSizeX();
```

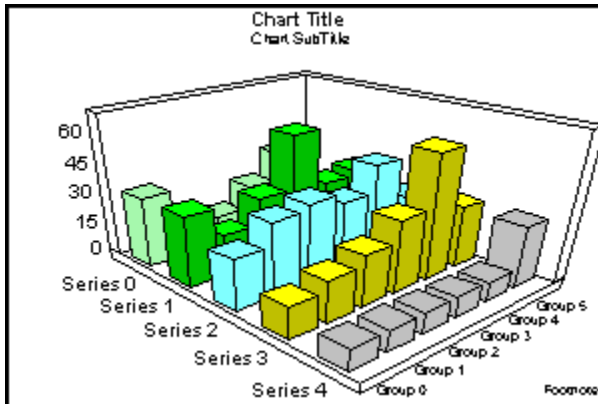
where:

value

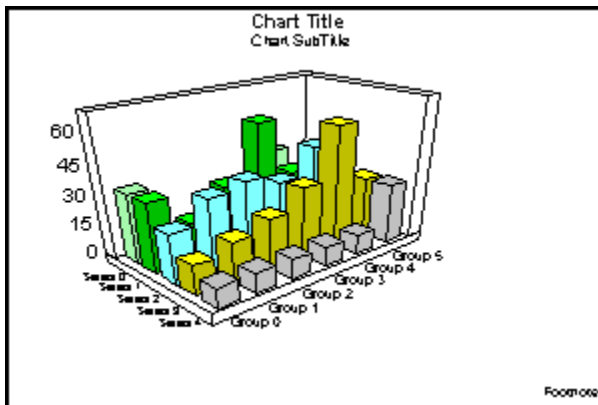
Is a value between 0 and 100. This value is a Java double-precision number. 30.0 is the default value.

Example:

```
setCubeSizeX(40.0);
setGraphType(0);
```



```
setCubeSizeX(15.0);
```



See [CubeSizeY](#) on page 281, [CubeSizeZ](#) on page 283.

CubeSizeY

This property gets and sets the size of the 3D cube in the (user) Y-direction. Before you use this property, the GraphType property must select one of the 3D graph types (0...7, 9...10, 12...14). The dimensions are expressed in virtual coordinates. For details, see [Virtual Coordinate System](#) on page 47.

Syntax:

```
setCubeSizeY(value);
```

```
value = getCubeSizeY();
```

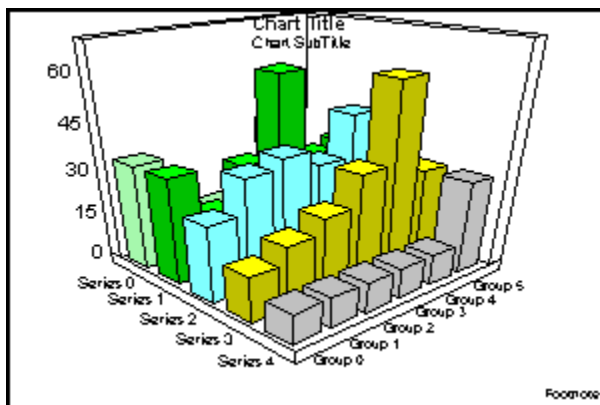
where:

value

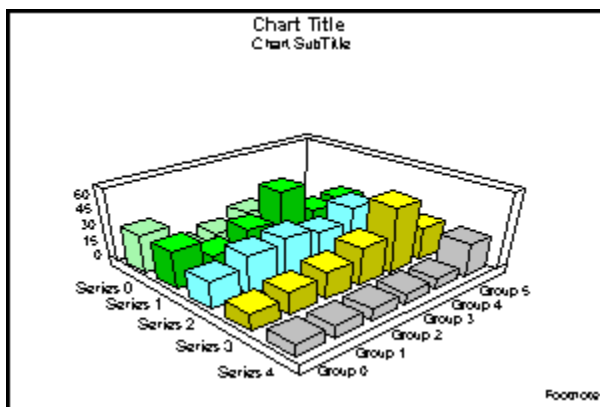
Is a value between 0 and 100. This value is a Java double-precision number. 25.0 is the default value.

Example:

```
setCubeSizeY(40.0);  
setGraphType(0);
```



```
setCubeSizeY(10.0);
```



See [CubeSizeX](#) on page 280, [CubeSizeZ](#) on page 283.

CubeSizeZ

This property gets and sets the size of the 3D cube in the (user) Z-direction. Before you use this property, the GraphType property must select one of the 3D graph types (0...7, 9...10, 12...14). The dimensions are expressed in virtual coordinates. For details, see [Virtual Coordinate System](#) on page 47.

Syntax:

```
setCubeSizeZ(value);
```

```
value = getCubeSizeZ();
```

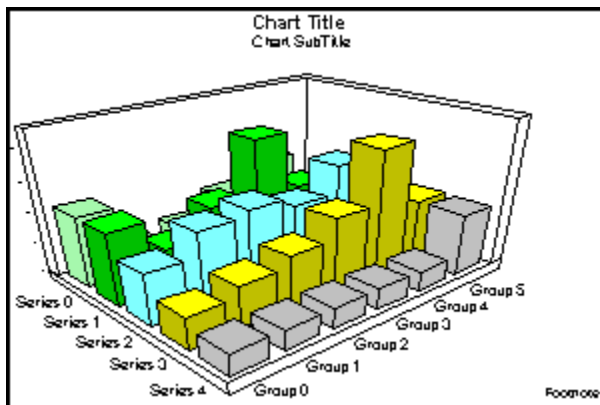
where:

value

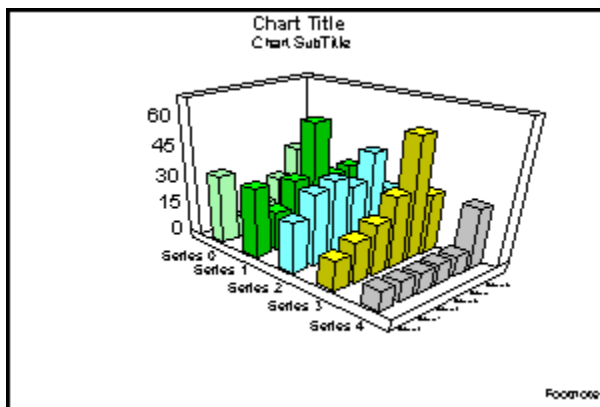
Is a value between 0 and 100. This value is a Java double-precision number. 30.0 is the default value.

Example:

```
setCubeSizeZ(40.0);
setGraphType(0);
```



```
setCubeSizeZ(15.0);
```



See [CubeSizeX](#) on page 280, [CubeSizeY](#) on page 281.

CubeSquareRisers

This property enables or disables square risers.

Syntax:

```
setCubeSquareRisers(boolean);
boolean = getCubeSquareRisers();
```


where:

boolean

Can be one of the following:

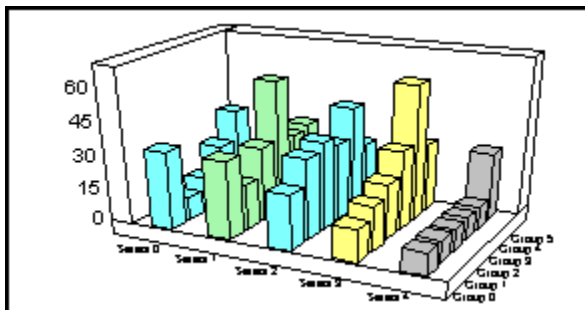
true forces the aspect ratio of 3D risers to be perfectly square regardless of the dimensions of the cube. This value is the default.

false draws risers using the RiserWidth property and the aspect ratio of the cube.

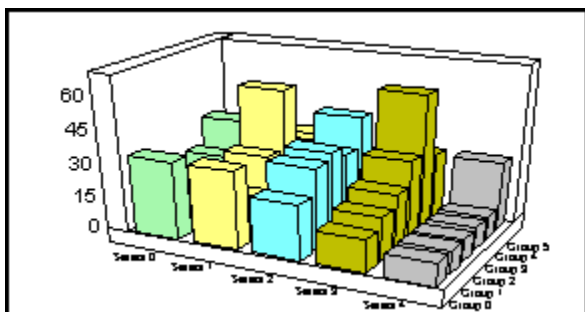
For ribbons and floating graph types, the Riser3DThicknessY property determines the height of the risers.

Example:

```
setCubeSquareRisers(true);
setGraphType(0);
setViewing3DAnglePreset(1);
```



```
setCubeSquareRisers(false);
```



CubeTranslationX

This property sets the translation of a 3D cube (in 3D cube coordinates) in the X-direction. In the typical configuration or view, smaller values move the cube to the left size of the graph frame. Larger values move the cube to the right size of the graph frame.

Syntax:

```
setCubeTranslationX(value);  
value = getCubeTranslationX();
```

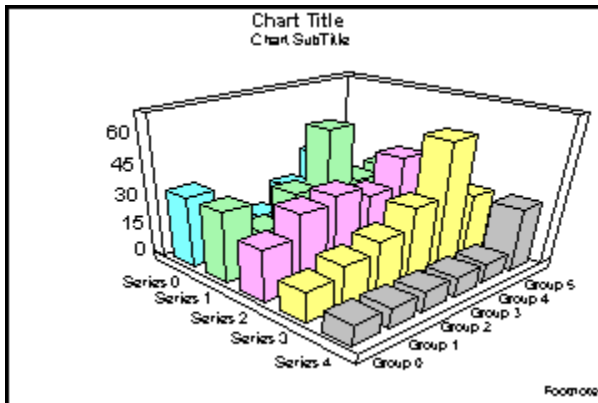
where:

value

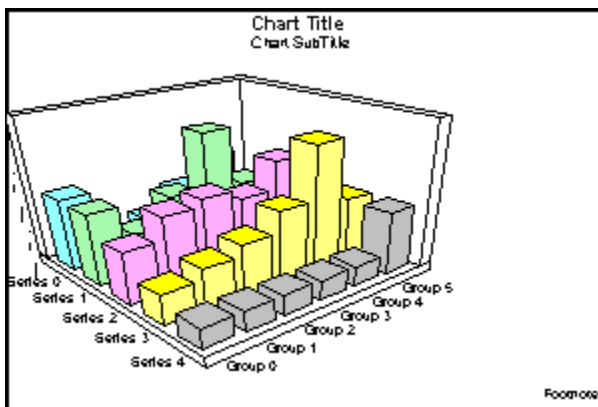
Is a value between 0 and 100. This value is a Java double-precision number. 65.0 is the default value.

Example:

```
setCubeTranslationX(75.0);
setGraphType(0);
```



```
setCubeTranslationX(50.0);
```



See [CubeTranslationY](#) on page 288, [CubeTranslationZ](#) on page 290.

CubeTranslationY

This property sets the translation of a 3D cube (in 3D cube coordinates) in the Y-direction. In the typical configuration or view, smaller values move the cube down within the graph frame. Larger values move the cube up toward the top of the graph frame.

Syntax:

```
setCubeTranslationY(value);  
value = getCubeTranslationY();
```

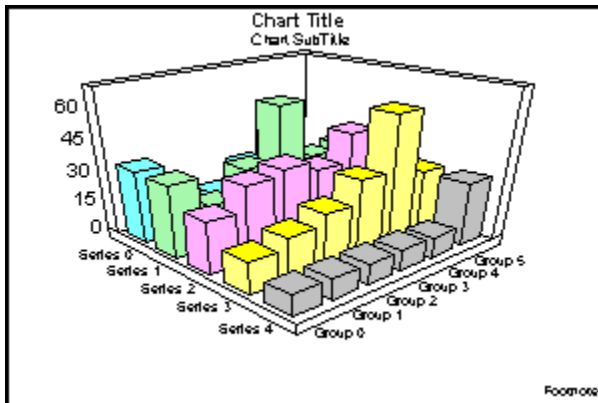
where:

value

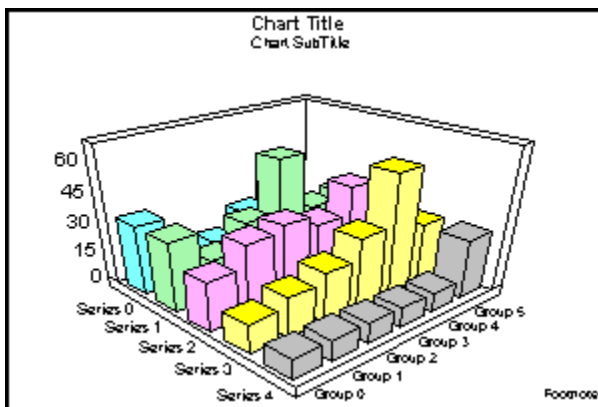
Is a value between 0 and 100. This value is a Java double-precision number. 42.5 is the default value.

Example:

```
setCubeTranslationY(50.0);
setGraphType(0);
```



```
setCubeTranslationY(35.0);
```



See [CubeTranslationX](#) on page 286, [CubeLightSourceZ](#) on page 274.

CubeTranslationZ

This property sets the translation of a 3D cube (in 3D cube coordinates) in the Z-direction. In the typical configuration or view, smaller values make the cube smaller within the graph frame. Larger values make the cube larger within the graph frame.

Syntax:

```
setCubeTranslationZ(value);  
value = getCubeTranslationZ();
```

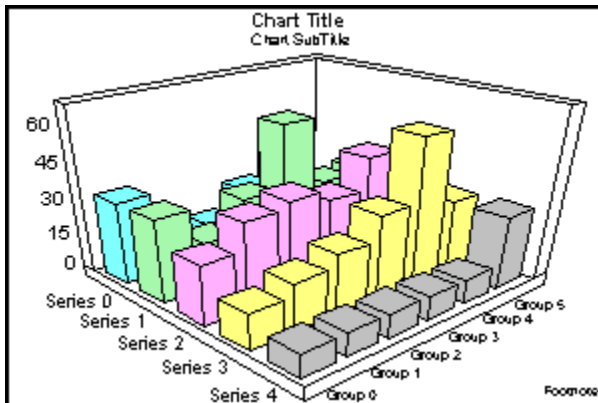
where:

value

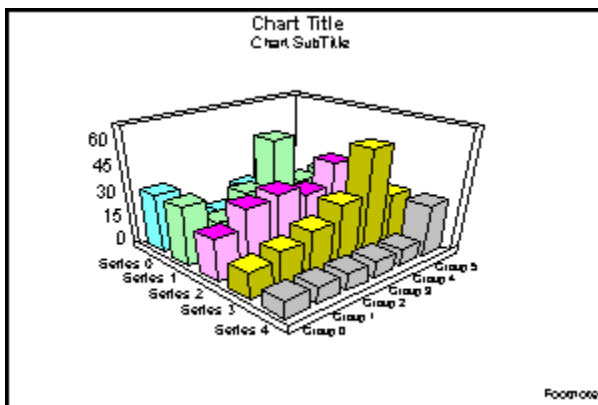
Is a value between 0 and 100. This value is a Java double-precision number. 50.0 is the default value.

Example:

```
setCubeTranslationZ(85.0);
setGraphType(0);
```



```
setCubeTranslationZ(15.0);
```



See [CubeTranslationX](#) on page 286, [CubeTranslationY](#) on page 288.

CubeViewerX

This property sets the viewer location (in 3D coordinates) in the X-direction. In the typical configuration or view, smaller values move the cube toward the right side of the graph frame. Larger values move the cube toward the left side of the graph frame.

Syntax:

```
setCubeViewerX(value);
```

```
value = getCubeViewerX();
```

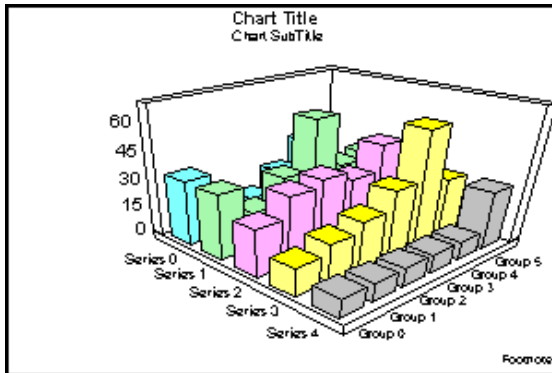
where:

value

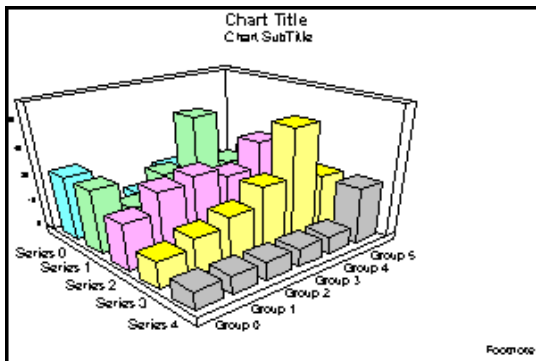
Is a value between 0 and 100. This value is a Java double-precision number. 50.0 is the default value.

Example:

```
setGraphType(0);
setCubeViewerX(25.0);
```



```
setCubeViewerX(75.0);
```



See [CubeViewerY](#) on page 293, [CubeViewerZ](#) on page 294.

CubeViewerY

This property sets the viewer location (in 3D coordinates) in the Y-direction. In the typical configuration or view, smaller values move the cube up toward the top of the graph frame. Larger values move the cube down within the graph frame.

Syntax:

```
setCubeViewerY(value);
value = getCubeViewerY();
```

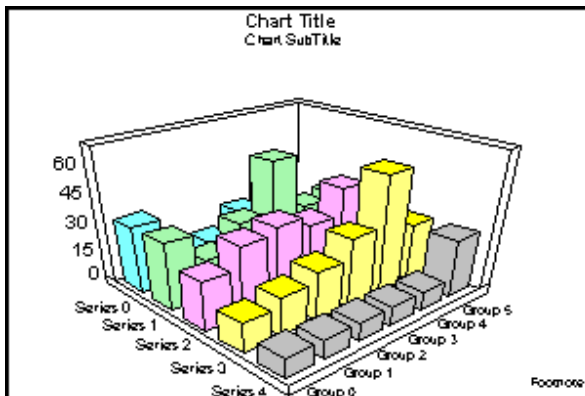
where:

value

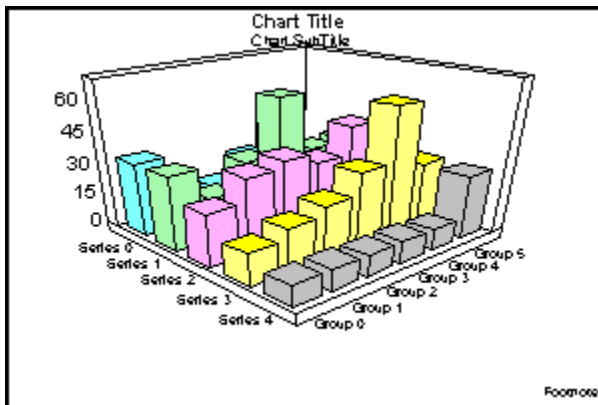
Is a value between 0 and 100. This value is a Java double-precision number. 60.0 is the default value.

Example:

```
setGraphType(0);
setCubeViewerY(80.0);
```



```
setCubeViewerY(40.0);
```



See [CubeViewerX](#) on page 292, [CubeViewerZ](#) on page 294.

CubeViewerZ

This property sets the viewer location (in 3D coordinates) in the Z-direction. Smaller values make the cube larger within the frame, larger values make the cube smaller within the graph frame.

Syntax:

```
setCubeViewerZ(value);
value = getCubeViewerZ();
```

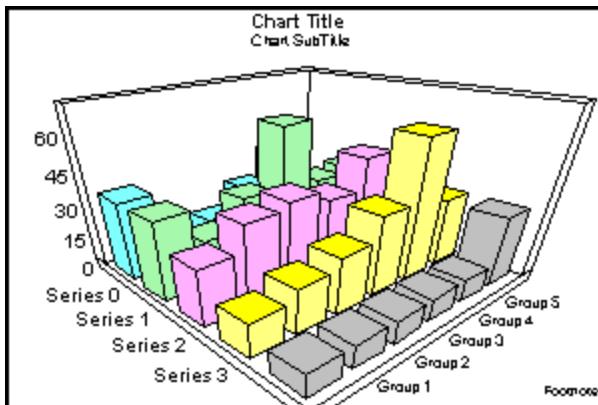
where:

value

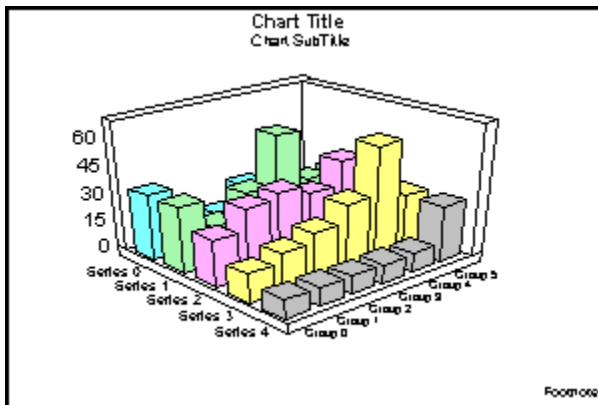
Is a value between 0 and 100. This value is a Java double-precision number. 36.8 is the default value.

Example:

```
setGraphType(0);
setCubeViewerZ(15.0);
```



```
setCubeViewerZ(75.0);
```



See [CubeFocusFactor](#) on page 267, [CubeViewerX](#) on page 292, [CubeViewerY](#) on page 293.

CubeWallThickX

This property sets the thickness of the 3D-cube wall in the (user) X-direction. In the typical configuration or view of a 3D graph, this property controls the thickness of the left wall of the 3D cube.

Syntax:

```
setCubeWallThickX(value);
```

```
value = getCubeWallThickX();
```

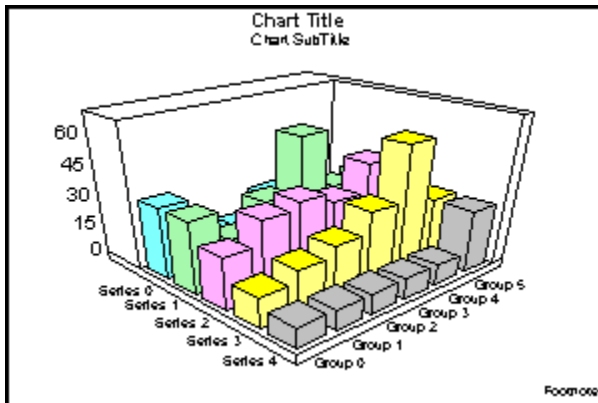
where:

value

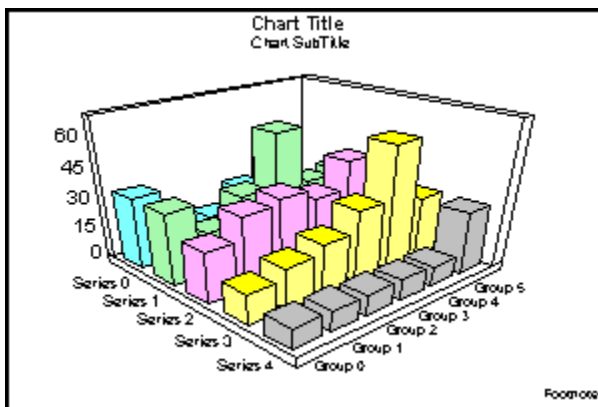
Is a value between 0 and 100. This value is a Java double-precision number. 20.0 is the default value.

Example:

```
setGraphType(0);
setCubeWallThickX(80.0);
```



```
setCubeWallThickX(10.0);
```



See [CubeWallThickY](#) on page 298, [CubeWallThickZ](#) on page 300.

CubeWallThickY

This property gets and sets the thickness of the 3D-cube wall in the (user) Y-direction. In the typical configuration or view of a 3D graph, this property controls the thickness of the floor of the 3D cube.

Syntax:

```
setCubeWallThickY(value);
```

```
value = getCubeWallThickY();
```

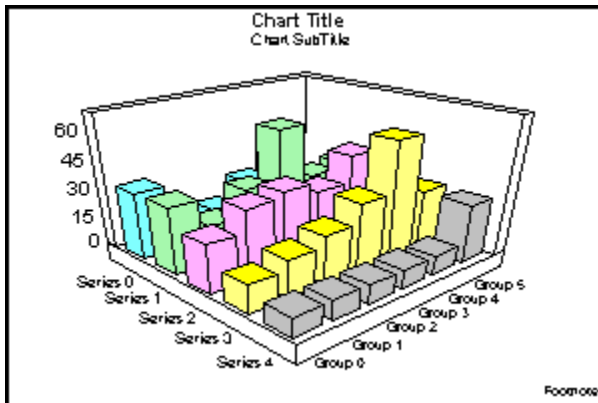
where:

value

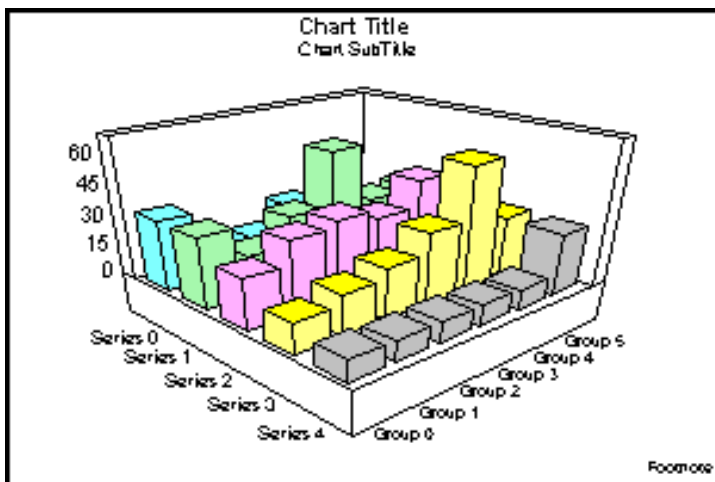
Is a value between 0 and 100. This value is a Java double-precision number. 20.0 is the default value.

Example:

```
setGraphType(0);
setCubeWallThickY(40.0);
```



```
setCubeWallThickY(80.0);
```



See [CubeWallThickX](#) on page 296, [CubeWallThickZ](#) on page 300.

CubeWallThickZ

This property gets and sets the thickness of the 3D cube wall in the (user) Z-direction. In the typical configuration or view of a 3D graph, this property controls the thickness of the right wall of the 3D cube.

Syntax:

```
setCubeWallThickZ(value);
```

```
value = getCubeWallThickZ();
```

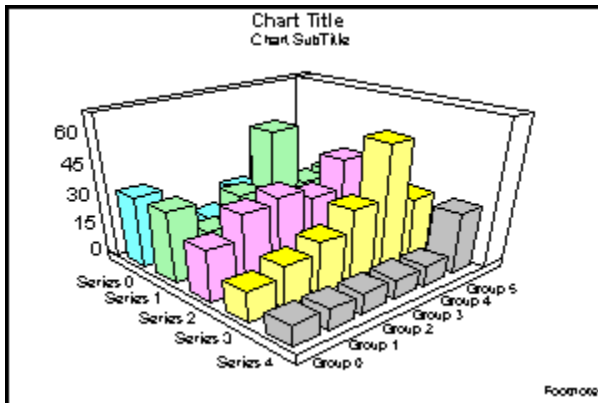
where:

value

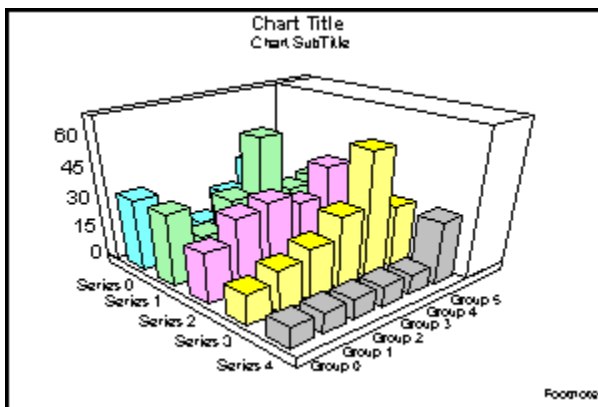
Is a value between 0 and 100. This value is a Java double-precision number. 20.0 is the default value.

Example:

```
setGraphType(0);
setCubeWallThickZ(40.0);
```



```
setCubeWallThickZ(80.0);
```



See [CubeWallThickX](#) on page 296, [CubeWallThickY](#) on page 298.

CubeZoomFactor

This property sets the global scaling factor for zooming in or out.

Syntax:

```
setCubeZoomFactor(value);
```

```
value = getCubeZoomFactor();
```

where:

value

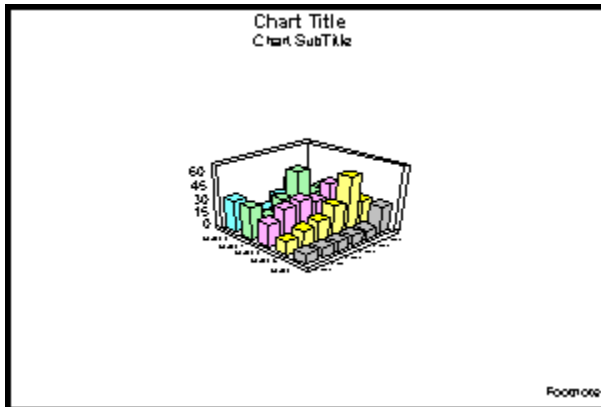
Is a value between 0 and 100. This value is a Java double-precision number. 50.0 is the default value.

Values greater than one zoom in and produce a large display of the graph within the frame.

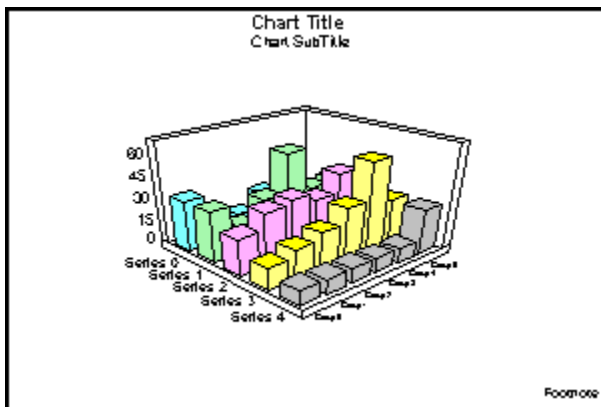
Values between zero and one zoom out and produce a smaller display of the graph within the frame.

Example:

```
setGraphType(0);
setCubeZoomFactor(25.0);
```



```
setCubeZoomFactor(40.0);
```



See [CubeFocusFactor](#) on page 267.

CurveFitEquationDisplay

The value assigned to this property determines whether the equation associated with a curve fit line is displayed. The curve fit line is drawn by the `setCurveFitType()` method.

Syntax:

```
setCurveFitEquationDisplay(boolean);
boolean = getCurveFitEquationDisplay();
```

where:

boolean

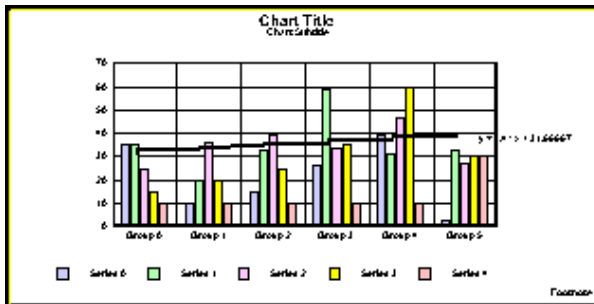
Can be one of the following:

true shows equation with the curve fit line. This value is the default.

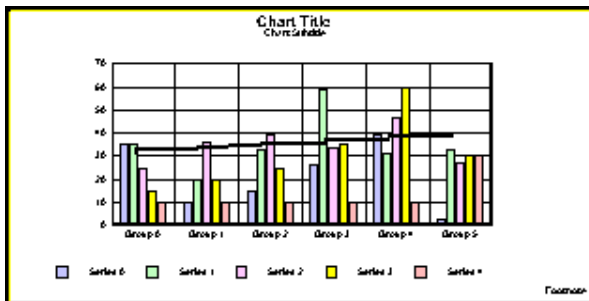
false does not show equation with the curve fit line.

Example:

```
setDepthAngle(0);
  setCurveFitEquationDisplay(true);
setCurveFitType(1,1);
setDepthRadius(0);
```



```
setCurveFitEquationDisplay(false);
```



See [CurveFitHighOrderFirst](#) on page 305, [setCurveFitType\(\)](#) on page 868.

CurveFitHighOrderFirst

When a curve fit equation is displayed with the CurveFitEquationDisplay property, this property determines the order of terms for equation text.

Syntax:

```
setCurveFitHighOrderFirst(boolean);  
boolean = getCurveFitHighOrderFirst();
```

where:

boolean

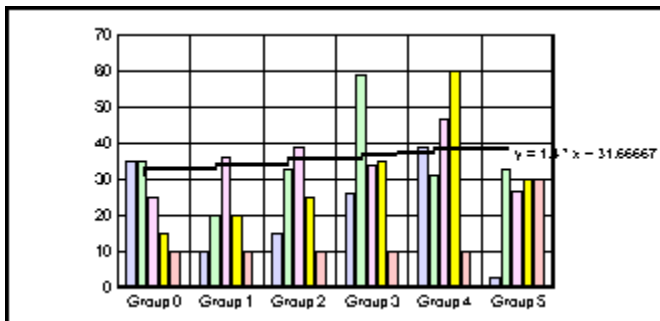
Can be one of the following:

true equals $ax + b$. This value is the default.

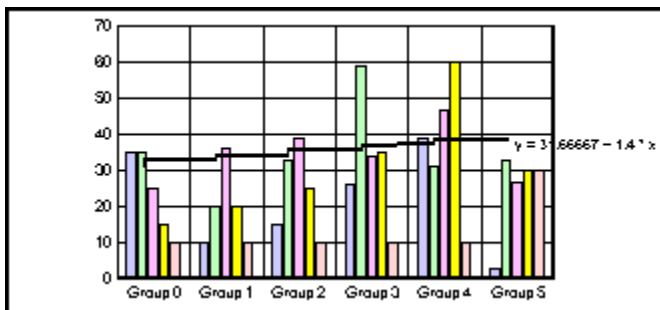
false equals $b + ax$.

Example:

```
setCurveFitType(1,1); setCurveFitHighOrderFirst(true);
```



```
setCurveFitHighOrderFirst(false);
```



See [CurveFitEquationDisplay](#) on page 303.

CurveFitNumSegments

This property determines the number of interpolated points to calculate to draw a curve fit line. The curve fit line is enabled or disabled with the `setCurveFitType()` method.

Syntax:

```
setCurveFitNumSegments(value);
value = getCurveFitNumSegments();
```

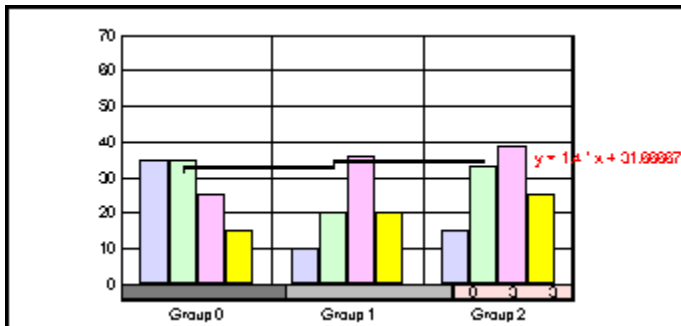
where:

value

Is any integer value. Larger values will draw a smoother line.

Example:

```
setDepthAngle(0);
setViewableSeries(4);
    setCurveFitNumSegments(100);
setCurveFitType(1,1);
setViewableGroups(3);
setDepthRadius(0);
```



See [setCurveFitType\(\)](#) on page 868.

DataItemsAlongSeries

This property determines whether data items are aligned parallel to the series (both rows or both columns). This property is only used with a graph type that requires more than one value per series (such as scatter, polar, or stock graphs). If any other graph type is selected, the property is ignored.

Syntax:

```
setDataItemsAlongSeries(boolean);
```

```
boolean = getDataItemsAlongSeries();
```

where:

`boolean`

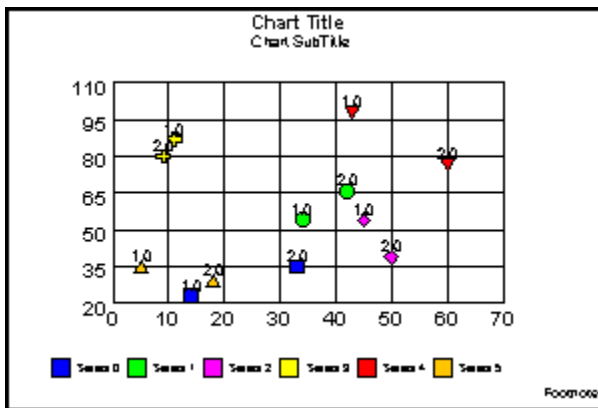
Can be one of the following:

`true` aligns data items parallel to series. This value is the default.

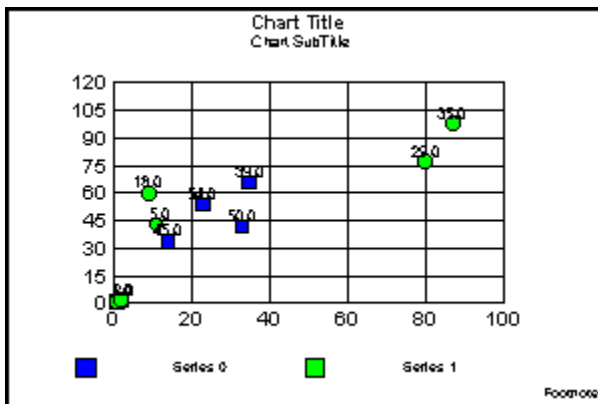
`false` aligns data items vertically within series.

Example:

```
setGraphType(63); setDataItemsAlongSeries(true);
```



```
setDataItemsAlongSeries(false);
```



DataLineThickness

When a 2.5D effect is applied to a vertical or horizontal line graph with DepthAngle and DepthRadius, the DataLineThickness property defines the width of the "fake 3D" lines in the graph.

Syntax:

```
setDataLineThickness(value);
```

```
value = getDataLineThickness();
```

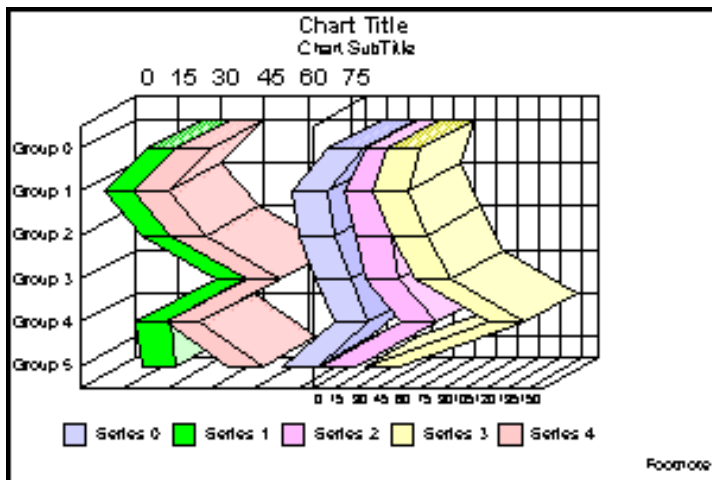
where:

value

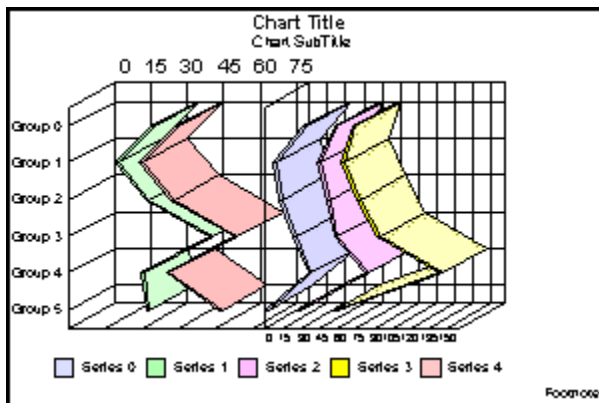
Is an integer value between 1 and 100. A value of one produces very thin lines, a value of 100 produces very thick 3D ribbons. 30 is the default value.

Example:

```
setGraphType(53); setDataLineThickness(90);
```



```
setDataLineThickness(10);
```



See [DepthAngle](#) on page 333, [DepthRadius](#) on page 334.

DataScrollerPresenceGroups

This property controls the presence of data scroller bars on the group axis. This property can be used for all 2D graphs except pies. It is not applicable for 3D graphs or 2D pie graphs. When the number of viewable groups is set to a value less than the number of groups in the data set, the scroller bar can be used to change the first group in the graph that is displayed (that is, it effectively changes the value assigned to ScrollOffsetGroup). The data scroller bar consists of two segments: 1) a segment that can be used to click and drag to the first group to be displayed and 2) a status bar that shows the number of the first group (for example, the ScrollOffsetGroup value), the total number of viewable groups, and the remaining number of groups in the data set. In the first segment, the darker, movable thumb can be dragged to change the group offset. In the second segment, the total of the three numbers indicates the total number of groups defined in the data set. For example, the status area of the scroll bar displays "2 3 1". These numbers indicate that ScrollOffsetGroup is "2", ViewableGroups is "3", there is "1" group in the data set that is not displayed, and there are a total of six groups defined in the data set. The user can click on these numbers and edit them with the arrow keys.

Syntax:

```
setDataScrollerPresenceGroups(value);
```

```
value = getDataScrollerPresenceGroups();
```

where:

value

Can be one of the following:

[0](#) to never display data scroller bar. 0 is the default value.

[1](#) to only display data scroller bar when ViewableGroups is greater than zero.

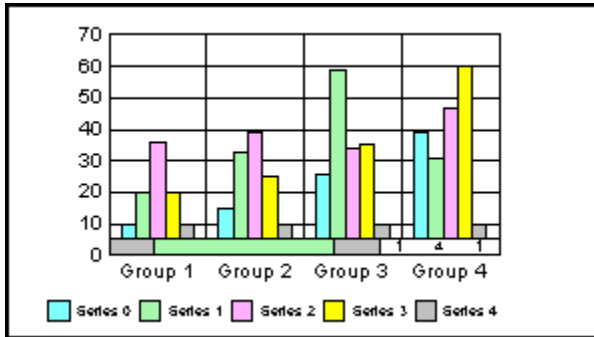
[2](#) to always display data scroller bar.

Example:

```

setDepthRadius(0);
  setDataScrollerPresenceGroups(2);
setViewableGroups(4);

```



See [DataScrollerPresenceSeries](#) on page 311, [ScrollOffsetGroup](#) on page 519, [ViewableGroups](#) on page 566.

DataScrollerPresenceSeries

This property controls the presence of data scroller bars on the series in a graph. This property can be used for all 2D graphs. It is not applicable for 3D graphs. When the number of viewable series is set to a value less than the number of series in the data set, the scroller bar can be used to change the first series in the graph that is displayed (that is, it effectively changes the value assigned to `ScrollOffsetSeries`). The data scroller bar consists of two segments: 1) a segment that can be used to click and drag to the first series to be displayed and 2) a status bar that shows the number of the first series (for example, the `ScrollOffsetSeries` value), the total number of viewable series, and the remaining number of series in the data set. In the first segment, the darker, movable thumb can be dragged to change the series offset. In the second segment, the total of the three numbers indicates the total number of series defined in the data set. For example, the status area of the scroll bar displays "2 3 1". These numbers indicate that `ScrollOffsetSeries` is "2", `ViewableSeries` is "3", there is "1" series in the data set that is not displayed, and there are a total of six series defined in the data set. The user can click on these numbers and edit them with the arrow keys.

Syntax:

```
setDataScrollerPresenceSeries(value);  
value = getDataScrollerPresenceSeries();
```

where:

value

Can be one of the following:

[0](#) to never display data scroller bar. 0 is the default value.

[1](#) to only display data scroller bar when ViewableSeries is greater than zero.

[2](#) to always display data scroller bar.

See [DataScrollerPresenceGroups](#) on page 310, [ScrollOffsetSeries](#) on page 520, [ViewableSeries](#) on page 568.

DataSortType

This property provides ability to sort data from high to low in Pareto charts only.

Syntax:

```
setDataSortType(value);  
value = getDataSortType();
```

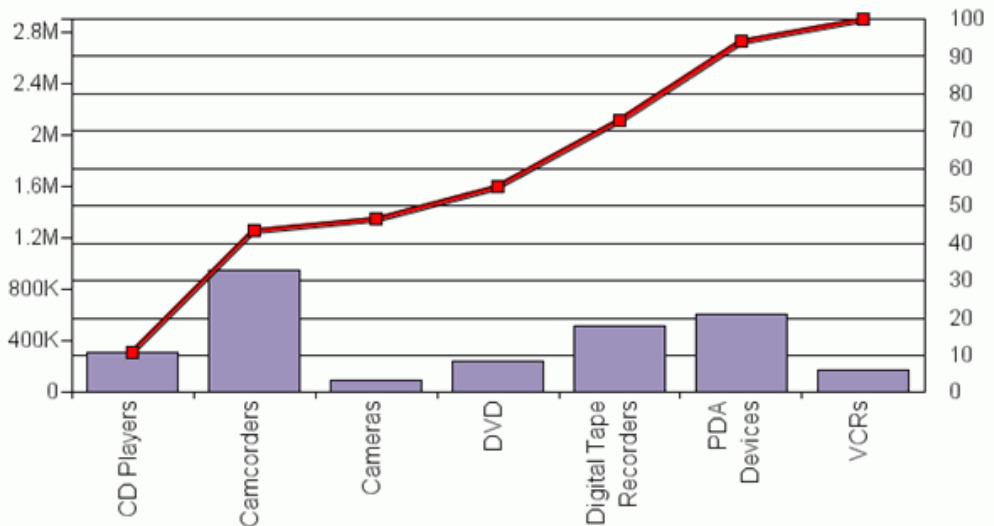
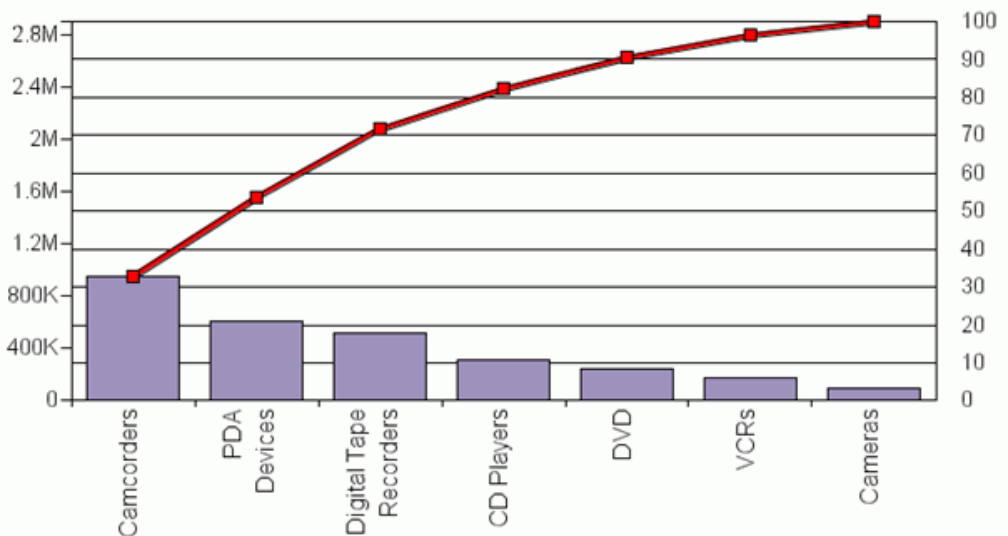
where:

value

Can be one of the following:

[0](#) to leave data unsorted. 0 is the default value.

[1](#) to sort data from high to low.

Example:`setDataSortType(0);``setDataSortType(1);`

DataTextAngleDefault

This property sets an angle from center-point that all data text is drawn from. The `setDataTextAngle()` method can override this property by applying an angle to an individual series or group intersection. This property is only used if `DataTextPosition` is set to zero.

Syntax:

```
setDataTextAngleDefault(value);  
value = getDataTextAngleDefault();
```

where:

value

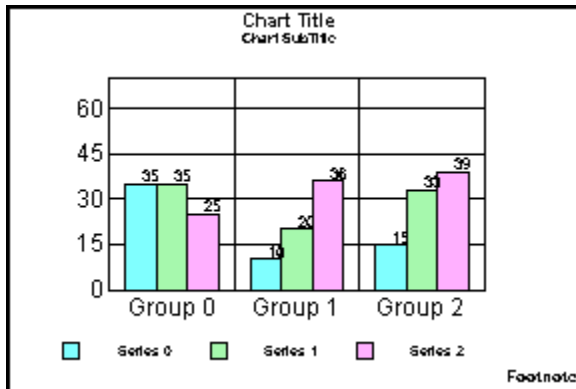
Is an integer value between 0 and 360. 90 is the default value.

Example:

```

setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextAngleDefault(45);

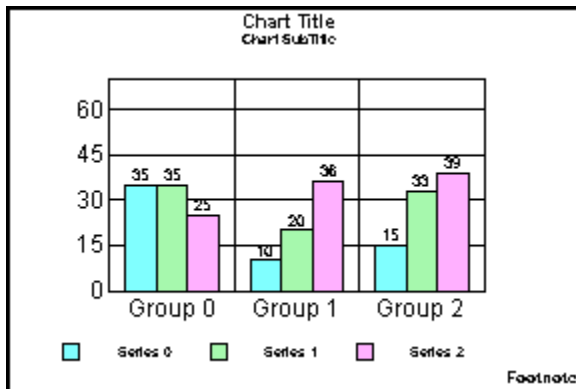
```



```

setDataTextAngleDefault(90);

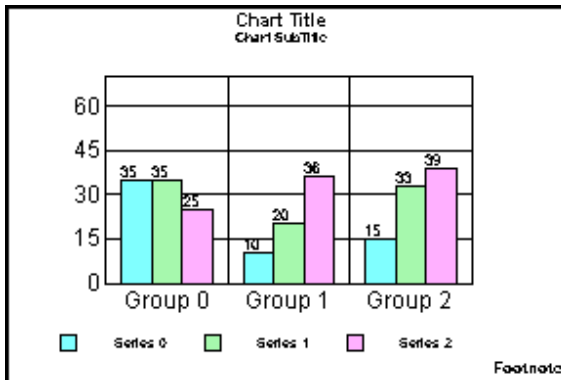
```



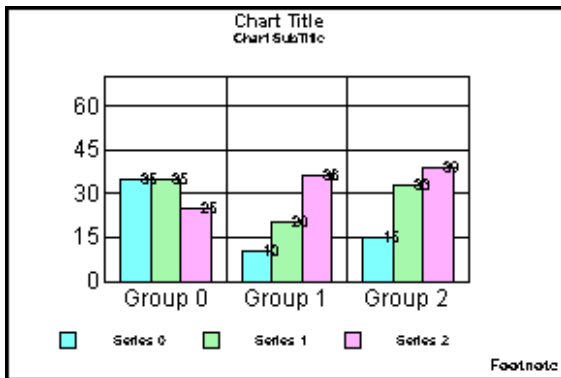
```

setDataTextAngleDefault(120);

```



`setDataTextAngleDefault(360);`



See [DataTextPosition](#) on page 322, [setDataTextAngle\(\)](#) on page 879.

DataTextDisplay

This property enables or disables the display of data values next to risers or markers in a graph. `DataTextFormat` can be used to define the format (for example: %, \$, and so on) of values displayed on the risers. You can use `DataTextPosition`, or `DataTextAngleDefault` and `DataTextRadiusDefault`, to define the location where data text values are displayed.

Syntax:

`setDataTextDisplay(boolean);`

`boolean = getDataTextDisplay();`

where:

boolean

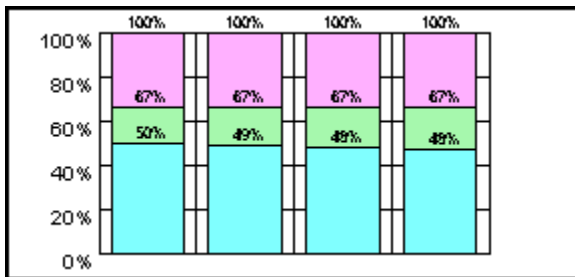
Can be one of the following:

true enables the display of data values.

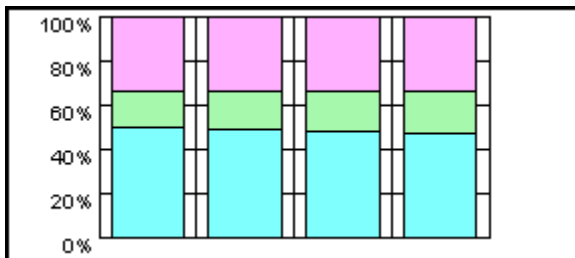
false disables the display of data values. This value is the default.

Example:

```
setDataTextDisplay(true);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataTextPosition(3);
setGraphType(23);
setDataScrollerPresenceGroups(0);
```



```
setDataTextDisplay(false);
```



Note: Use `PieFeelerTextDisplay` and `PieRingTotalDisplay` to display data values in a pie graph.

See [DataTextAngleDefault](#) on page 314, [DataTextFormat](#) on page 318, [DataTextPosition](#) on page 322, [DataTextRadiusDefault](#) on page 324.

DataTextFormat

When DataTextDisplay is set to true and data values are displayed next to markers and risers in a graph, this property is used to select one of the preset data text formats.

Syntax:

```
setDataTextFormat(value);
```

```
value = getDataTextFormat();
```

where:

value

Is an integer value between -1 and 21. 0 is the default value.

-1 to use the pattern defined by DataTextFormatPattern.

0/1 for # (for example, 123 = 123)

2 for #% (for example, 123 = 12,300%)

3 for #.##% (for example, 123 = 12,300.0%)

4 for #.###% (for example, 123 = 12,300.00%)

5 for \$#.## (for example, 123 = \$123.00)

6 for \$# (for example, 123 = \$123)

7 for #K (show K for values over 999) (for example, 1,234 = 1K)

8 for \$#K (show K for values over 999) (for example, 1,234 = \$1K)

9 for #M (show M for millions) (for example, 1,234,567 = 1M)

10 for \$#M (show M for millions) (for example, 1,234,567 = \$1M)

11 for #B (show B for billions) (for example, 1,234,567,891 = 1B)

12 for \$#B (show B for billions) (for example, 1,234,567,891 = \$1B)

13 for #T (show T for trillions) (for example, 1,234,567,891,234 = 1T)

14 for \$#T (show T for trillions) (for example, 1,234,567,891,234 = \$1T)

15 for number with thousands separators, no decimal places (for example, 1,234 = 1K)

16 for number with thousands separators, two decimal places (for example, 1,234 = 1.23K)

17 for general currency format for current Locale

18 for Short Date Format: MM/DD/YY (for example, 10/01/03)

19 for Medium Date Format: Mon. DD, YYYY (for example, Oct. 01, 2003)

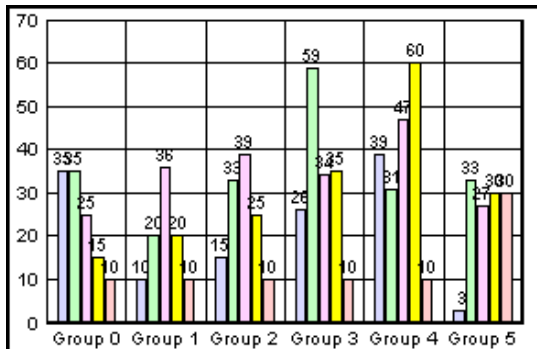
20 for Long Date Format: Month DD, YYYY (for example, October 01, 2003)

21 for Full Date Format: Day, Month DD, YYYY (for example, Wednesday, October 01, 2003)

See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setDataTextFormat(1);
setDataTextDisplay(true);
```



Note:

- ❑ Use `DataTextPosition` or `DataTextAngleDefault` and `DataTextRadiusDefault` to define the location where data text values are displayed.
- ❑ Use `PieFeelerTextFormat` and `PieFeelerTextFormatPattern` to format data text associated with feelers in a pie graph.
- ❑ Use `PieRingTotalFormat` and `PieRingTotalFormatPattern` to format the total value in the center of a ring pie graph.

See [DataTextAngleDefault](#) on page 314, [DataTextRadiusDefault](#) on page 324, [DataTextDisplay](#) on page 316, [DataTextFormatPattern](#) on page 320, [DataTextPosition](#) on page 322.

DataTextFormatPattern

When DataTextDisplay is set to true and DataTextFormat is set to minus one, this property is used to specify a standard Java number format pattern for data text values.

Syntax:

```
setDataTextFormatPattern(pattern);  
pattern = getDataTextFormatPattern();
```

where:

pattern

Is a pattern string in the following format:

```
pattern:= subpattern{;subpattern}  
subpattern:= {prefix}integer{.fraction}{suffix}  
prefix:= '\\u0000'..'\\uFFFFD' - specialCharacters  
suffix:= '\\u0000'..'\\uFFFFD' - specialCharacters  
integer:= '#'* '0'* '0'  
fraction:= '0'* '#'*
```

"#.#" is the default value.

Notation	For...
X*	0 or more instances of X.
(X Y)	Either X or Y.
X..Y	Any character from X up to Y, inclusive.
S - T	Characters in S, except those in T.

The first subpattern is for positive numbers. The second (optional) subpattern is for negative numbers. In both cases, a comma (,) can occur inside the integer portion. Here are the special characters used in the parts of the subpattern, with notes on their usage.

Notation	For...
0	A digit.
#	A digit, zero shows as absent.

Notation	For...
.	A period (.) is a placeholder for decimal separator.
,	A comma (,) is a placeholder for grouping separator.
;	A semicolon (;) separates formats.
-	A minus sign/dash (-) is the default negative prefix.
%	Divide by 100 and show as percentage.
x	Any other characters can be used in the prefix or suffix.
'	A single quote (') is used to quote special characters in a prefix or suffix.

If there is no explicit negative subpattern, a minus sign (-) is prefixed to the positive form (that is, "0.00" alone is equivalent to "0.00;-0.00"). Illegal formats, such as "#.#.#" or mixing '_' and '*' in the same format, will cause a ParseException. The ParseException string will show where the error occurred. The grouping separator is commonly used for thousands, but in some countries for ten-thousands. The interval is a constant number of digits between the grouping characters, such as 100,000,000 or 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "#####,####" == "##,####,####". This class only handles localized digits where the 10 digits are contiguous in Unicode, from 0 to 9.

Note:

- ❑ DataTextFormat must be set to minus one in order for this property to have any effect on the graph.
- ❑ Use DataTextPosition or DataTextAngleDefault and DataTextRadiusDefault to define the location where data text values are displayed.
- ❑ Use PieFeelerTextFormat and PieFeelerTextFormatPattern to format data text associated with feelers in a pie graph.
- ❑ Use PieRingTotalFormat and PieRingTotalFormatPattern to format the total value in the center of a ring pie graph.

See [DataTextAngleDefault](#) on page 314, [DataTextRadiusDefault](#) on page 324, [DataTextDisplay](#) on page 316, [DataTextFormat](#) on page 318, [DataTextPosition](#) on page 322.

DataTextPosition

When the DataTextDisplay property is enabled, this property defines the position where data text is displayed next to risers in a graph.

Syntax:

```
setDataTextPosition(value);  
value = getDataTextPosition();
```

where:

value

Can be one of the following:

[0](#) for custom, use angle and radius set by DataTextAngleDefault and DataTextRadiusDefault.

[1](#) for above risers. 1 is the default value.

[2](#) for center on riser top edge.

[3](#) for below riser top edge.

[4](#) for center of riser.

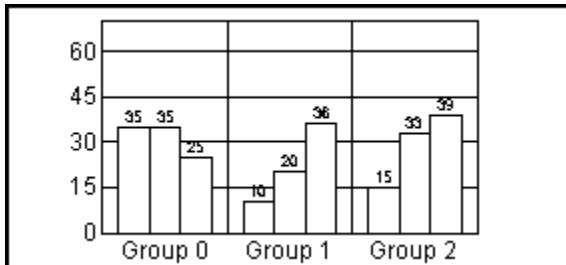
[5](#) for center of riser.

Example:

```

setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setDataTextDisplay(true);
  setDataTextPosition(1);

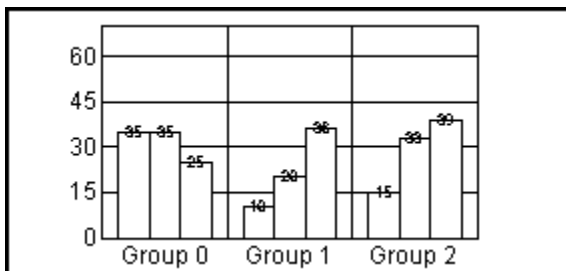
```



```

setDataTextPosition(2);

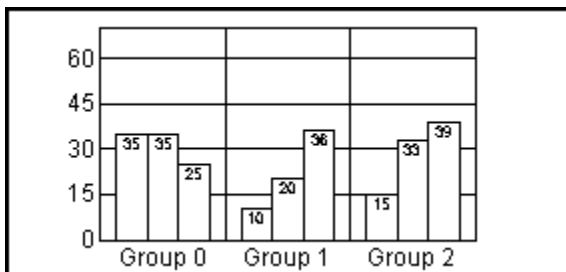
```



```

setDataTextPosition(3);

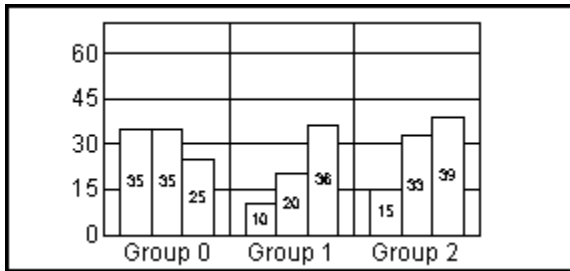
```



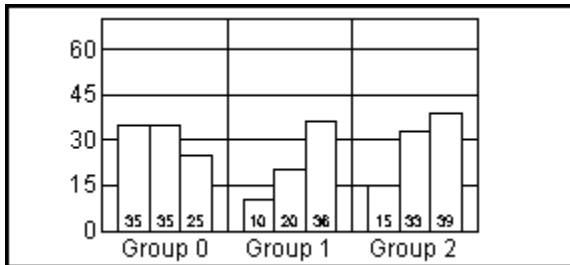
```

setDataTextPosition(4);

```



`setDataTextPosition(5);`



Note: If this property is set to zero, the `DataTextAngleDefault` and `DataTextRadiusDefault` properties and the `setDataTextAngle()` and `setDataTextRadius()` methods control the location of data text.

See [DataTextDisplay](#) on page 316, [DataTextAngleDefault](#) on page 314, [DataTextRadiusDefault](#) on page 324.

DataTextRadiusDefault

This property sets the radius that a data text value will be drawn out from the center of a graph riser by default. The `setDataTextRadius()` method can override this value by setting a specific series or group intersection format for the default data text position.

Syntax:

`setDataTextRadiusDefault(value);`

`value = getDataTextRadiusDefault();`

where:

value

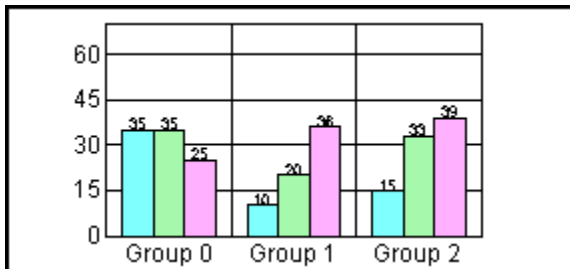
Is an integer value between 0 and 360. 20 is the default value.

Example:

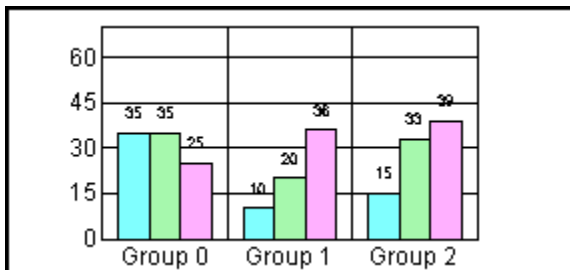
```

setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextAngleDefault(90);  setDataTextRadiusDefault(10);

```



```
setDataTextRadiusDefault(40);
```



Note: This property will be ignored if DataTextPosition is not set to zero.

See [DataTextAngleDefault](#) on page 314, [DataTextDisplay](#) on page 316, [DataTextPosition](#) on page 322.

DataTextStackedTotalOnTop()

This property controls the display of cumulative values on top of the stacked riser. This is different from regular Data Text Display on Stacked charts where cumulative totals appear on each segment of the riser. This option is available for bar, line, and area stacked charts.

Syntax:

```
setDisplay(getDataTextStackedTotalOnTop(), boolean);
```

where:

boolean

Is true or false. When boolean is true, cumulative values appear on the risers in the graph output. When it is false, cumulative values do not appear.

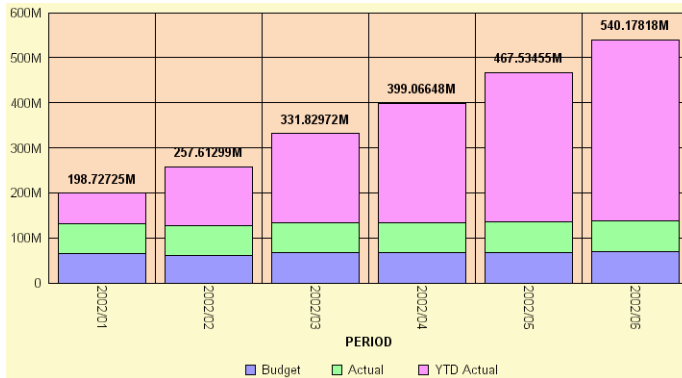
Note: The position of StackedTotal data text is on the bottom of the stack when the sum is negative and on the top of the stack when the sum is positive.

Example:

The following example illustrates a stacked bar chart where cumulative values appear on the risers.

```
GRAPH FILE CENTSTMT
SUM  CENTSTMT.CONSOL.BUDGET_AMT
CENTSTMT.CONSOL.ACTUAL_AMT
CENTSTMT.CONSOL.ACTUAL_YTD
ACROSS CENTSTMT.CONSOL.PERIOD
ON GRAPH SET LOOKGRAPH VBRSTK1
ON GRAPH SET 3D OFF
ON GRAPH SET GRID ON
ON GRAPH SET GRAPHEDIT SERVER
ON GRAPH SET VZERO ON
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
setFillColor(getFrame(),new Color(255,218,185));
setDisplay(getDataTextStackedTotalOnTop(), true);
setPlace(true);
*END
TYPE=REPORT,BACKCOLOR=rgb(255 250 205), $
ENDSTYLE
END
```

The output is:



Format Option for the Stack Total Value

When generating a stacked graph, you can set the `setUseDataTextStackTotalOnTopFormat(Boolean);` API to `true`. This enables you to format the stack total value using the `setTextFormatPreset(getDataTextStackedTotalOnTop(), int);` API call.

For example, the following calls set the format option to `true` and then format the stack total using preset format 6:

```
setUseDataTextStackTotalOnTopFormat(true);
setTextFormatPreset(getDataTextStackedTotalOnTop(), 6);
```

Syntax: How to Format the Stack Total Value

```
setUseDataTextStackTotalOnTopFormat(Boolean);
```

where:

Boolean

Can have the following values:

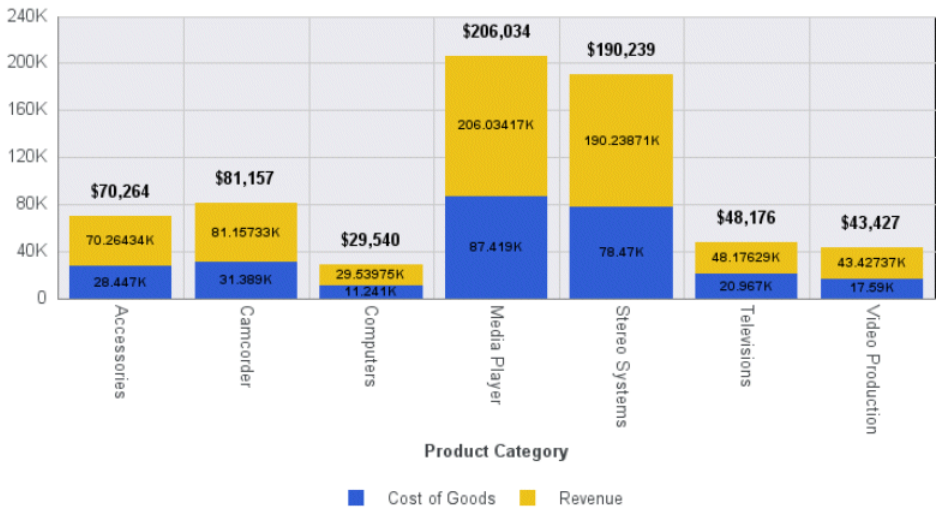
- ☐ **true**, which enables formatting the stack total value using the `setTextFormatPreset(getDataTextStackedTotalOnTop(), int);` API call.
- ☐ **false**, which disables reformatting the total stack value. This is the default value.

Example: Formatting the Stack Total Value

The following request enables the stacked total value format option, and then formats the stack total using preset format 6 (\$#), which prefixes the number with a dollar sign (\$):

```
GRAPH FILE WF_RETAIL_LITE
SUM COGS_US REVENUE_US
BY PRODUCT_CATEGORY
ON GRAPH PCHOLD FORMAT PNG
ON GRAPH SET LOOKGRAPH VBRSTK1
ON GRAPH SET STYLE *
INCLUDE=ENIADefault_combine.sty,$
*GRAPH_SCRIPT
setPlace(true);
setDataTextDisplay(true);
setDataTextPosition(4);
setDisplay(getDataTextStackedTotalOnTop(),true);
  setUseDataTextStackTotalOnTopFormat(true);
setTextFormatPreset(getDataTextStackedTotalOnTop(), 6);
*END
ENDSTYLE
END
```

The output is:



DataTextTemplateBubble

This is a global string property that can be used to define a macro-based format for data text labels in a bubble graph. This property can be used to customize the data text labels and insert macros for the actual data any place in a defined string. The DataTextDisplay property must be set to true in order for the data text labels to appear in the graph.

Syntax:

```
setDataTextTemplateBubble("string");  
string = getDataTextTemplateBubble();
```

where:

string

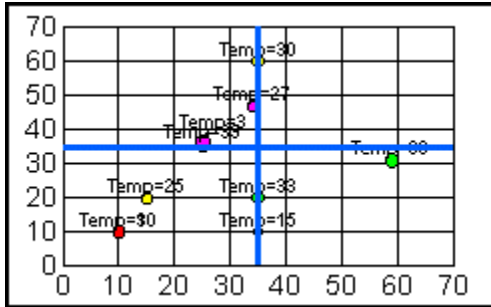
Can be any printable characters which may or may not include this macro:

```
X_VALUE_MACRO = "[X]";  
Y_VALUE_MACRO = "[Y]";  
X_Y_VALUE_MACRO = "[X], [Y]";  
LABEL_SERIES_MACRO = "[SL]";  
SIZE_VALUE_MACRO = "[SIZE]"
```

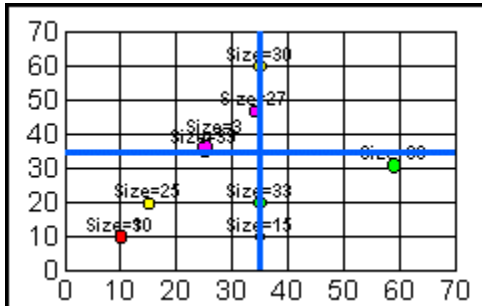
[SIZE] is the default value.

Example:

```
setDataTextDisplay(true);
setGraphType(89);  setDataTextTemplateBubble("Temp=[SIZE]");
```



```
setDataTextTemplateBubble("Size=[SIZE]");
```



See [DataTextDisplay](#) on page 316.

DataTextTemplateHiLo

This is a global string property that can be used to define a macro-based format for data text labels in a stock graph. The default is a null string. If the default is used, a "[CLOSE]" value is used if the Graph Type has a close data item, and "[HIGH]" is used otherwise. This property can be used to customize the data text labels and insert macros for the actual data any place in a defined string. The DataTextDisplay property must be set to true in order for the data text labels to display in the graph.

Syntax:

```
setDataTextTemplateHiLo("string");
"string" = getDataTextTemplateHiLo();
```

where:

string

Can be any printable characters which may or may not include these macros:

OPEN_VALUE_MACRO = "[OPEN]"

HI_VALUE_MACRO = "[HIGH]"

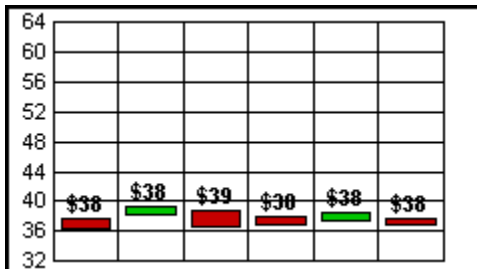
LOW_VALUE_MACRO = "[LOW]"

CLOSE_VALUE_MACRO = "[CLOSE]"

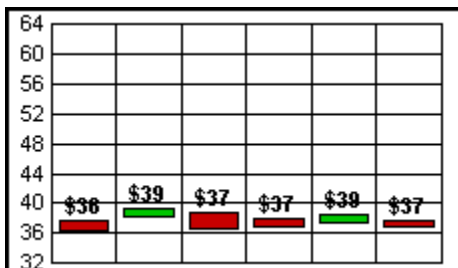
A null string is the default value.

Example:

```
setDataTextDisplay(true);  setDataTextTemplateHiLo("[OPEN]");
setGraphType(72);
setDataTextFormat(6);
setViewableGroupsStock(6);
setDataScrollerPresenceGroups(0);
```



```
setDataTextTemplateHiLo("[CLOSE]");
```



See [DataTextDisplay](#) on page 316.

DataTextTemplateScatter

This is a global string property that can be used to define a macro-based format for data text labels in a scatter graph. Currently defined macros are "[X]" and "[Y]". This property can be used to customize the data text labels and insert macros for the actual data any place in a defined string. The DataTextDisplay property must be set to true in order for the data text labels to display in the graph.

Syntax:

```
setDataTextTemplateScatter("string");  
"string" = getDataTextTemplateScatter();
```

where:

string

Can be any printable characters which may or may not include these macros:

```
X_VALUE_MACRO = "[X]"
```

```
Y_VALUE_MACRO = "[Y]"
```

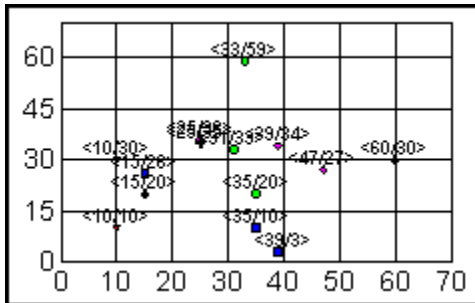
```
X_Y_VALUE_MACRO = "[X], [Y]"
```

```
LABEL_SERIES_MACRO = "[SL]";
```

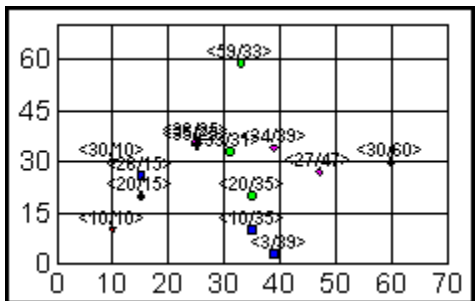
[X],[Y] is the default value.

Example:

```
setDataTextDisplay(true);  setDataTextTemplateScatter("<[X]/[Y]>");
setGraphType(61);
```



```
setDataTextDisplay(true);  setDataTextTemplateScatter("<[Y]/[X]>");
setGraphType(61);
```



See [DataTextDisplay](#) on page 316.

DepthAngle

This property sets the angle of a line from the front of the graph frame to the back.

Syntax:

```
setDepthAngle(value);
value = getDepthAngle();
```

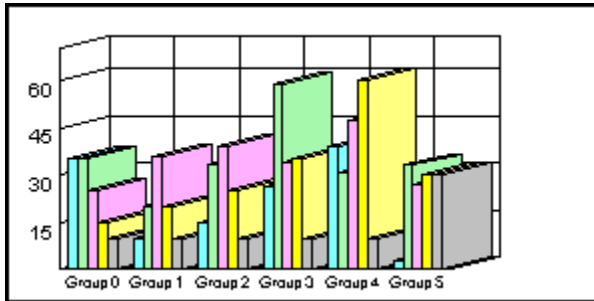
where:

value

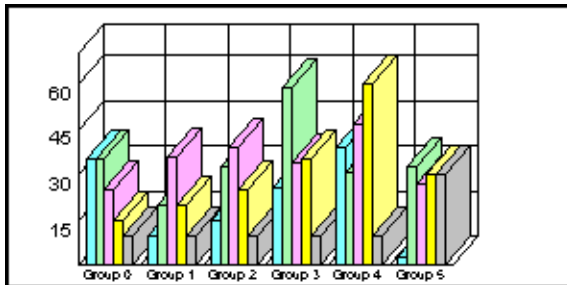
Is an integer value, in degrees, between 0 and 180. 45 is the default value.

Example:

```
setDepthAngle(20);
```



```
setDepthAngle(60);
```

**Note:**

- ❑ This property applies to bar, line, and area graphs only. It will be ignored if any other graph type is selected.
- ❑ The DepthRadius property must be greater than zero in order for this property to be properly applied to the graph.

See [DepthRadius](#) on page 334.

DepthRadius

When graphs are drawn with a 2.5D effect, this property sets how far out the extruded frame will be extended. Small values produce a very narrow graph, larger values produce a thicker graph.

Syntax:

```
setDepthRadius(value);
```

```
value = getDepthRadius();
```

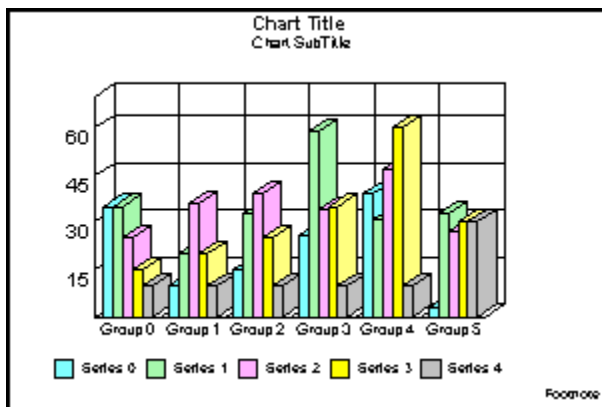
where:

value

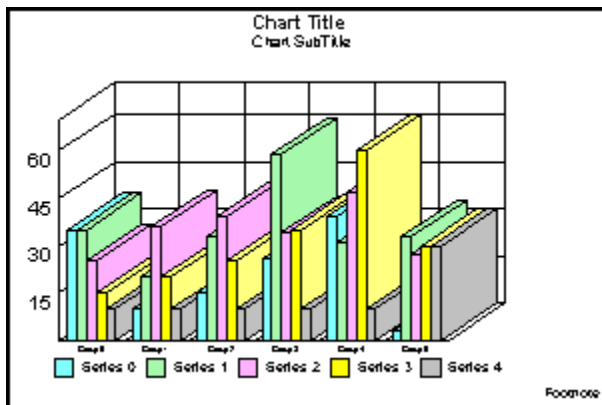
Is an integer value between 0 and 100. 40 is the default value.

Example:

`setDepthRadius(20);`



`setDepthRadius(60);`



Note: This property applies to bar, line, and area graphs only. It is ignored if any other graph type is selected.

See [DepthAngle](#) on page 333.

Display3DFloor

This property enables or disables drawing of the cube floor in a 3D graph.

Syntax:

```
setDisplay3DFloor(boolean);
```

```
boolean = getDisplay3DFloor();
```

where:

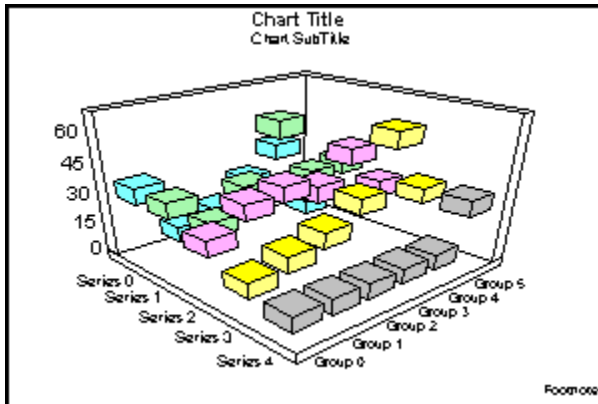
boolean

Can be one of the following:

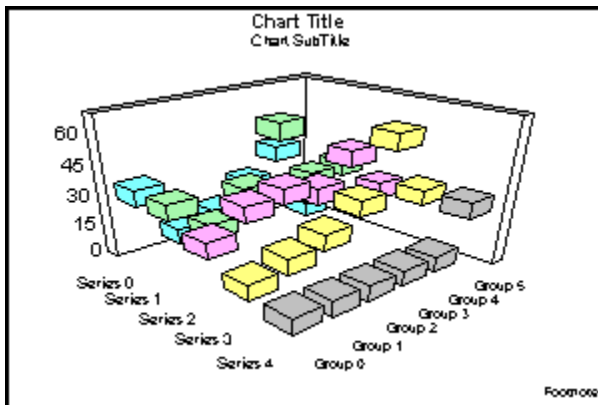
- ☐ **true** draws cube floor. This value is the default.
- ☐ **false** does not draw cube floor.

Example:

```
setGraphType(4);
setDisplay3DFloor(true);
```



```
setDisplay3DFloor(false);
```



Note: This property only applies to 3D graph types. It is ignored if any other graph type is selected.

See [Display3DLeftWall](#) on page 338, [Display3DRightWall](#) on page 340.

Display3DLeftWall

This property enables or disables drawing of the cube left wall in a 3D graph.

Syntax:

```
setDisplay3DLeftWall(boolean);  
boolean = getDisplay3DLeftWall();
```

where:

boolean

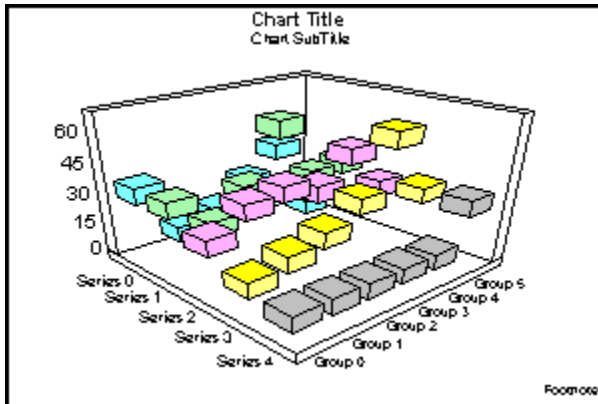
Can be one of the following:

[true](#) draws left wall of 3D cube. This value is the default.

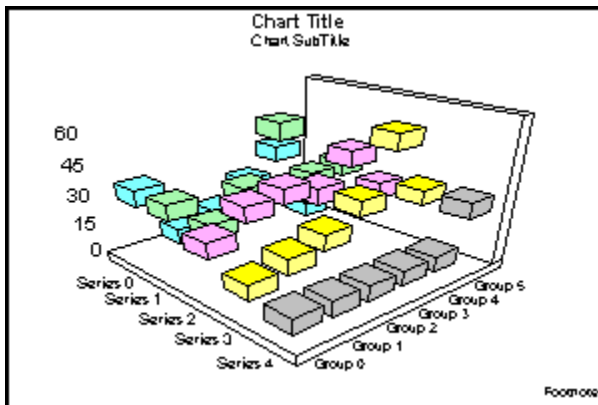
[false](#) does not draw left wall of 3D cube floor.

Example:

```
setDisplay3DLeftWall(true);
setGraphType(4);
```



```
setDisplay3DLeftWall(false);
```



Note: This property only applies to 3D graph types. It is ignored if any other graph type is selected.

See [Display3DFloor](#) on page 336, [Display3DRightWall](#) on page 340.

Display3DRightWall

This property enables or disables drawing of the cube right wall in a 3D graph.

Syntax:

```
setDisplay3DRightWall(boolean);
```

```
boolean = getDisplay3DRightWall();
```

where:

boolean

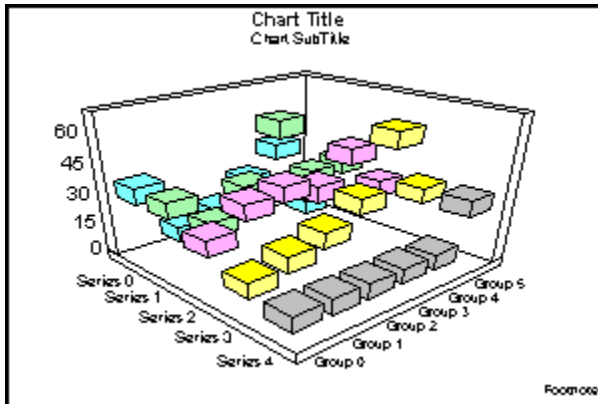
Can be one of the following:

[true](#) draws right wall of 3D cube. This value is the default.

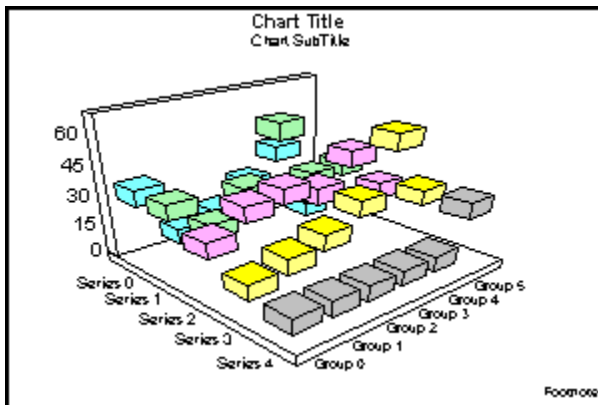
[false](#) does not draw right wall of 3D cube floor.

Example:

```
setDisplay3DRightWall(true);
setGraphType(4);
```



```
setDisplay3DRightWall(false);
```



Note: This property only applies to 3D graph types. It is ignored if any other graph type is selected.

See [Display3DLeftWall](#) on page 338, [Display3DFloor](#) on page 336.

DisplayBarAsPictograph

This property enables or disables the display of bars as pictographs in a flat 2D graph.

Syntax:

```
setDisplayBarAsPictograph(boolean);  
boolean = getDisplayBarAsPictograph();
```

where:

boolean

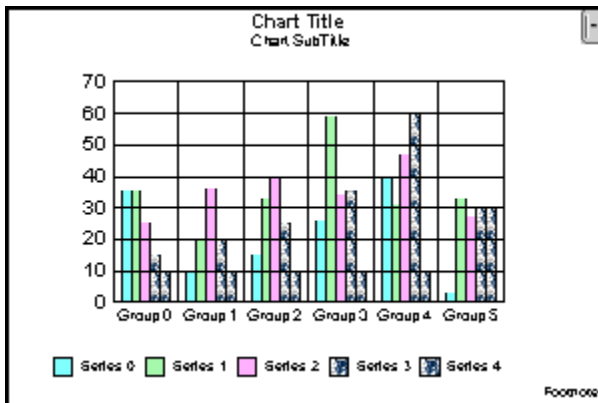
Can be one of the following:

[true](#) displays bars as pictographs. The picture (.GIF image) identified by the `setTextureURL()` method is applied to the bar. The `setTextureURL()` method must be used to identify a valid .GIF image. If a texture has not been identified with `setTextureURL()`, this property may cause jagged lines to display on the bars. The `setFillType()` method should also be set to identify textures (for example, `setFillType(3);`) as the fill type.

[false](#) does not display bars as pictographs. This value is the default.

Example:

```
setFillType(getSeries(3),3);  
setFillType(getSeries(4),3);  
setTextureDisplayMode(0);  
setTextureURL(getSeries(3),"/images/tdg/Azul_Granite.gif");  
setTextureURL(getSeries(4),"/images/tdg/Azul_Granite.gif");  
setDisplayBarAsPictograph(true);
```



DisplayFunnelGroup

In a Funnel Graph (GraphType = 106), the value of this property defines the group to be drawn in the graph. Funnel graphs are similar to pie graphs except they show only a single group. The series in the selected group are stacked from top to bottom in the funnel.

Syntax:

```
setDisplayFunnelGroup(value);
value = getDisplayFunnelGroup();
```

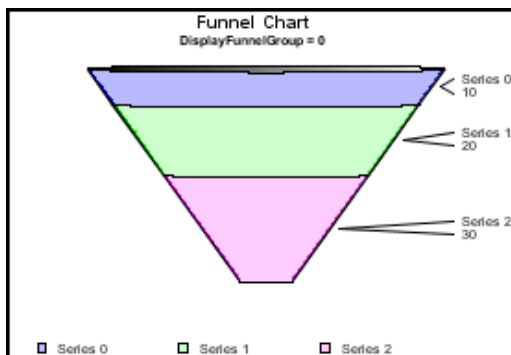
where:

value

Is an integer value between 0 and the number of groups in the data set. 0 is the default value.

Example:

```
setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
    setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleString("DisplayFunnelGroup = 0");
setTransparentBorderColor(getFrame(), true);
```



See [FunnelLabelValueLocation](#) on page 363, [PieDepth](#) on page 474, [PieRotate](#) on page 488, [PieTilt](#) on page 493.

DisplayParetoSeries

This property selects the series to be highlighted in a Pareto graph.

Syntax:

```
setDisplayParetoSeries(value);
```

```
value = getParetoSeries();
```

where:

value

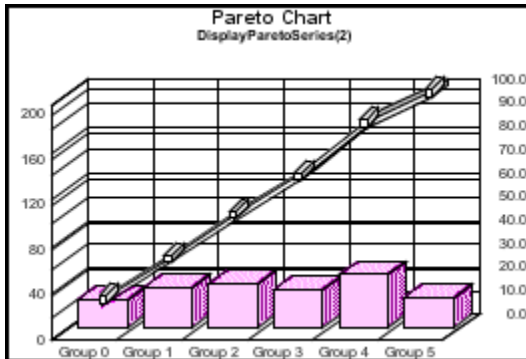
Is an integer value between 0 and the number of series in the data set. 0 is the default value.

Example:

```

setFootnoteDisplay(false);
  setDisplayParetoSeries(2);
setGraphType(102);
setTitleString("Pareto Chart");
setSubtitleString("DisplayParetoSeries(2)");

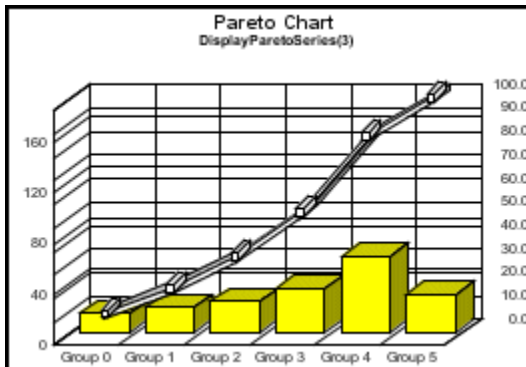
```



```

setFootnoteDisplay(false);
  setDisplayParetoSeries(3);
setGraphType(102);
setTitleString("Pareto Chart");
setSubtitleString("DisplayParetoSeries(3)");

```



DocumentHeightInTwips

This property defines the document height in TWIPs (1/20th point or 1/100th millimeter). The document size set by this property and DocumentWidthInTwips is used as a reference to calculate the size of text using absolute font sizes. If no document size is specified, the size of the graph in pixels and the screen resolution are used for conversion (from pixels to points). If no document size is specified (no screen resolution available), the conversion from pixels to points is simply 1 to 1.

Syntax:

```
setDocumentHeightInTwips(value);  
value = getDocumentHeightInTwips();
```

where:

value

Is an integer between 0 and 10000. 0 (use the size in pixels at the given resolution) is the default value.

Note: If a non-zero value is used, Perspective will also put the size in pixels in the SVG viewport.

See [DocumentWidthInTwips](#) on page 346, [getDocumentHeightInPoints\(\)](#) on page 738, [getDocumentWidthInPoints\(\)](#) on page 739, [setFontSizeAbsolute\(\)](#) on page 896, [setFontSizeInPoints\(\)](#) on page 897.

DocumentWidthInTwips

This property defines the document width in TWIPs (1/20th point or 1/100th millimeter). The document size set by this property and DocumentHeightInTwips is used as a reference to calculate the size of text using absolute font sizes. If no document size is specified, the size of the graph in pixels and the screen resolution are used for conversion (from pixels to points). If no document size is specified (no screen resolution available), the conversion from pixels to points is simply 1 to 1.

Syntax:

```
setDocumentWidthInTwips( value );  
value = getDocumentWidthInTwips();
```

where:

value

Is an integer between 0 and 10000. 0 is the default value.

Note: If a non-zero value is used, Perspective will also put the size in pixels in the SVG viewport.

See [DocumentHeightInTwips](#) on page 346, [getDocumentHeightInPoints\(\)](#) on page 738, [getDocumentWidthInPoints\(\)](#) on page 739, [setFontSizeAbsolute\(\)](#) on page 896, [setFontSizeInPoints\(\)](#) on page 897.

DrawAreaMarkers

This property controls the display of markers on an area chart. By default, the area chart is drawn without markers.

Syntax:

```
setDrawAreaMarkers (boolean);
```

```
boolean = getDrawAreaMarkers();
```

where:

boolean

Can be one of the following:

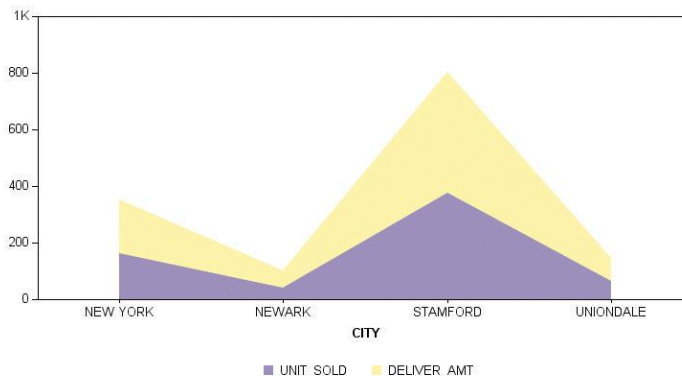
[true](#) draws markers for area charts.

[false](#) does not draw markers for area charts. This value is the default.

Example:

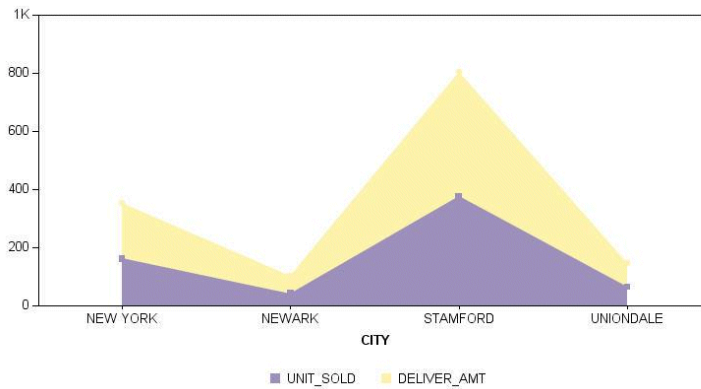
With `setDrawAreaMarkers(false)`, the markers are not drawn on the area chart:

```
setDrawAreaMarkers(false);
```



With `setDrawAreaMarkers(true)`, the markers are drawn on the area chart:

```
setDrawAreaMarkers(true);
```



DrawCompleteAreaRiser

This property controls the display of divider lines that serve as visual separators between each group in area charts. By default, the area chart is drawn as a complete area and the divider lines are not visible.

Syntax:

```
setDrawCompleteAreaRiser (boolean);boolean = getDrawCompleteAreaRiser ();
```

where:

boolean

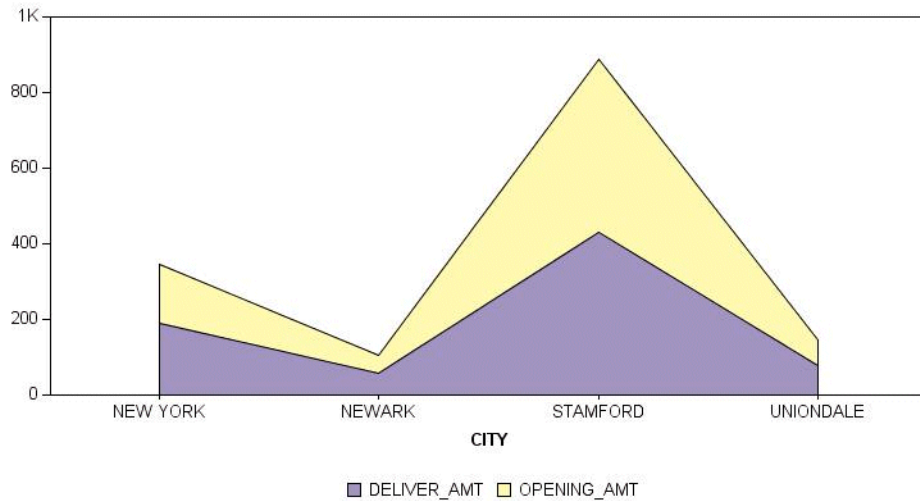
Can be one of the following:

[`true`](#) disables divider lines. This value is the default.

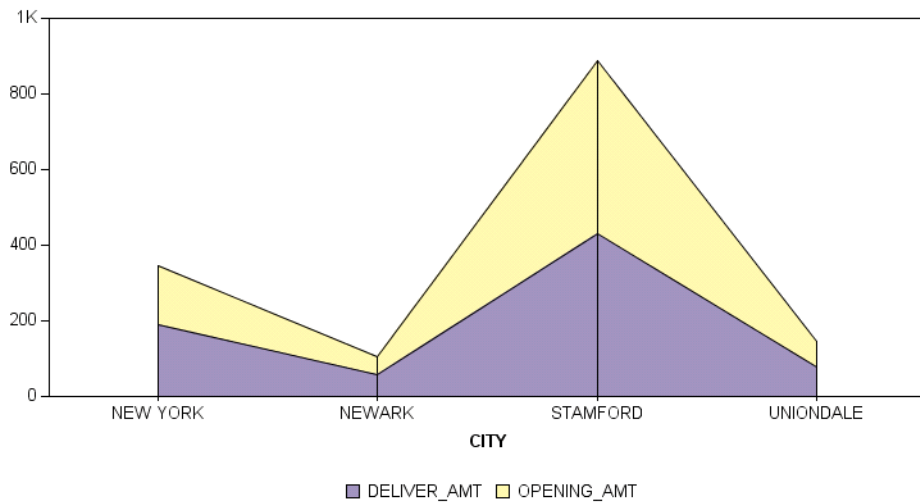
[`false`](#) enables divider lines.

Example:

```
setDrawCompleteAreaRiser(true);
```



```
setDrawCompleteAreaRiser(false);
```



DualAxisLineDisplay

This property enables or disables the display of a line that separates the two halves of a dual axes graph. The graph type must be a bipolar graph or a stock graph with volume in order to see the effect of this property.

Syntax:

```
setDualAxisLineDisplay(boolean);  
boolean = getDualAxisLineDisplay();
```

where:

boolean

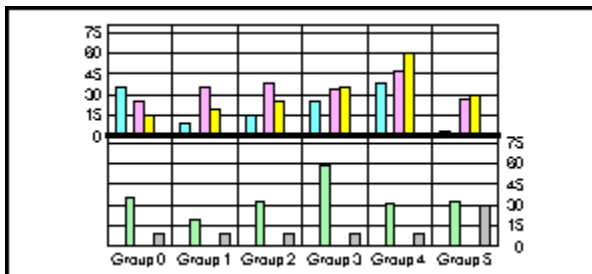
Can be one of the following:

[true](#) displays the dual axes line. This value is the default.

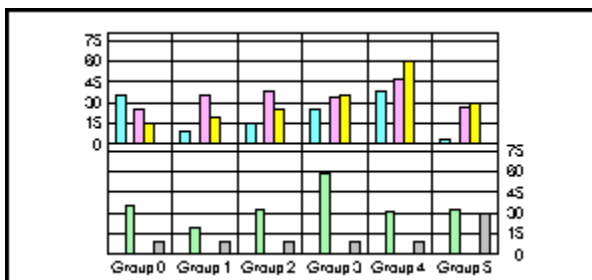
[false](#) does not display the dual axes line.

Example:

```
setDepthRadius(0);  
    setDualAxisLineDisplay(true);  
setGraphType(21);
```



```
setDualAxisLineDisplay(false);
```



Note: For bar, line, and area graphs with DepthRadius set to any value greater than zero, a BiPolar plane displays instead of a solid line.

See [DualAxisSplitPosition](#) on page 351.

DualAxisSplitPosition

This property sets the position within the graph frame where the Dual-Y split position is created.

Syntax:

```
setDualAxisSplitPosition(value);  
value = getDualAxisSplitPosition();
```

where:

value

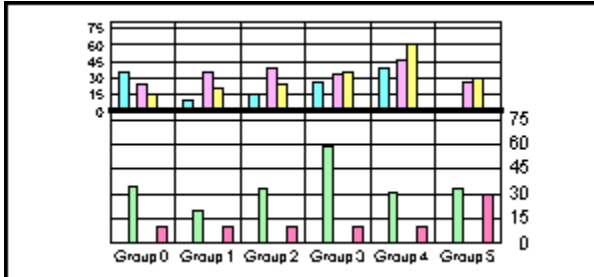
Is an integer value between 10 and 90. A lower number will move the split position closer to the bottom of the graph; a higher value will move it closer to the top of the graph. 50 is the default value, exactly in the middle.

Example:

```

setDepthRadius(0);
  setDualAxisSplitPosition(60);
setGraphType(21);
setLineWidth(getFrameSeparator(),4);

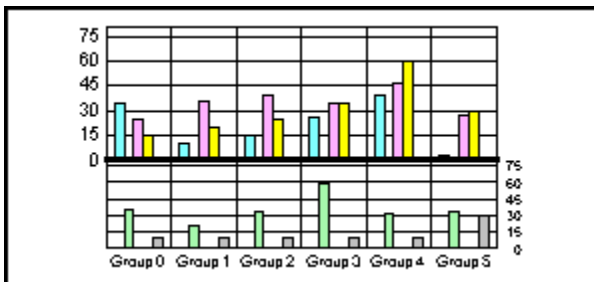
```



```

setDualAxisSplitPosition(40);

```



See [DualAxisLineDisplay](#) on page 350.

ExactColorByHeight

Enables or disables exact coloring by height in spectral maps and 3D surface graphs.

Syntax:

```

setExactColorByHeight(boolean);
boolean = setExactColorByHeight();

```

where:

boolean

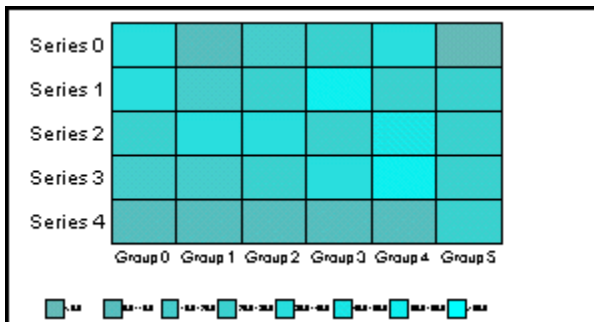
Can be one of the following:

[true](#) uses the actual data value to get a color from the "color-by-height gradient." This value is the default.

`false` uses only the colors on the spectral legend, and the data value is compared to see which of those "buckets" it falls into.

Example:

```
setExactColorByHeight(false);
setGradientPinLeftColor(getColorByHeight(),
    new Color(106,189,187),0);
setGradientPinLeftColor(getColorByHeight(),
    new Color(0,255,255),1);
setGradientPinRightColor(getColorByHeight(),
    new Color(0,255,255),1);
setGradientPinRightColor(getColorByHeight(),
    new Color(106,189,187),0);
setGraphType(87);
```



See [ColorMode](#) on page 260.

ExtendToFrameEdge

In area graphs, line graphs, and combination graphs that do not include bars, this property enables or disables extension of the risers to the graph frame edge.

Syntax:

```
setExtendToFrameEdge(boolean);
boolean = setExtendToFrameEdge();
```

where:

boolean

Can be one of the following:

`true` extends risers to frame edge.

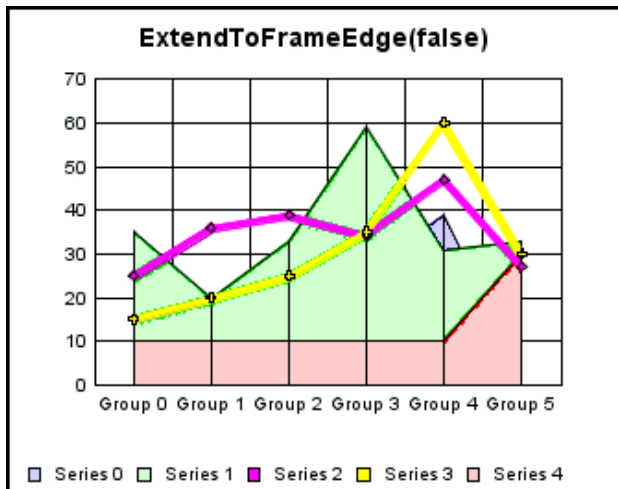
`false` does not extend risers to frame edge. This value is the default.

Example:

```

setDepthRadius(0);
setDepthAngle(0);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setTitleString("ExtendToFrameEdge(false)");
setSeriesType(getSeries(2),2);
setSeriesType(getSeries(3),2);
setSeriesType(getSeries(0),3);
setSeriesType(getSeries(4),3);
setSeriesType(getSeries(1),3);  setExtendToFrameEdge(false);

```

**FillMissingData**

When data points are missing from a data set, this property defines how the missing data is handled.

Syntax:

```

setFillMissingData(value);
value = getFillMissingData();

```

where:

value

Can be one of the following:

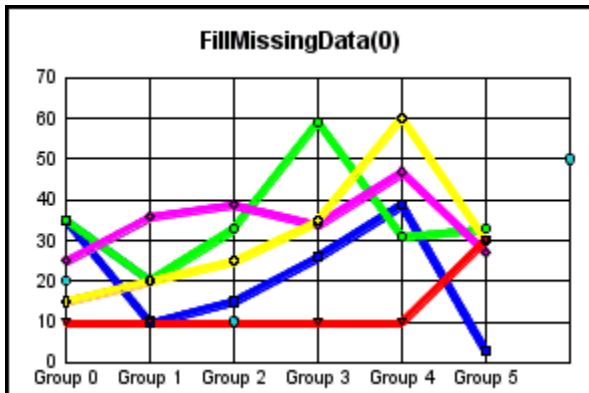
[0](#) skips missing data. 0 is the default value.

[1](#) sets missing data to zero.

2 interpolates.

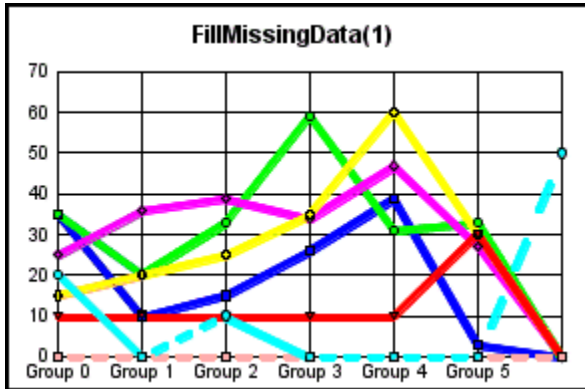
Example:

```
setGraphType(41);
setLegendDisplay(false);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setData(7,0,20.0);
setData(7,2,10.0);
setData(7,6,50.0);
setDataRangeToExtent();
setDepthRadius(0);
setFillMissingData(0);
setTitleString("FillMissingData(0)");
```



```
setGraphType(41);
setLegendDisplay(false);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setData(7,0,20.0);
setData(7,2,10.0);
setData(7,6,50.0);
setDataRangeToExtent();
setDepthRadius(0);
setFillMissingData(1);
```

```
setTitleString("FillMissingData(1)");
```



See [NumMissingDataSegments](#) on page 418.

FootnoteDisplay

This property enables or disables drawing of the footnote string that is defined by the FootnoteString property.

Syntax:

```
setFootnoteDisplay(boolean);
```

```
boolean = getFootnoteDisplay();
```

where:

boolean

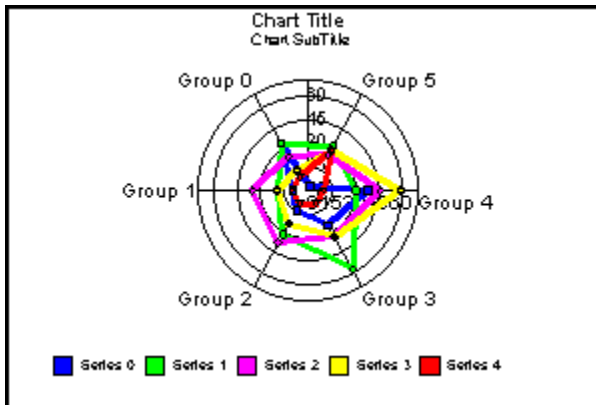
Can be one of the following:

[true](#) draws footnote string. This value is the default.

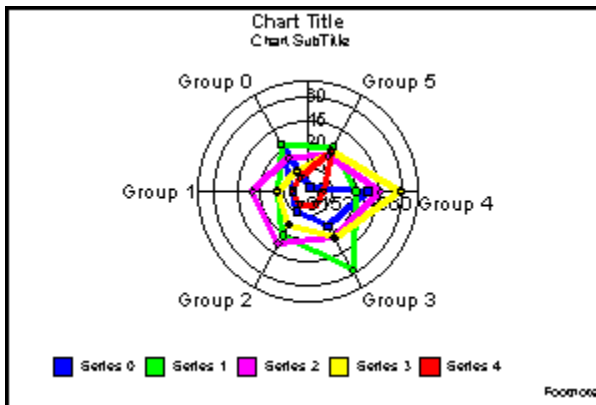
[false](#) does not draw footnote string.

Example:

```
setGraphType(69);  setFootnoteDisplay(false);
```



```
setFootnoteDisplay(true);
```



See [FootnoteString](#) on page 357.

FootnoteString

This property is used to define the footnote text that is drawn when the FootnoteDisplay property is enabled. When a footnote string is defined with this property, the FootnoteDisplay property is automatically set to true.

Syntax:

```
setFootnoteString("string");  
string = getFootNoteString();
```

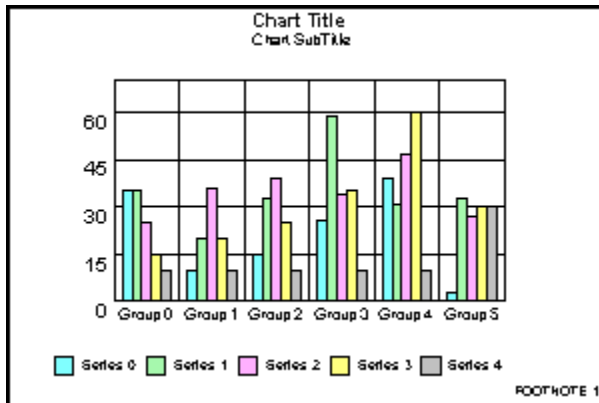
where:

string

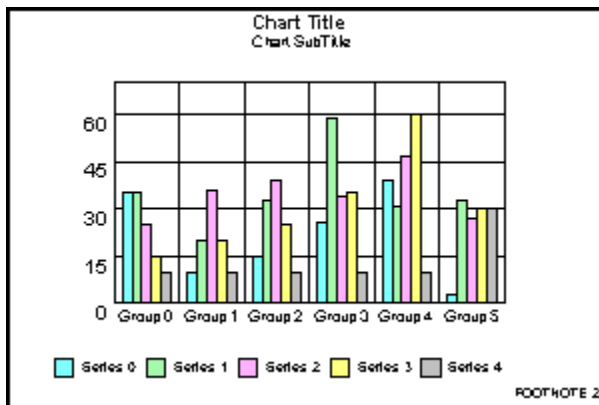
Represents any printable characters. A null string is the default value.

Example:

```
setDepthRadius(0);  
setFootnoteString("FOOTNOTE 1");
```



```
setFootnoteString("FOOTNOTE 2");
```



See [FootnoteDisplay](#) on page 356

ForceSeriesAbsolute

This property allows creating a graph with stacked and non-stacked risers. This type of graph is useful when showing cumulative sales of several products in monetary units along with sales volume and return volume. It is also useful for financial ratios, statistical information (such as standard deviations), or control limits.

Syntax:

```
setForceSeriesAbsolute(getSeries(id), boolean);
```

where:

id

Is the object ID of the Y-Axis field.

Using a Stacked chart type will cumulate all the fields, therefore this setting has to be added manually and set to true for each field that should be non-stacked.

boolean

Can be one of the following:

true enables stacked and non-stacked charts.

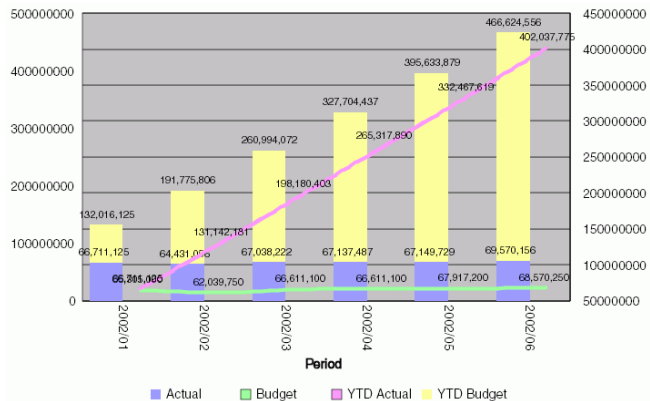
false disables stacked and non-stacked charts. This is the default value.

Example

The following code shows how to use stacked and non-stacked charts in the same graph. The relevant code is shown in bold. Note the numbers at each data point. The bars are plotted on the Y-axis on the left, and the lines are plotted on the Y-axis on the right for clarity.

```
setForceSeriesAbsolute(getSeries(1), true);
setForceSeriesAbsolute(getSeries(2), true);
setForceSeriesAbsolute(getSeries(3), false);
```

The output is:



FrameAutoShade

This property controls auto-shading of Side and Bottom Frames in a 2.5D graph. When enabled, the side and bottom frames are automatically colored and shaded based on the color applied to the graph frame. When auto-shading is disabled, side and bottom frames are not filled with color automatically.

Syntax:

`setFrameAutoShade (boolean);`

where:

boolean

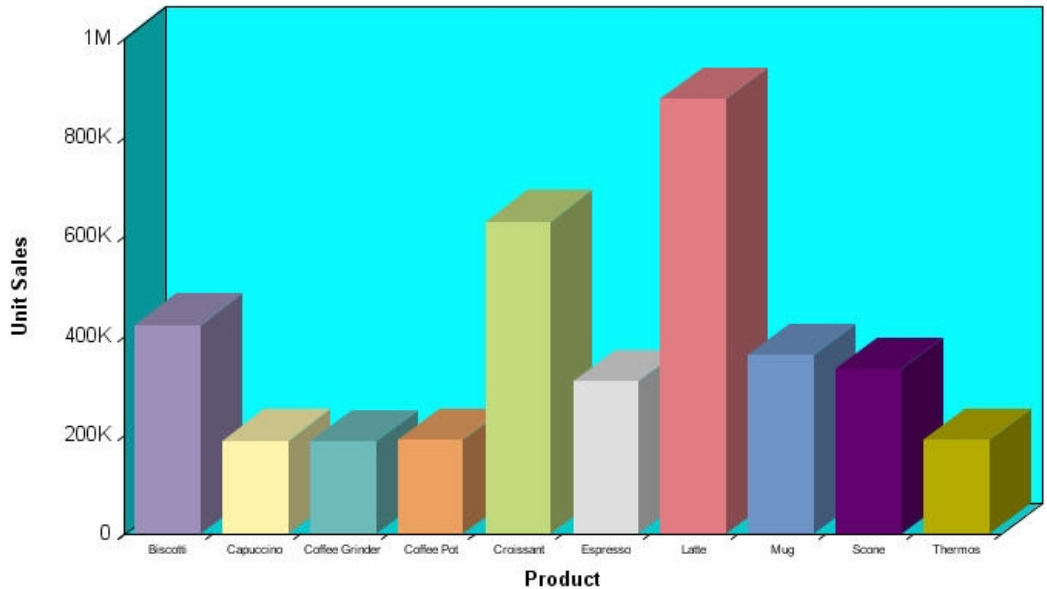
Can be one of the following:

`true` enables side and bottom frame auto-shading.

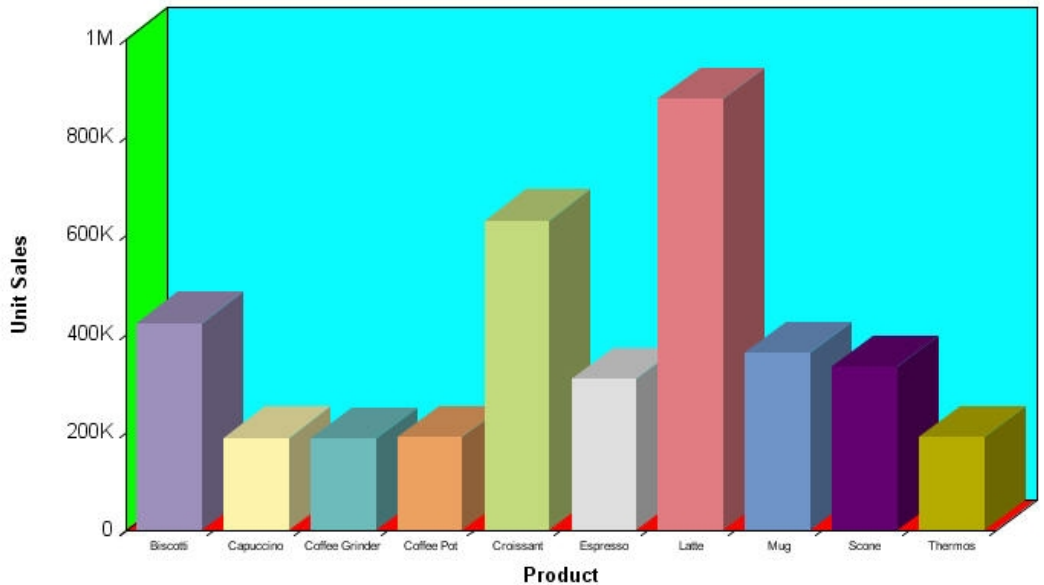
`false` disables side and bottom frame auto-shading. This value is the default.

Example:

```
setFrameAutoShade(true);
setFillColor(getFrame(),new Color(0,255,255));
```



```
setFrameAutoShade(false);
setFillColor(getFrame(),new Color(0,255,255));
setFillColor(getFrameSide(),new Color(0,255,0));
setFillColor(getFrameBottom(),new Color(255,0,0));
```



See [setFillColor\(\)](#) on page 890, [getFrame\(\)](#) on page 1059, [getFrameSide\(\)](#) on page 1060, [getFrameBottom\(\)](#) on page 1059.

FrameDisplay

This property enables or disables drawing of the graph frame.

Syntax:

```
setFrameDisplay(boolean);
```

```
boolean = getFrameDisplay();
```

where:

boolean

Can be one of the following:

[true](#) draws graph frame. This value is the default.

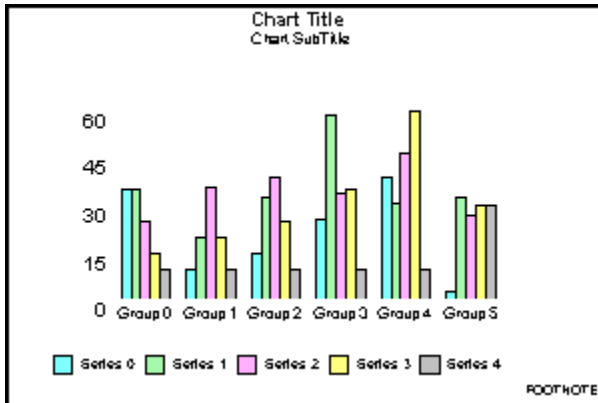
[false](#) does not draw graph frame.

Example:

```

setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setDepthRadius(0);
  setFrameDisplay(false);

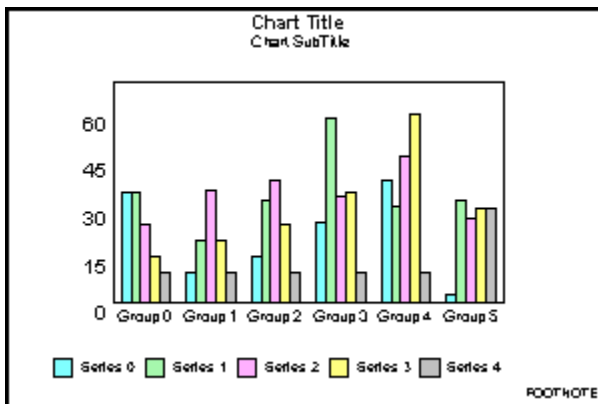
```



```

setFrameDisplay(true);

```

**FunnelLabelValueLocation**

In a Funnel graph, this property defines the location where values labels are drawn.

Syntax:

```

setFunnelLabelValueLocation(value);

```

```
value = getFunnelLabelValueLocation();
```

where:

value

Can be one of the following:

```
LOCATION_LABELS_ONTOP_OF_VALUES_BOTH_ON_RIGHT =
0; LOCATION_LABELS_ONLEFT_VALUES_ON_RIGHT =
1; LOCATION_VALUES_ONLEFT_LABELS_ON_RIGHT = 2;
```

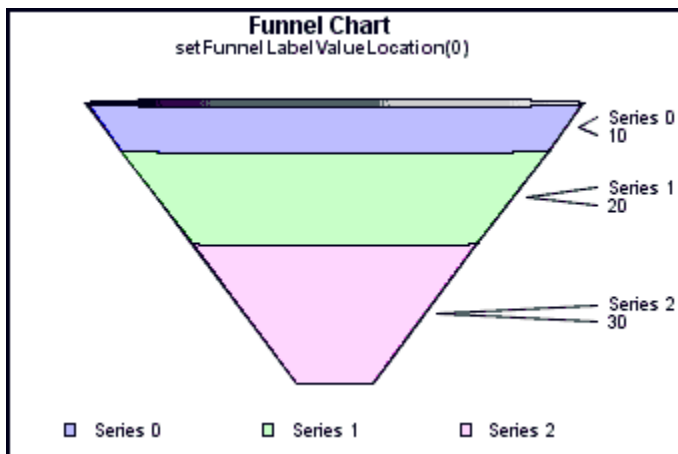
0 for labels on top of values drawn on the right of the funnel. 0 is the default value.

1 for labels on the left of the funnel or Values on the right of the funnel.

2 for values on the left of the funnel or Labels on the right of the funnel.

Example:

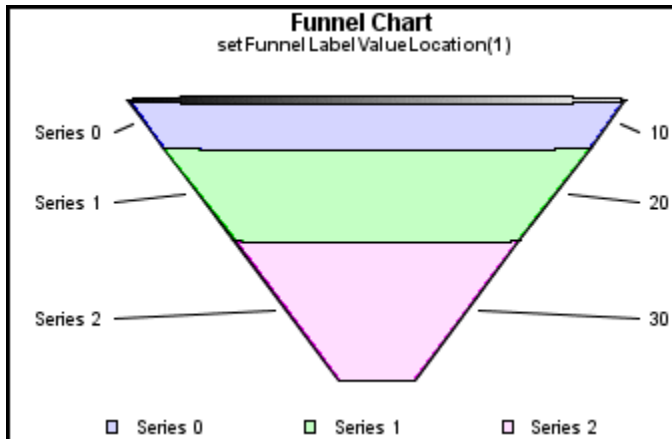
```
setTitleString("Funnel Chart");
setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTransparentBorderColor(getFrame(), true);
setFunnelLabelValueLocation(0);
setSubtitleString("setFunnelLabelValueLocation(0)");
```



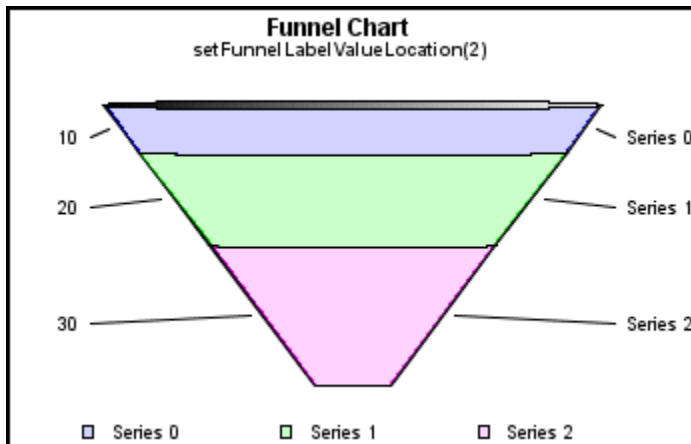
```
setFunnelLabelValueLocation(1);
```



```
setSubtitleString("setFunnelLabelValueLocation(1)");
```



```
setFunnelLabelValueLocation(2);
setSubtitleString("setFunnelLabelValueLocation(2)");
```



See [DisplayFunnelGroup](#) on page 343, [PieDepth](#) on page 474, [PieRotate](#) on page 488, [PieTilt](#) on page 493.

GaugeBackgroundStyle

When the GraphType property is set to 110 (gauge graph), this property selects the gauge's background style.

Syntax:

```
setGaugeBackgroundStyle(value);
```

```
value = getGaugeBackgroundStyle()
```

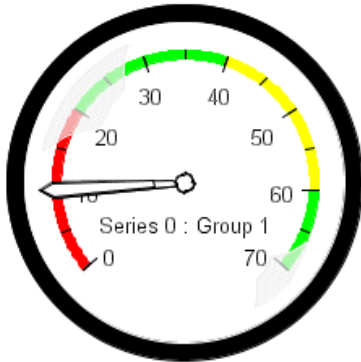
where:

value

Is an integer between 0 and 11. 1 is the default value.

Example:

```
setGraphType(110);
  setGaugeBackgroundStyle(5);
setDisplay(getFrame(), false);
```



```
setGaugeBackgroundStyle(2);
setDisplay(getFrame(), false);
```

**GaugeColorCenterByQuality**

This property controls whether the state of the quality band is reflected in the center of the gauge needle. When enabled, the center of the needle is colored the same color as the quality band the Gauge needle is pointing to. This is useful to provide an instant status of the quality bands. When disabled, the center of the needle is drawn with the default styling.

Syntax:

```
setGaugeColorCenterByQuality(boolean);  
boolean =getGaugeColorCenterByQuality();
```

where:

boolean

Can be one of the following:

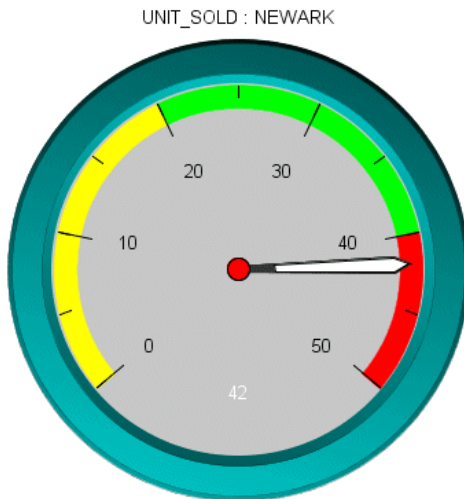
[true](#) colors the center of the needle the same as the quality band that the Gauge needle is pointing to.

[false](#) draws the center with default styling. This value is the default.

Example:

In the following example, GaugeColorCenterByQuality is enabled (true), so the center of the needle is red because the needle is pointing to a red quality band:

```
setGaugeColorCenterByQuality(true);
```



In the following example, GaugeColorCenterByQuality is disabled (false), so the center of the needle is the default color:

```
setGaugeColorCenterByQuality(false);
```



GaugeDescending

When the GraphType property is set to 110 (gauge graph), this property defines how gauge bands are drawn.

Syntax:

```
setGaugeDescending(boolean);
```

```
boolean = getGaugeDescending();
```

where:

boolean

Can be one of the following:

true draws gauge bands in descending order.

false draws gauge bands in ascending order. This value is the default.

GaugeIgnoreBackgroundStyle

When the GraphType property is set to 110 (gauge graph), this property enables or disables use of the GaugeBackgroundStyle property.

Syntax:

```
setGaugeIgnoreBackgroundStyle(boolean);  
boolean = getIgnoreBackgroundStyle();
```

where:

boolean

Can be one of the following:

[true](#) does not use the GaugeBackgroundStyle property.

[false](#) uses the value of the GaugeBackgroundStyle property to format the gauge background.
This value is the default.

GaugeMajorAutoStep

When the GraphType property is set to 110 (gauge graph), this property enables or disables automatic calculation of the number of major grid steps. When automatic calculation is disabled, the Y1MajorGridStep property defines the number of major grid steps in the gauge.

Syntax:

```
setGaugeMajorAutoStep(boolean);  
boolean = getGaugeMajorAutoStep();
```

where:

boolean

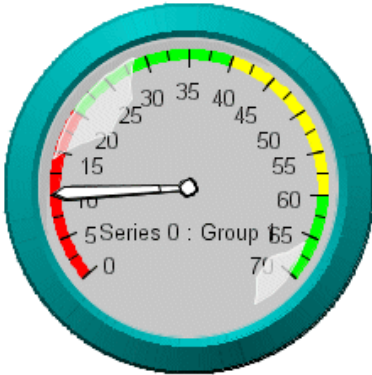
Can be one of the following:

[true](#) automatically calculates the number of major grid steps.

[false](#) uses the number of major grid steps set by the Y1MajorGridStep property. This value is the default.

Example:

```
setGraphType(110);
    setGaugeMajorAutoStep(false);
setDisplay(getFrame(), false);
setY1MajorGridStep(5.0);
```

**GaugeMajorTickRelativeTickLength**

When the GraphType property is set to 110 (gauge graph), this property defines the relative length of major tick marks.

Syntax:

```
setGaugeMajorTickRelativeTickLength(value);
value = getGaugeMajorTickRelativeTickLength()
```

where:

value

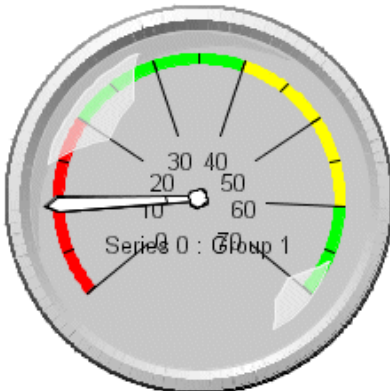
Is a value between 0.0 and 0.5. This value is a Java double-precision number. The default value is 0.1.

Example:

```
setGraphType(110);  
setGaugeBackgroundStyle(3);  
setDisplay(getFrame(), false);  setGaugeMajorTickRelativeTickLength(0.2);
```



```
setGaugeBackgroundStyle(3);  
setDisplay(getFrame(), false);  setGaugeMajorTickRelativeTickLength(0.4);
```



GaugeMinorAutoStep

When the GraphType property is set to 110 (gauge graph), this property enables or disables automatic calculation of the number of minor grid steps. When automatic calculation is disabled, the Y1MinorGridStep property defines the number of minor grid steps in the gauge.

Syntax:

```
setGaugeMinorAutoStep(boolean);  
boolean = getGaugeMinorAutoStep();
```

where:

boolean

Can be one of the following:

true automatically calculates the number of minor grid steps.

false uses the number of minor grid steps set by the Y1MinorGridStep property. This value is the default.

GaugeMinorTickRelativeTickLength

When the GraphType property is set to 110 (a gauge chart), this property defines the relative length of minor tick marks.

Syntax:

```
setGaugeMinorTickRelativeTickLength(value);  
value = getGaugeMinorTickRelativeTickLength()
```

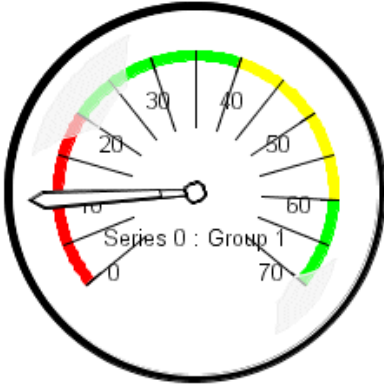
where:

value

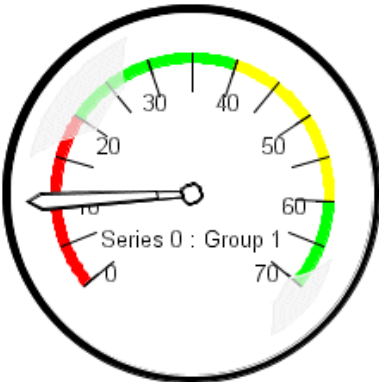
Is a value between 0.0 and 0.5. This value is a Java double-precision number. The default value is 0.1.

Example:

```
setGraphType(110);  
setGaugeBackgroundStyle(6);  
setDisplay(getFrame(), false); setGaugeMinorTickRelativeTickLength(0.4);
```



```
setGaugeBackgroundStyle(6);  
setDisplay(getFrame(), false); setGaugeMinorTickRelativeTickLength(0.2);
```

**GaugeNeedleStyle**

When the GraphType property is set to 110 (gauge graph), this property selects the gauge's needle style.

Syntax:

```
setGaugeNeedleStyle(value);  
value = getGaugeNeedleStyle()
```

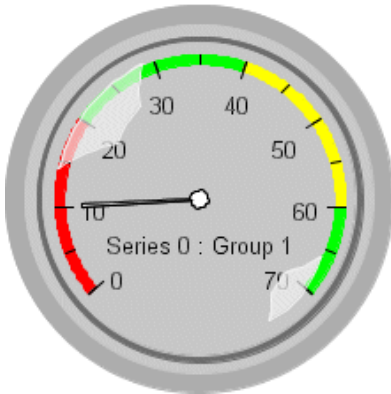
where:

value

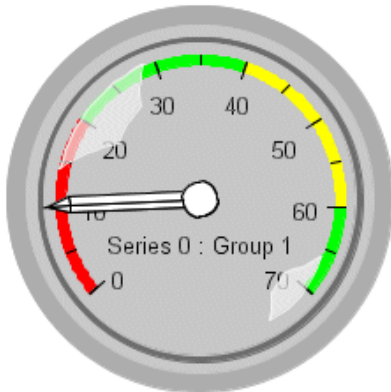
Is an integer between 0 and 5. The default value is 2.

Example:

```
setGraphType(110);
setGaugeBackgroundStyle(7);
setDisplay(getFrame(), false); setGaugeNeedleStyle(0);
```



```
setGaugeBackgroundStyle(7);
setDisplay(getFrame(), false); setGaugeNeedleStyle(1);
```



GaugeRelativeInnerRadius

When the GraphType property is set to 110 (gauge graph), this property defines the inner radius of the gauge bands and labels relative to the gauge's outer background. Smaller numbers place the inner radius closer to the center of the gauge.

Syntax:

```
setGaugeRelativeInnerRadius(value);  
value = getGaugeRelativeInnerRadius()
```

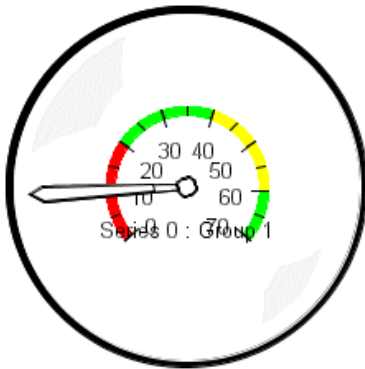
where:

value

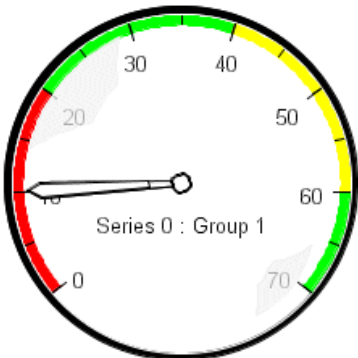
Is a value between 0.0 and 0.9. Is a Java double-precision number. Default value is 0.6.

Example:

```
setGraphType(110);  
setGaugeBackgroundStyle(6);  
setDisplay(getFrame(), false); setGaugeRelativeInnerRadius(0.4);
```



```
setGaugeBackgroundStyle(6);  
setDisplay(getFrame(), false); setGaugeRelativeInnerRadius(0.9);
```



GaugeRelativeThickness

When the `GraphType` property is set to 110 (gauge graph), this property selects the relative thickness of the gauge bands.

Syntax:

```
setGaugeRelativeThickness (value);  
value = getGaugeRelativeThickness()
```

where:

value

Is a value between 0.0 and 0.7. This value is a Java double-precision number. The default value is 0.1.

Example:

```
setGraphType(110);  setGaugeRelativeThickness(0.15);
```



GaugeSizeMode

When there are multiple gauge charts drawn on a line, the size of the gauges may vary due to the space available. This property allows you to control the size of the gauges and whether they should all be the same size or the size should be flexible.

Syntax:

```
setGaugeSizeMode(boolean);
```

where:

boolean

Can be one of the following:

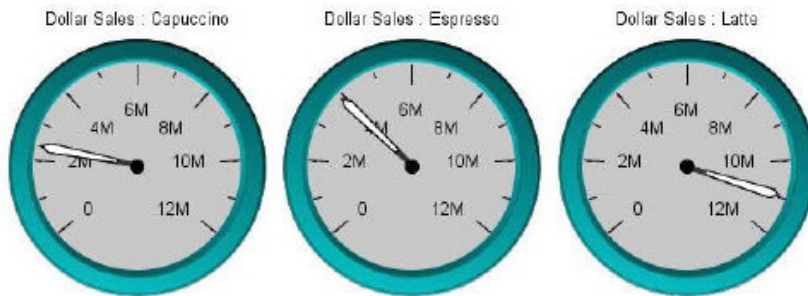
`true` forces gauges of the same size to be drawn. If there is one or three gauges per row, the charts will be the same size.

`false` disables drawing gauges of the same size. The size of the gauge is dynamic and will be based on the available space. This value is the default.

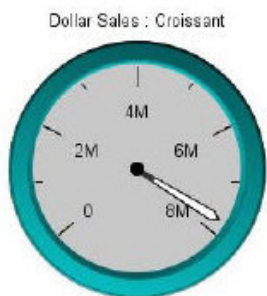
Example:

```
setPiesPerRow(3);  
setGaugeSizeMode(true);
```

Coffee

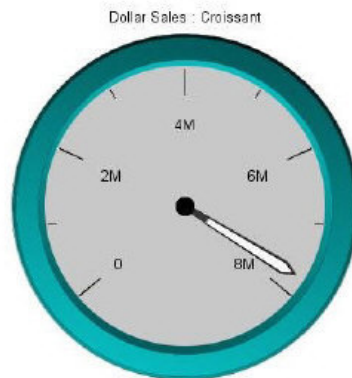


Food



Example:

```
setPiesPerRow(3);
setGaugeSizeMode(false);
```

Coffee**Food****GaugeStartAngle**

When the GraphType property is set to 110 (gauge graph), this property selects the gauge's start angle.

Syntax:

```
setGaugeStartAngle(value);
value = getGaugeStartAngle ()
```

where:

value

Is an integer between 0 and 359. The default value is 220.

GaugeStopAngle

When the GraphType property is set to 110 (gauge graph), this property selects the gauge's stop angle.

Syntax:

```
setGaugeStopAngle(value);
```

```
value = getGaugeStopAngle();
```

where:

value

Is an integer between 0 and 359. The default value is 320.

GraphType

This property selects a graph type. A wide variety of two- and three-dimensional graphs are available. They are listed and illustrated in [Graph Types and 3D Preset Viewing Angles](#) on page 1213.

Syntax:

```
setGraphType(value);
```

```
value = getGraphType();
```

where:

value

Can be one of the following:

0 through 7, 9, 10, 12 through 14 for 3D Graphs.

17 through 23 for Vertical Bar Graphs. 17 is the default.

24 through 30 for Horizontal Bar Graphs.

31 through 35 for Vertical Area Graphs.

36 through 40 for Horizontal Area Graphs.

41 through 47 for Vertical Line Graphs.

48 through 54 for Horizontal Line Graphs.

55 through 60 for Pie Graphs.

61 through 64 for Scatter Graphs.

65, 66 for Polar Graphs.

67 through 69 for Radar Graphs.

70 through 84 and 88 for Stock Graphs.

85, 86 for Histograms.

87 for Spectral Maps.

90 through 92 for Bubble Graphs.

93, 94 for Pie-Bar Graphs.

100, 101 for Waterfall Graphs.

102 for Pareto Graphs.

103 through 105 for Multi-Y Graphs.

106 for Funnel Graphs.

See [Graph Types and 3D Preset Viewing Angles](#) on page 1213 for an illustration of each graph type.

Grid3DFloorDisplayX

When a 3D graph is selected with `GraphType`, this property enables or disables X-axis grid lines on the 3D cube floor.

Syntax:

```
setGrid3DFloorDisplayX(boolean);
```

```
boolean = get3DFloorDisplayX();
```

where:

boolean

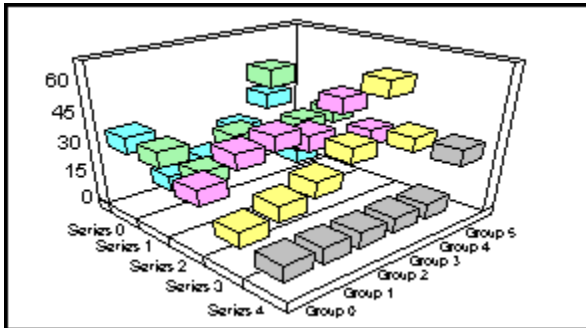
Can be one of the following:

`true` draws grid lines.

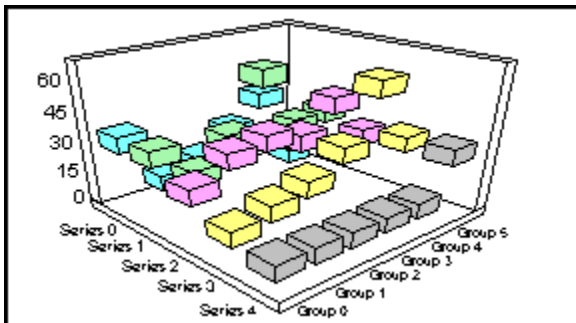
`false` does not draw grid lines. This value is the default.

Example:

```
setGrid3DFloorDisplayX(true);
setGraphType(4);
setAutoshading(false);
```



```
setGrid3DFloorDisplayX(false);
```



See [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DFloorDisplayZ

When a 3D graph is selected with GraphType, this property enables or disables Z-axis grid lines on the 3D cube floor.

Syntax:

```
setGrid3DFloorDisplayZ(boolean);
boolean = get3DFloorDisplayZ();
```

where:

boolean

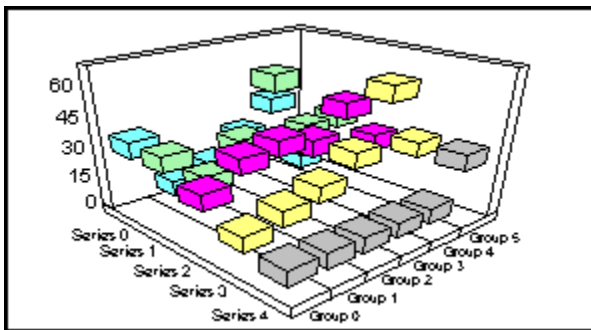
Can be one of the following:

true draws grid lines.

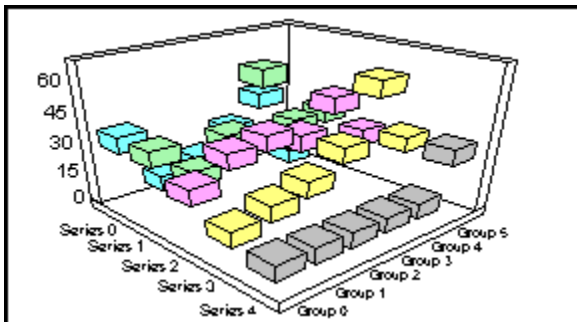
false does not draw grid lines. This value is the default.

Example:

```
setGrid3DFloorDisplayZ(true);
setGraphType(4);
setAutoshading(false);
```



```
setGrid3DFloorDisplayZ(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DLeftWallDisplayY

When a 3D graph is selected with GraphType, this property enables or disables Y-axis grid lines on the 3D cube left wall.

Syntax:

```
setGrid3DLeftWallDisplayY(boolean);  
boolean = getGrid3DLeftWallDisplayY();
```

where:

boolean

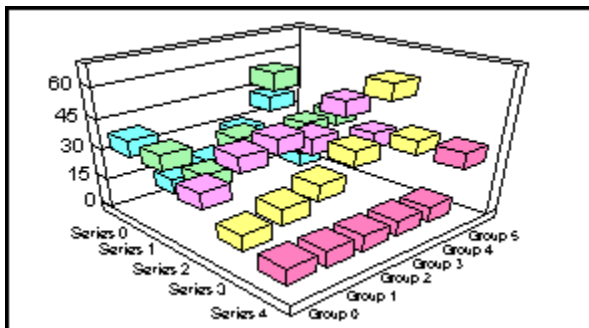
Can be one of the following:

`true` draws grid lines.

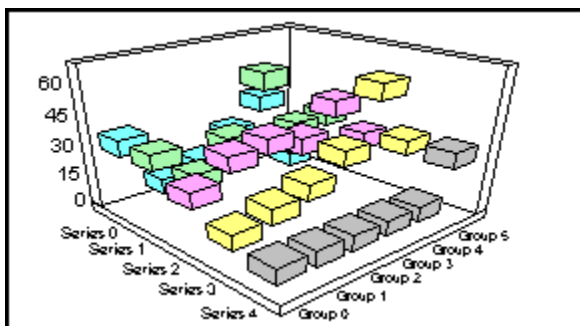
`false` does not draw grid lines. This value is the default.

Example:

```
setGraphType(4);
  setGrid3DLeftWallDisplayY(true);
setAutoshading(false);
```



```
setGrid3DLeftWallDisplayY(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DLeftWallDisplayZ

When a 3D graph is selected with GraphType, this property enables or disables Z-axis grid lines on the 3D cube left wall.

Syntax:

```
setGrid3DLeftWallDisplayZ(boolean);
boolean = getGrid3DLeftWallDisplayZ();
```

where:

boolean

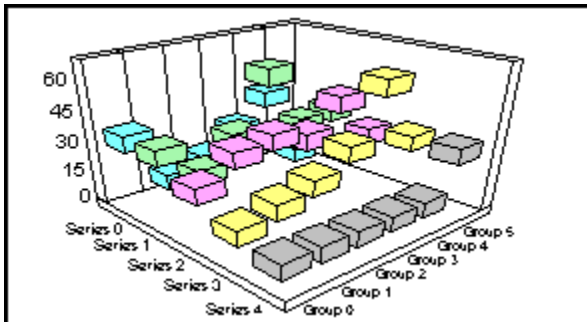
Can be one of the following:

true draws grid lines.

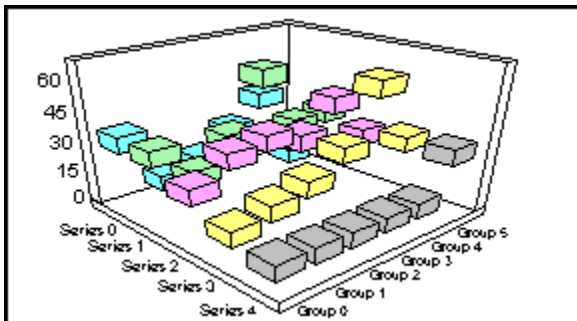
false does not draw grid lines. This value is the default.

Example:

```
setGraphType(4);
  setGrid3DLeftWallDisplayZ(true);
setAutoshading(false);
```



```
setGrid3DLeftWallDisplayZ(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DRightWallDisplayX

When a 3D graph is selected with GraphType, this property enables or disables X-axis grid lines on the right wall of the 3D cube.

Syntax:

```
setGrid3DRightWallDisplayX(boolean);  
boolean = getGrid3DRightWallDisplayX();
```

where:

boolean

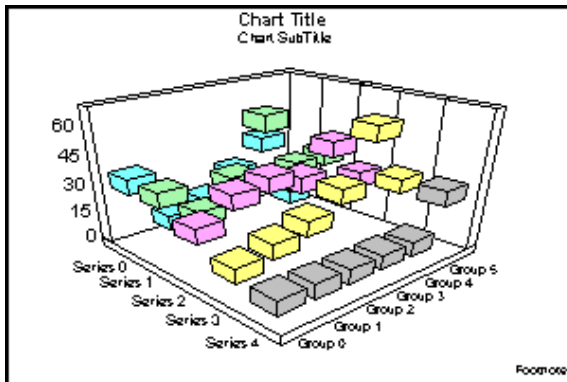
Can be one of the following:

`true` draws grid lines.

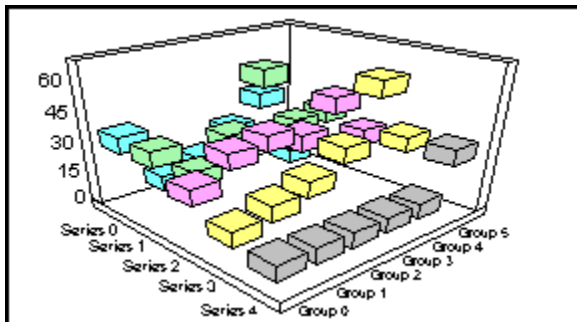
`false` does not draw grid lines. This value is the default.

Example:

```
setGraphType(4);
  setGrid3DRightWallDisplayX(true);
setAutoshading(false);
```



```
setGrid3DRightWallDisplayX(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DRightWallDisplayY

When a 3D graph is selected with GraphType, this property enables or disables Y-axis grid lines on the right wall of the 3D cube.

Syntax:

```
setGrid3DRightWallDisplayY(boolean);
boolean = getGrid3DRightWallDisplayY();
```

where:

boolean

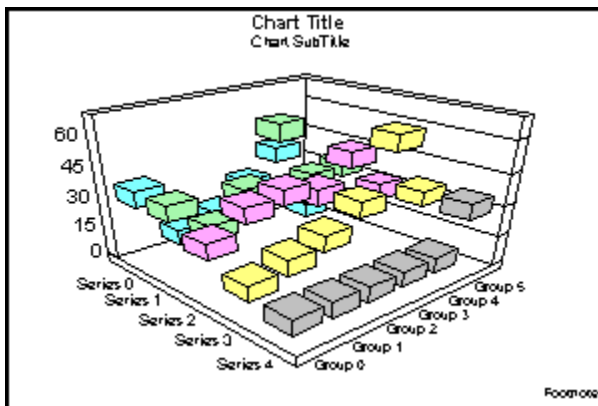
Can be one of the following:

true draws grid lines.

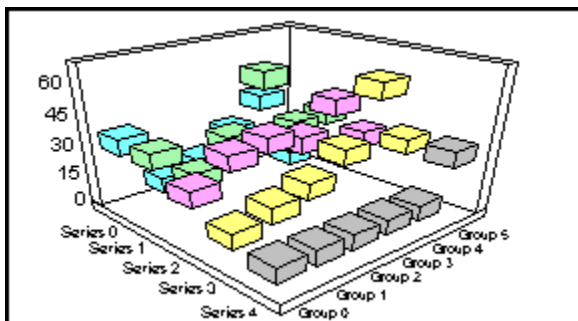
false does not draw grid lines. This value is the default.

Example:

```
setGraphType(4);
  setGrid3DRightWallDisplayY(true);
setAutoshading(false);
```



```
setGrid3DRightWallDisplayY(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DRiserDisplayX

When a 3D surface graph is selected with GraphType, this property enables or disables X-axis grid lines on the risers in a 3D graph.

Syntax:

```
setGrid3DRiserDisplayX(boolean);  
boolean = getGrid3DRiserDisplayX();
```

where:

boolean

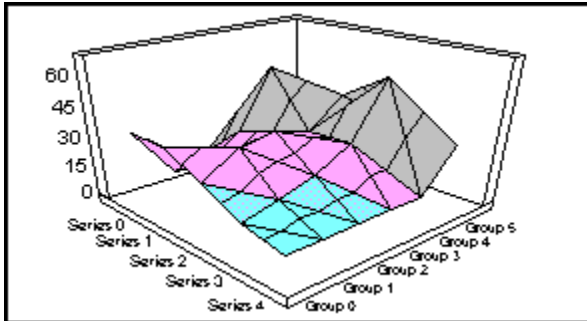
Can be one of the following:

[true](#) draws grid lines.

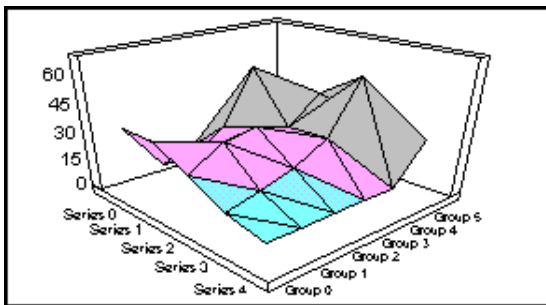
[false](#) does not draw grid lines. This value is the default.

Example:

```
setGrid3DRiserDisplayX(true);
setGraphType(12);
```



```
setGrid3DRiserDisplayX(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayY](#) on page 391, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DRiserDisplayY

When a 3D graph is selected with GraphType, this property enables or disables Y-axis grid lines on the risers in a 3D graph.

Syntax:

```
setGrid3DRiserDisplayY(boolean);
boolean = getGrid3DRiserDisplayY();
```

where:

boolean

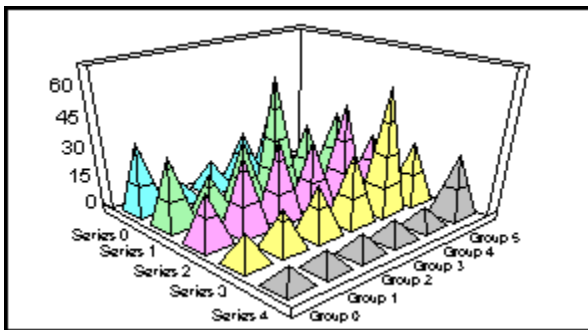
Can be one of the following:

true draws grid lines.

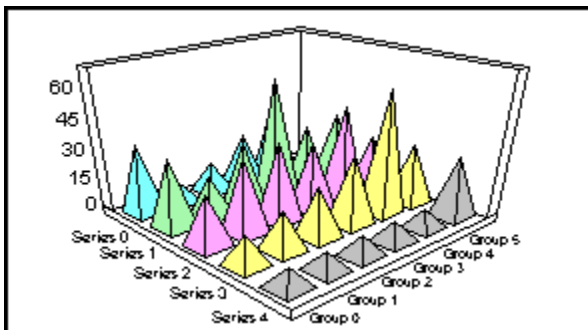
false does not draw grid lines. This value is the default.

Example:

```
setGrid3DRiserDisplayY(true);  
setGraphType(1);
```



```
setGrid3DRiserDisplayY(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayZ](#) on page 393.

Grid3DRiserDisplayZ

When a 3D surface graph is selected with GraphType, this property enables or disables Z-axis grid lines on the risers in a 3D graph.

Syntax:

```
setGrid3DRiserDisplayZ(boolean);
```

```
boolean = getGrid3DRiserDisplayZ();
```

where:

boolean

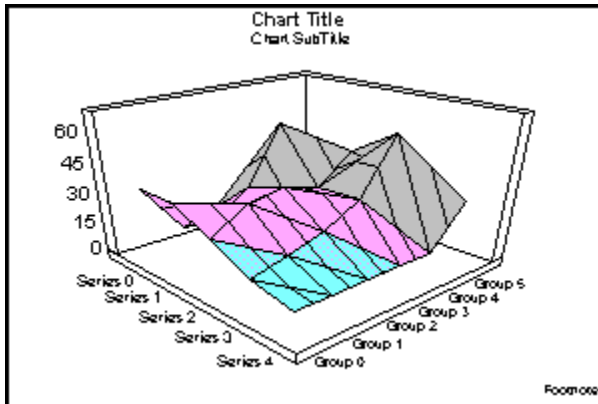
Can be one of the following:

`true` draws grid lines.

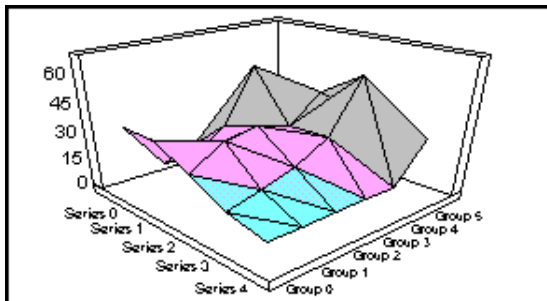
`false` does not draw grid lines. This value is the default.

Example:

```
setGrid3DRiserDisplayZ(true);
setGraphType(12);
```



```
setGrid3DRiserDisplayZ(false);
```



See [Grid3DFloorDisplayX](#) on page 381, [Grid3DFloorDisplayZ](#) on page 382, [Grid3DLeftWallDisplayY](#) on page 384, [Grid3DLeftWallDisplayZ](#) on page 385, [Grid3DRightWallDisplayX](#) on page 387, [Grid3DRightWallDisplayY](#) on page 388, [Grid3DRiserDisplayX](#) on page 390, [Grid3DRiserDisplayY](#) on page 391.

HiliteBarResize

This property enables or disables showing a riser's bar value when a bar is resized in a clustered bar graph.

Syntax:

```
setHiliteBarResize(boolean);
boolean = setHiliteBarResize();
```

where:

boolean

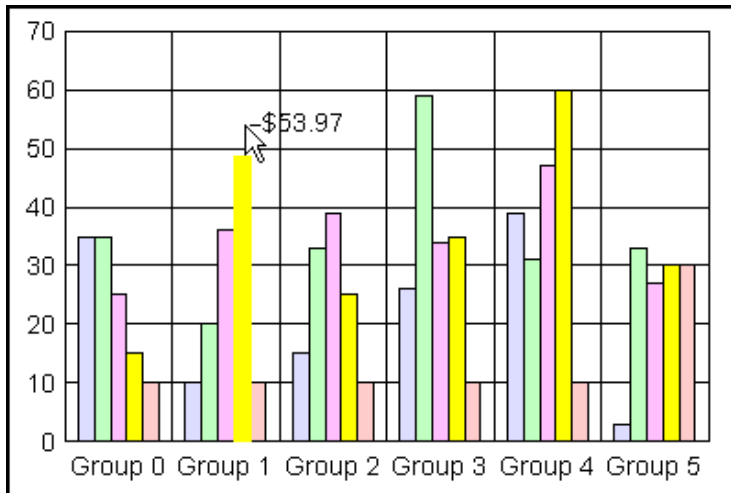
Can be one of the following:

true shows riser's bar value when a bar is resized using the option key. This value is the default.

false does not show riser's bar value.

Example:

```
setDepthRadius(0);
setDepthAngle(0);
setHiliteBarResize(true);
```



IgnoreTickSkip

When O1-axis label skipping is defined with the O1LabelAutoSkip, O1LabelSkipBegin, and O1LabelSkipCount properties, this property enables or disables the O1-Axis tick marks (if present) to follow label skipping.

Syntax:

```
setIgnoreTickSkip(boolean);
boolean = setIgnoreTickSkip();
```

where:

boolean

Can be one of the following:

true does not adjust O1-axis tick marks to follow label skipping.

false adjusts O1-axis tick marks to follow label skipping set by O1LabelAutoSkip, O1LabelSkipBegin, and O1LabelSkipCount. This value is the default.

InterpretAsHLOC

This property is included to support backward compatibility with Hi-Lo-Open-Close data sets that were created in stock graphs.

Syntax:

```
setInterpretAsHLOC(boolean);
```

```
boolean = setInterpretAsHLOC();
```

where:

boolean

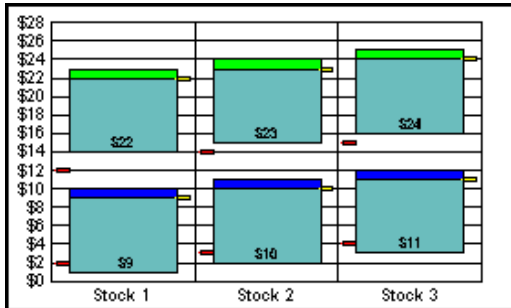
Can be one of the following:

true interprets data set values as High/Low/Open/Close.

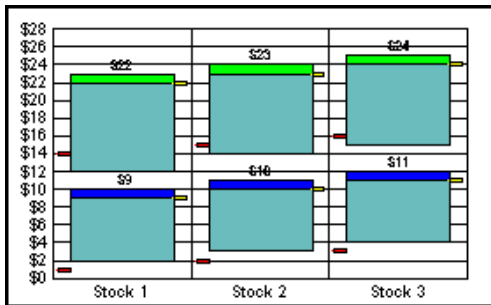
false interprets data set values as Open/High/Low/Close. This is the default value.

Example:

```
setGraphType(79);  setInterpretASHLOC(true);
```



```
setGraphType(79);  setInterpretASHLOC(false);
```

**LegendAutomatic**

This property enables or disables automatic sizing and positioning of the legend box.

Syntax:

```
setLegendAutomatic(boolean);
```

```
boolean = getLegendAutomatic();
```

where:

boolean

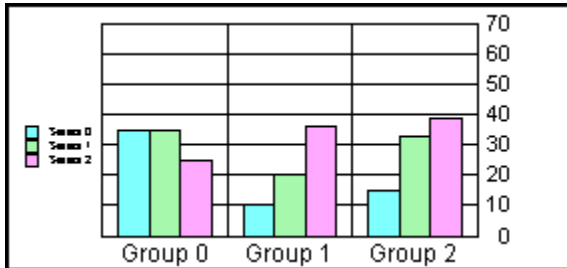
Can be one of the following:

true automatically sets the size of the legend box to half of the height or width of the graph and positions the legend box in the center of the graph frame.

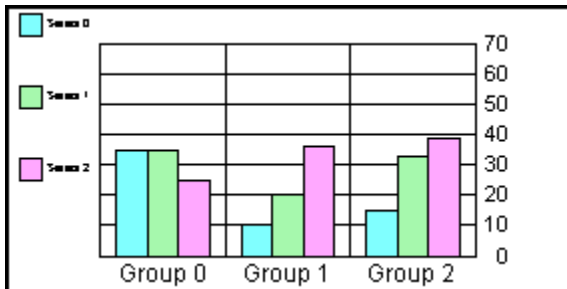
false does not auto-calculate legend position and size. This value is the default.

Example:

```
setViewableGroups(3);  
setViewableSeries(3);  
setDataScrollerPresenceGroups(0);  
setDepthRadius(0);  
setLegendOrient(0);  
    setLegendAutomatic(true);  
setY1AxisSide(1);  
setLegendRect(new Rectangle(-15340, -3478, 4948, 18550));
```



```
setLegendAutomatic(false);
```



See [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400, [LegendReverse](#) on page 404, [setLegendRect\(\)](#) on page 921.

LegendDisplay

This property enables or disables drawing of the legend box. It also affects all legend box objects (legend text, legend marker, and legend line). When the legend box is not drawn, neither are its components. Legends are not displayed for histogram graphs or any graph that is colored by group (ColorMode=2).

Syntax:

```
setLegendDisplay(boolean);  
boolean = getLegendDisplay();
```

where:

boolean

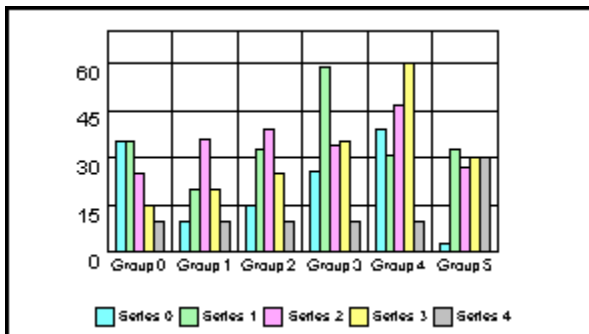
Can be one of the following:

[true](#) shows the legend.

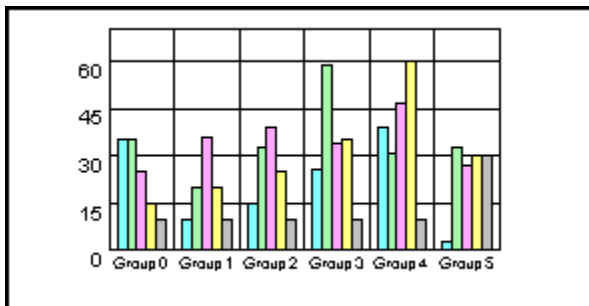
[false](#) does not show the legend. This is the default value.

Example:

```
setDepthRadius(0);
setLegendDisplay(true);
```



```
setLegendDisplay(false);
```



See [LegendAutomatic](#) on page 397, [LegendMarkerPosition](#) on page 400, [LegendReverse](#) on page 404, [UseSeriesShapes](#) on page 562, [setLegendRect\(\)](#) on page 921.

LegendMarkerPosition

This property sets the location and format of the graph legends.

Syntax:

```
setLegendMarkerPosition(value);
```

```
value = getLegendMarkerPosition();
```

where:

value

Can be one of the following:

[0](#) sets legend box to the left of legend text. 0 is the default value.

[1](#) sets legend text to the left of legend box.

[2](#) sets legend text below legend box.

[3](#) sets legend text above legend box.

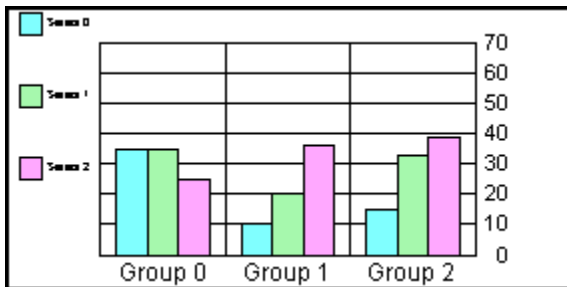
[4](#) sets legend text inside legend box.

Example:

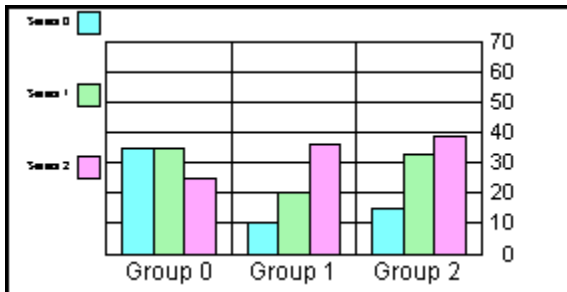
```

setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setDepthRadius(0);
setLegendOrient(0);
setLegendAutomatic(false);
setY1AxisSide(1);
    setLegendMarkerPosition(0);
setLegendRect(new Rectangle(-15340, -3478, 4948, 18550));

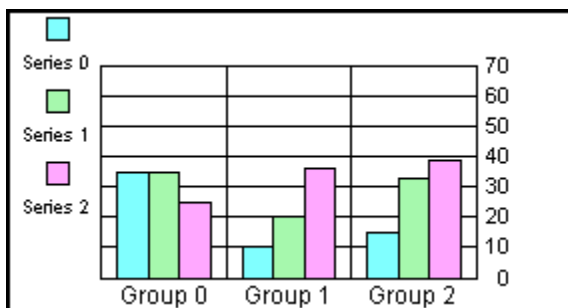
```



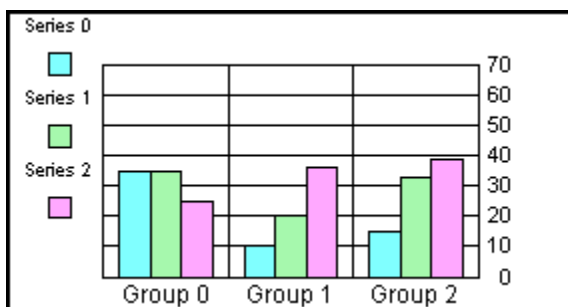
```
setLegendMarkerPosition(1);
```



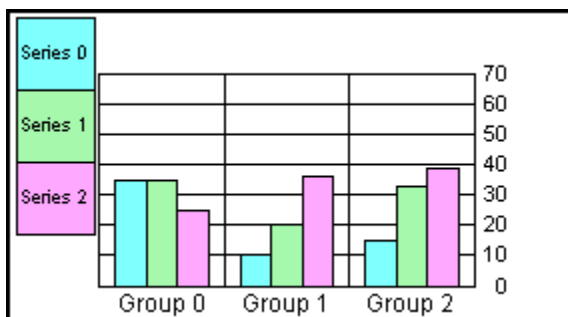
```
setLegendMarkerPosition(2);
```



`setLegendMarkerPosition(3);`



`setLegendMarkerPosition(4);`



See [LegendAutomatic](#) on page 397, [LegendDisplay](#) on page 398, [LegendReverse](#) on page 404, [UseSeriesShapes](#) on page 562, [setLegendRect\(\)](#) on page 921.

LegendMarkersPerRow

This property specifies the number of legend markers to draw in a single row in the legend area. LegendAutomatic must be set to false.

Syntax:

```
setLegendMarkersPerRow(value);
value = getLegendMarkersPerRow();
```

where:

value

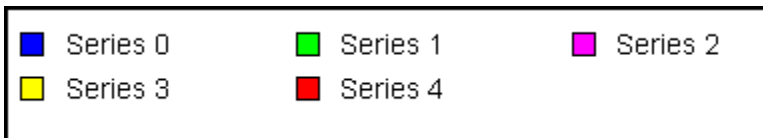
Is any integer between 0 and the number of series in the graph. 0 is the default and will draw the same number of markers in a row as the number of series in the graph. Any other value will draw that number of series in a row.

Example:

```
setLegendAutomatic(false); setLegendMarkersPerRow(2);
```



```
setLegendAutomatic(false); setLegendMarkersPerRow(3);
```

**LegendOrient**

This property controls the orientation of the legend markers in the legend area. If vertical (0) or horizontal (1) is selected, the size and placement of the legend area may need to be changed to accommodate the legend markers and legend text. The `setLegendRect()` method can be used for this purpose.

Syntax:

```
setLegendOrient(value);
value = getLegendOrient();
```

where:

value

Can be one of the following:

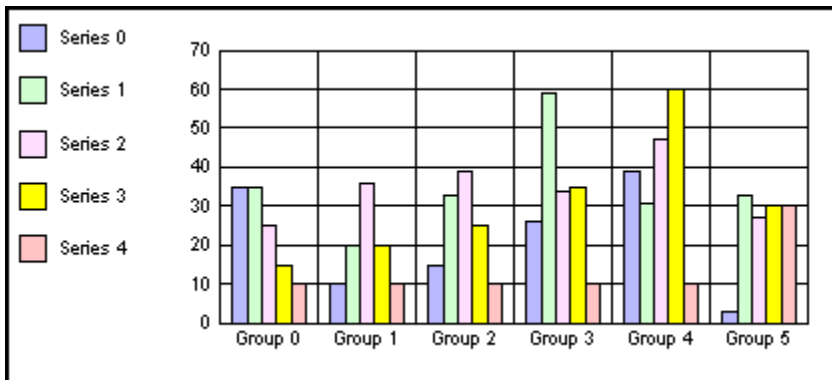
0 for vertical

1 for horizontal

[2](#) for auto (determined by height or width of Legend Rectangle). This value is the default.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(false);
setLegendOrient(0);
setLegendRect(new Rectangle(-16340, -2478, 5948, 17550));
```



Note: [LegendAutomatic](#) on page 397 must be set to false.

LegendReverse

This property enables or disables drawing of legends in reverse order.

Syntax:

```
setLegendReverse(boolean);
boolean = getLegendReverse();
```

where:

boolean

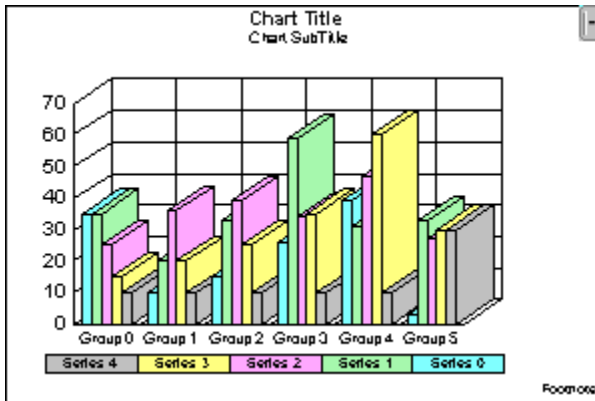
Can be one of the following:

[true](#) draws legends in reverse order.

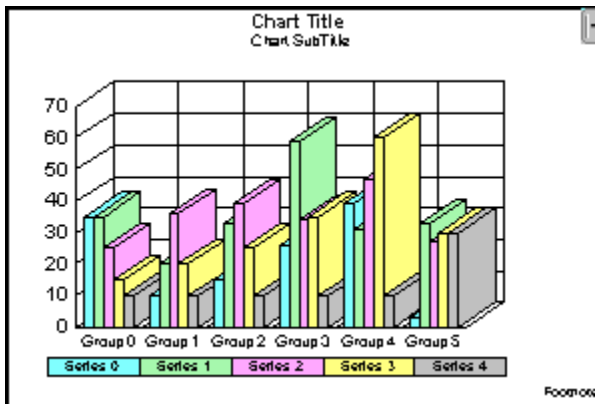
[false](#) draws legends in normal order. This value is the default.

Example:

```
setLegendMarkerPosition(4); setLegendReverse(true);
```



```
setLegendReverse(false);
```



See [LegendAutomatic](#) on page 397, [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400, [UseSeriesShapes](#) on page 562, [setLegendRect\(\)](#) on page 921.

LegendScroll

This property controls the presence of scroll bars in the legend area. If the legend scroll bar is enabled with `LegendScroll(1)` or `LegendScroll(2)`, you must set the starting series to show with `LegendSeriesStart` and the number of series to show in the scroll area with `LegendSeriesCount`.

Syntax:

```
setLegendScroll(value);
```

```
value = getLegendScroll();
```

where:

value

Can be one of the following:

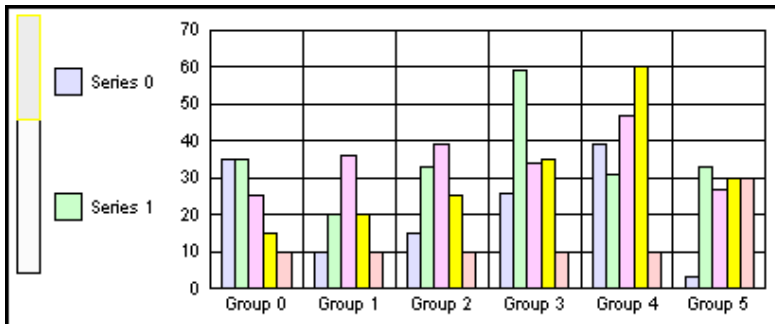
[0](#) never shows legend scroll bar. 0 is the default value.

[1](#) only shows legend scroll bar when the number of series exceeds the size of the legend area.

[2](#) always shows legend scroll bar.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(true);
setLegendScroll(2);
setLegendSeriesCount(2);
setLegendSeriesStart(0);
setLegendOrient(0);
setLegendRect(new Rectangle(-15340, -2478, 5948, 17550));
```



See [LegendSeriesCount](#) on page 406, [LegendSeriesStart](#) on page 407.

LegendSeriesCount

When the legend scroll bar is enabled with `LegendScroll(1)` or `LegendScroll(2)`, this property defines the number of series to show in the scrolled legend area.

Syntax:

```
setLegendSeriesCount(value);
integer = getLegendSeriesCount();
```

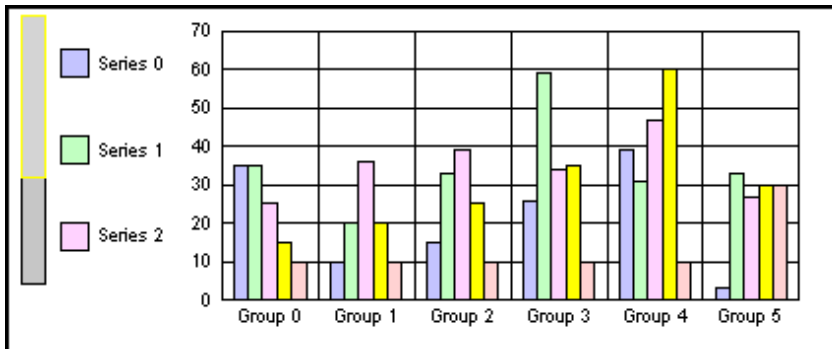
where:

integer

Is any integer between 0 and the number of series in the graph. 0 is the default value.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(true);
setLegendScroll(2);
    setLegendSeriesCount(3);
setLegendSeriesStart(0);
setLegendOrient(0);
setLegendRect(new Rectangle(-15340, -2478, 5948, 17550));
```



See [LegendScroll](#) on page 405, [LegendSeriesStart](#) on page 407.

LegendSeriesStart

When the legend scroll bar is enabled with `LegendScroll(1)` or `LegendScroll(2)`, this property defines the first series to show in the scrolled legend area.

Syntax:

```
setLegendSeriesStart(value);
integer = getLegendSeriesStart();
```

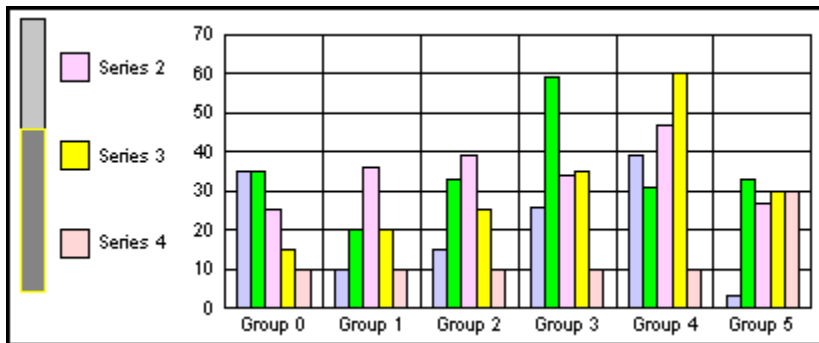
where:

integer

Is any integer between 0 and the number of series in the graph. 0 is the default value.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(true);
setLegendScroll(2);
setLegendSeriesCount(3);
    setLegendSeriesStart(2);
setLegendOrient(0);
setLegendRect(new Rectangle(-15340, -2478, 5948, 17550));
```



See [LegendScroll](#) on page 405, [LegendSeriesCount](#) on page 406.

MajorGridDrawEveryCount

This property controls whether some grid lines can be skipped or shown only every *n*th value on axis. It can be applied to a horizontal or vertical axis, or a combination of both. By default, this property is disabled and Major Grid lines are drawn for every value.

Syntax:

```
setMajorGridDrawEveryCount (IdentObj id, int);
```

where:

id

Is the object ID returned by a getXXMajorGrid() method, for example getO1MajorGrid(), where:

`XXX` can be one of the following:

`01` for X-axis

`Y1` for Y1-axis

`Y2` for Y2-axis

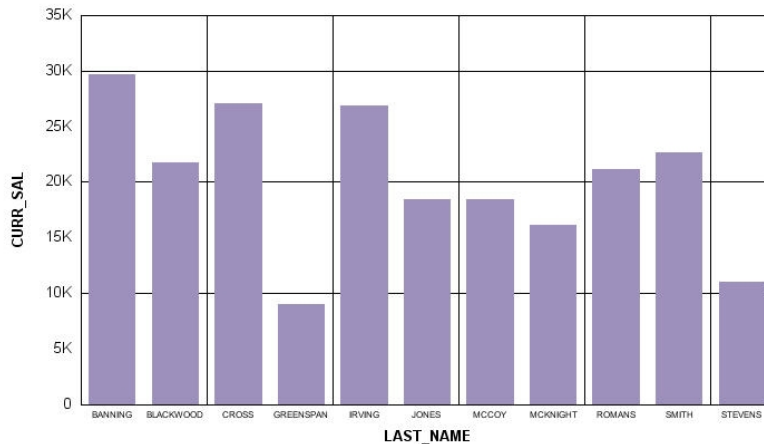
`X1` for X1-axis

`int`

Is an integer value starting from 1, which is the default value.

Example:

```
setMajorGridDrawEveryCount(get01MajorGrid(), 2);
setMajorGridDrawEveryCount(getY1MajorGrid(), 2);
```



MajorGridTickInterval

This property gets and sets major grids and ticks on interval (seconds to years) boundaries in a chart that uses a time scale axis.

Syntax:

```
setMajorGridTickInterval(value);
```

```
value = getMajorGridTickInterval();
```

where:

`value`

Can be one of the following:

[0](#) for TIME_INTERVAL_UNDEFINED. [0](#) is the default value.

[1](#) for TIME_INTERVAL_SECONDS

[2](#) for TIME_INTERVAL_MINUTES

[3](#) for TIME_INTERVAL_HOURS

[4](#) for TIME_INTERVAL_DAYS

[5](#) for TIME_INTERVAL_WEEKS

[6](#) for TIME_INTERVAL_MONTHS

[7](#) for TIME_INTERVAL_QUARTERS

[8](#) for TIME_INTERVAL_YEARS

MinorGridTickInterval

This property gets/sets minor grids and ticks on interval (seconds to years) boundaries in a chart that uses a time scale axis.

Syntax:

```
setMinorGridTickInterval(value);
```

```
value = getMinorGridTickInterval();
```

where:

value

Can be one of the following:

[0](#) for TIME_INTERVAL_UNDEFINED. [0](#) is the default value.

[1](#) for TIME_INTERVAL_SECONDS.

[2](#) for TIME_INTERVAL_MINUTES.

[3](#) for TIME_INTERVAL_HOURS.

[4](#) for TIME_INTERVAL_DAYS.

[5](#) for TIME_INTERVAL_WEEKS.

[6](#) for TIME_INTERVAL_MONTHS.

[7](#) for TIME_INTERVAL_QUARTERS.

[8](#) for TIME_INTERVAL_YEARS.

MarkerDisplay

This property enables or disables the display of markers in a 2D graph. The DepthRadius property must be set to zero in order for this property to be properly applied to the graph.

Syntax:

```
setMarkerDisplay(boolean)
```

```
boolean = getMarkerDisplay();
```

where:

boolean

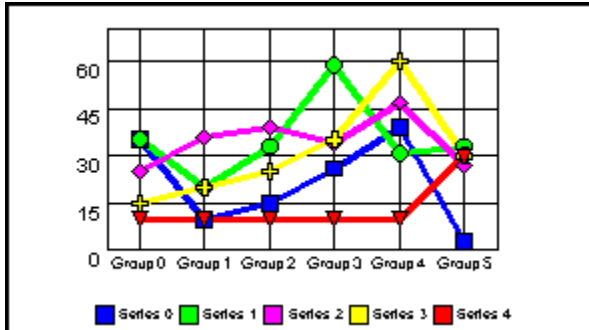
Can be one of the following:

true displays markers. This value is the default.

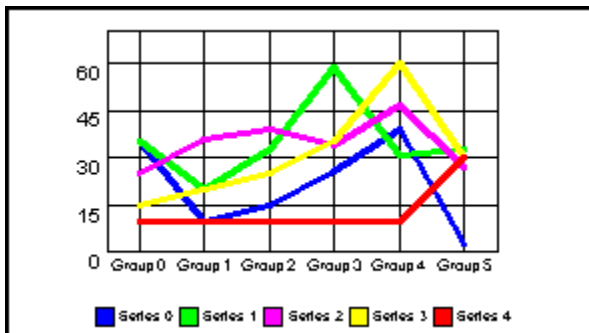
false does not display markers. Use this option to turn off markers in a line graph when you only want to see the lines themselves, and not the markers.

Example:

```
setGraphType(41);
setMarkerDisplay(true);
```



```
setMarkerDisplay(false);
```



See [ConnectLineMarkers](#) on page 262, [ConnectScatterMarkers](#) on page 265, [MarkerSizeDefault](#) on page 412, [setMarkerShape\(\)](#) on page 940, [setMarkerSize\(\)](#) on page 942.

MarkerSizeDefault

This property defines the size of markers in a 2D graph. The `setMarkerSize()` method can override this property by assigning a marker size to a specific series or group point.

Syntax:

```
setMarkerSizeDefault(value);
value = getMarkerSizeDefault();
```

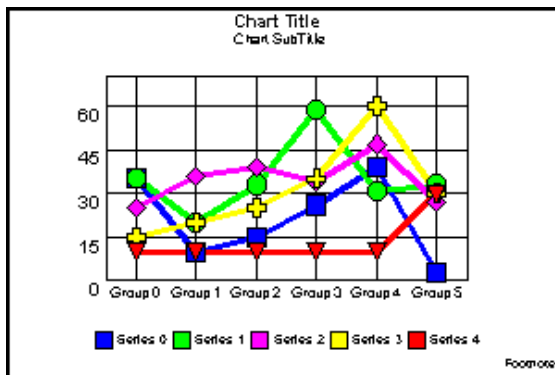

where:

value

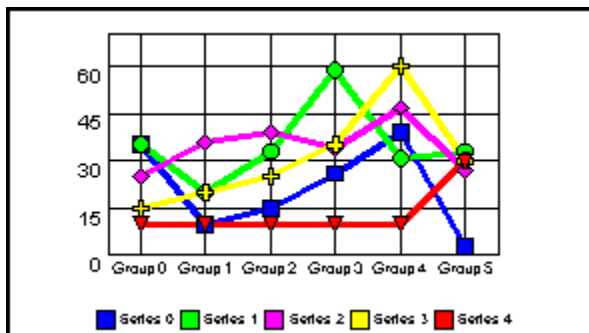
Is any integer between 0 and 100. 30 is the default value.

Example:

```
setGraphType(41);
setDepthRadius(0);
setMarkerSizeDefault(90);
```



```
setMarkerSizeDefault(60);
```



See [ConnectLineMarkers](#) on page 262, [ConnectScatterMarkers](#) on page 265, [MarkerDisplay](#) on page 411, [UseSeriesShapes](#) on page 562, [setMarkerShape\(\)](#) on page 940, [setMarkerSize\(\)](#) on page 942.

MoveAxisTitles

This property enables or disables the movement of axis titles. The axis title is selected and dragged with the mouse. A title string must be defined and setTitleDisplay must be set to true.

Syntax:

```
setMoveAxisTitles(boolean);
```

```
boolean = getMoveAxisTitles();
```

where:

boolean

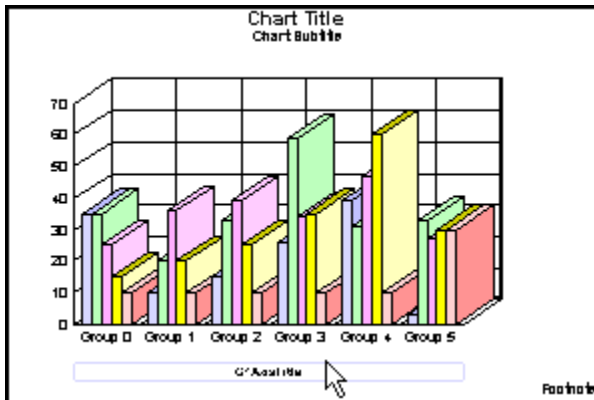
Can be one of the following:

`true`.

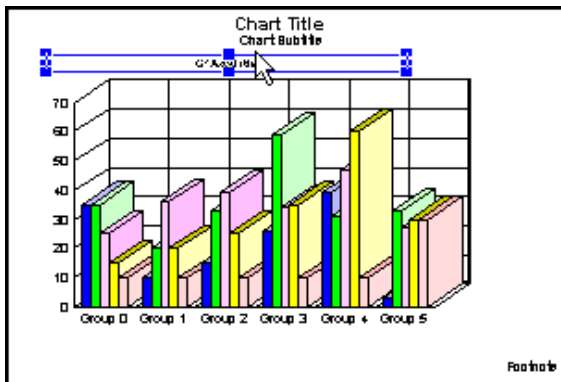
`false`. This is the default value.

Example:

```
setMoveAxisTitles(false);
setO1TitleString("O1AxisTitle");
setO1TitleDisplay(true);
```



```
setMoveAxisTitles(true);
setO1TitleString("O1AxisTitle");
setO1TitleDisplay(true);
```

**NestedLabels**

This property enables or disables the use of the nested labels callback interface. If this property is enabled, you must create your own version of the callback which must implement all of the abstract methods listed in `TDGNestedLabel.Java`:

- ❑ `int getNumLevels();` Used to determine the number of levels of nested labels.
- ❑ `int getNumLabelsOnLevel(int nLevel);` Used to determine the number of labels at `nLevel`.

- ❑ Vector getAllLabels(int nLevel); Used to get all the labels at nLevel.
- ❑ String getLabel(int nGroup, int nLevel); Get the string specified at nGroup and nLevel.
- ❑ int getLabelGrouping(int nGroup, int nLevel); Get the number of labels grouped at nLevel.

In addition to creating these methods, you must use the getO1LabelCallback() method to define the callback.

Syntax:

```
setNestedLabels(boolean);  
boolean = getNestedLabels();
```

where:

boolean

Can be one of the following:

true.

false. This is the default value.

Note: You may use getO1LabelCallback() to determine if a callback has been assigned with setO1LabelCallback.

NullLegendMarkerShapesAsSquares

This property enables or disables drawing of the legend marker when marker shape is set to null.

Syntax:

```
setNullLegendMarkerShapesAsSquares(boolean);
```

where:

boolean

Can be one of the following:

true draws the legend marker. This value is the default.

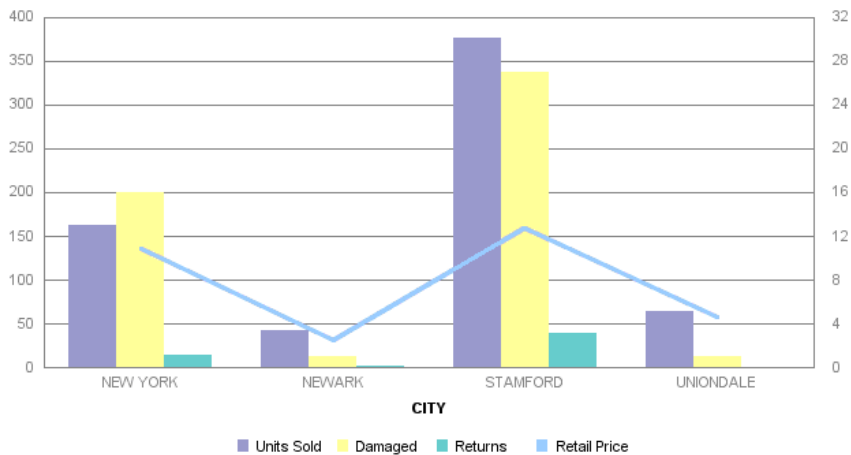
false disables drawing of the legend marker.

Example:

```

setUseSeriesShapes(true);
setMarkerShape(getSeries(0),1);
setMarkerShape(getSeries(1),1);
setMarkerShape(getSeries(2),1);
setMarkerShape(getSeries(3),0); setNullLegendMarkerShapesAsSquares(true);

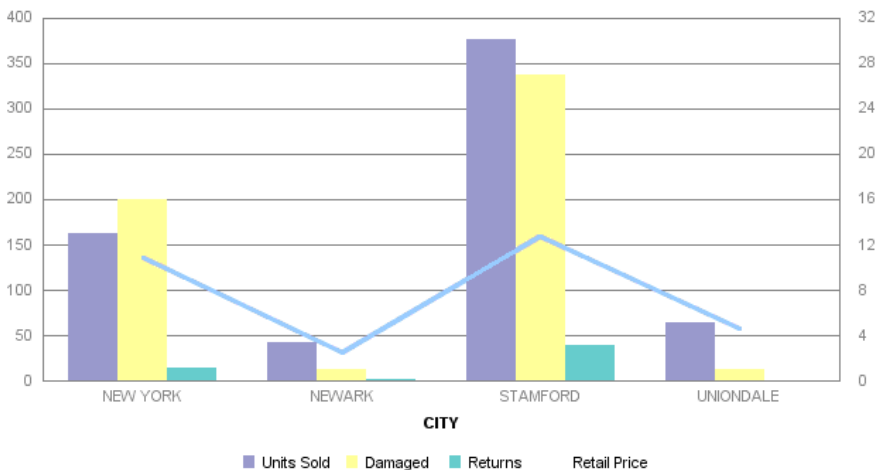
```



```

setUseSeriesShapes(true);
setMarkerShape(getSeries(0),1);
setMarkerShape(getSeries(1),1);
setMarkerShape(getSeries(2),1);setMarkerShape(getSeries(3),0);
setNullLegendMarkerShapesAsSquares(false);

```



Note: UseSeriesShapes must be enabled, and setMarkerShape must be set to null to disable drawing of the legend marker.

See [UseSeriesShapes](#) on page 562, [setMarkerShape\(\)](#) on page 940, [LegendAutomatic](#) on page 397, [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400.

NumMissingDataSegments

This property is used in conjunction with FillMissingData to define the number of missing data segments for filling in missing data in a line graph.

Syntax:

```
setNumMissingDataSegments(value);  
integer = getNumMissingDataSegments();
```

where:

integer

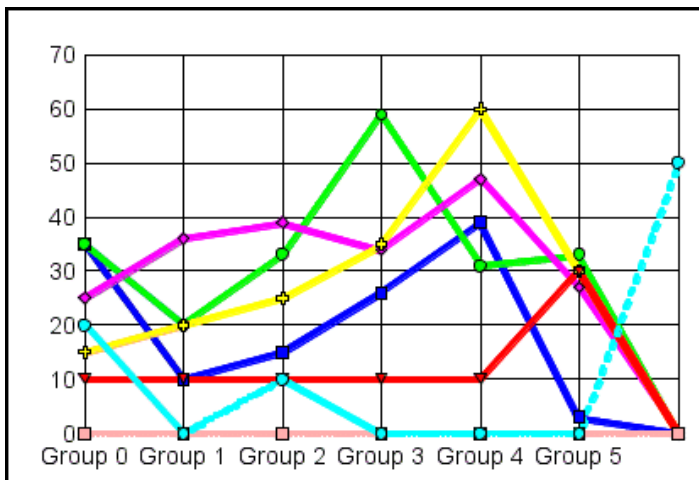
Is any integer between 1 and 25. 3 is the default value.

Example:

```

setGraphType(41);
setLegendDisplay(false);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setData(7,0,20.0);
setData(7,2,10.0);
setData(7,6,50.0);
setDataRangeToExtent();
setDepthRadius(0);
setFillMissingData(1);
    setNumMissingDataSegments(20);

```

**NumPieArcSegmentsPerPixel**

This property is used to improve the appearance of pie graphs by specifying the number of segments to draw in the arc of the pie.

Syntax:

```

setNumPieArcSegmentsPerPixel(value);
value = getNumPieArcSegmentsPerPixel();

```

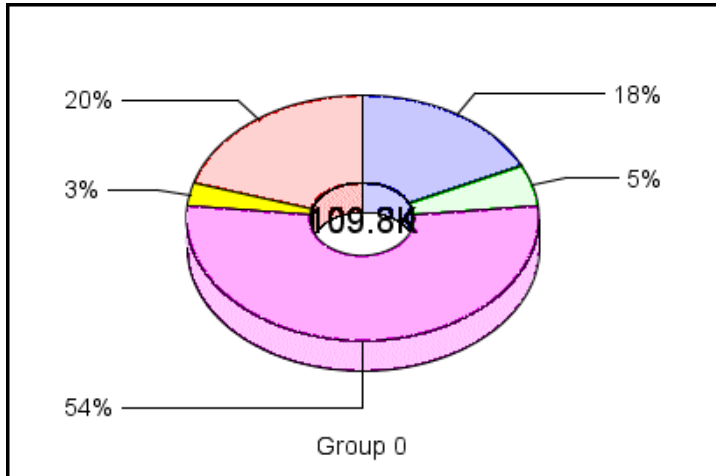
where:

value

Is a value between 0 and 1. This value is a Java double-precision number. 0.15 is the default value.

Example:

```
setPieDepth(60);  
setPieTilt(30);  
setGraphType(56);  
setPrimitiveAntialiasing(true); setNumPieArcSegmentsPerPixel(1.0);
```

**OtherPercentage**

If OtherSeries is true, very small slices in a pie graph are not drawn individually. Instead, they are summed together in an Other slice. Very small is defined as a slice whose percent of the total pie is less than the value of the property OtherPercentage (default is 0.5 percent).

Syntax:

```
setOtherPercentage(value);  
value = getOtherPercentage();
```

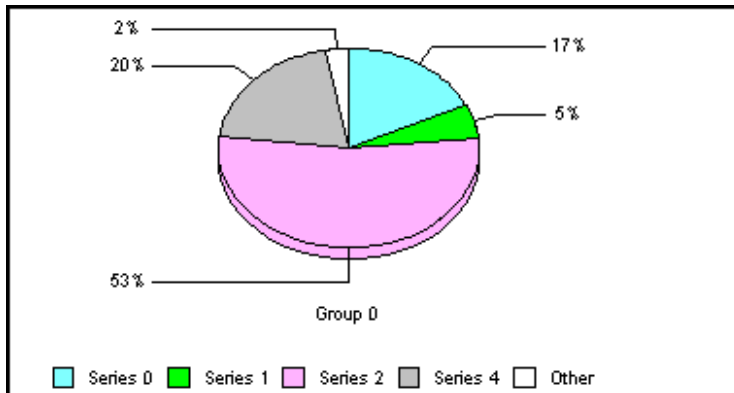
where:

value

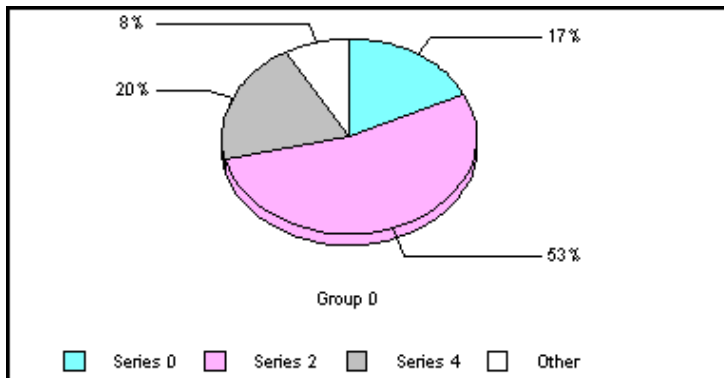
Is a value between 0 and 100. 0.5 is the default value.

Example:

```
setGraphType(55);
setOtherSeries(true);
setOtherPercentage(3.0);
```



```
setOtherPercentage(6.0);
```



See [OtherSeries](#) on page 421.

OtherSeries

This property controls whether or not the Other Series is shown on the chart. When enabled, pie slices with the values that are equal to or less than what is defined in Other Percentage property (default is 0.5 percent) are grouped into one slice, called "Other" and can be displayed in the "Other" Legend. When disabled, all slices are drawn on the pie chart.

Syntax:

```
setOtherSeries(boolean);
```

```
boolean = getOtherSeries();
```

where:

boolean

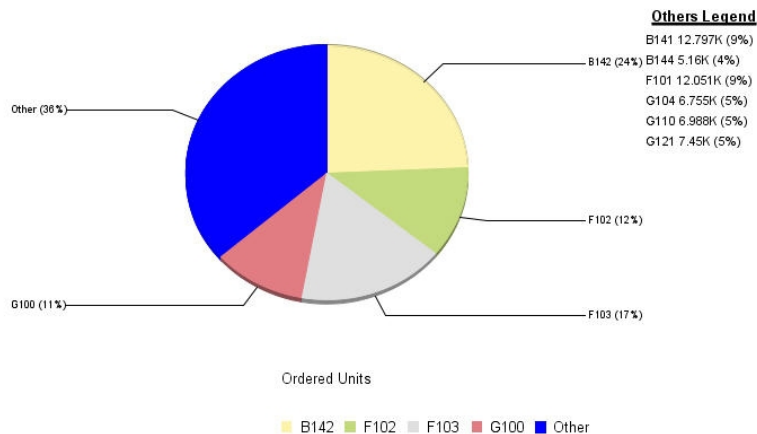
Can be one of the following:

true draws slices less than or equal to OtherPercentage together as an "Other" slice.

false draws all slices individually.

Example:

```
setGraphType(55);
  setOtherSeries(true);
setOtherPercentage(10.0);
setDisplay(getPieOtherFrame(), true);
setFontSizeAbsolute(getPieOtherLabel(), true);
setFontSizeInPoints(getPieOtherLabel(), 8);
setPlaceResize(getPieOtherLabel(), 0);
setFillColor(getSeries(-1), new Color(0,0,255));
setBorderColor(getSeries(-1), new Color(0,0,255));
setPieLabelDisplay(3);
```



See *OtherPercentage* on page 420.

OutputSVG

This property enables or disables output in Scalable Vector Graphics (SVG) format by the `sendSVGToStream()` method.

Syntax:

```
setOutputSVG(boolean);
boolean = getOutputSVG();
```

where:

boolean

Can be one of the following:

[true](#) allows output in SVG format. This value is the default.

[false](#) does not allow output in SVG format.

OverDrawSkip

This property can be used to accelerate the drawing of 2D and 2.5D bar graphs that represent very large data sets. When this property is enabled and there are so many bar risers so that several would be drawn on each pixel of the screen (overdraw), the graphing engine selects and draws only the largest and smallest values. This property has no effect on most data sets.

Syntax:

```
setOverdrawSkip(boolean);
boolean = getOverdrawSkip();
```

where:

boolean

Can be one of the following:

[true](#) skips overdraw. This value is the default.

[false](#) does not skip overdraw.

O1AxisLineDisplay

This property enables or disables drawing of the O1 axis line. The value assigned to O1AxisSide determines the location of the O1 axis line. The O1 axis line is typically displayed on the bottom of a vertical graph or on the left side of a horizontal graph. The O1AxisSide property may be used to change the location of the O1 axis line. Because the O1 axis base line is typically at the base or side of the graph frame, the line may not be visible even when this property is enabled. To make the O1 axis line wider and, therefore, more visible, use the getO1AxisLine() method to get the object ID of the line and the setLineWidth() method define the width of the line.

Syntax:

```
setO1AxisLineDisplay(boolean);
boolean = getO1AxisLineDisplay();
```

where:

boolean

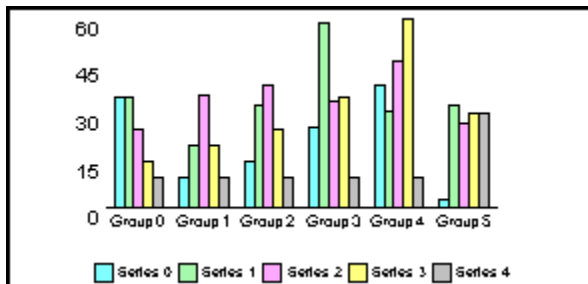
Can be one of the following:

[true](#) draws O1-axis base line. This value is the default.

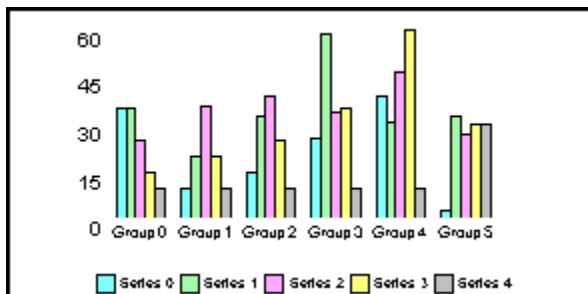
[false](#) does not draw O1-axis line.

Example:

```
setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setFrameDisplay(false);
  setO1AxisLineDisplay(true);
```



```
setO1AxisLineDisplay(false);
```



See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [getAxisSide\(\)](#) on page 720, [getLineWidth\(\)](#) on page 771.

O1AxisSide

This property controls which side of the graph the primary ordinal (O1) axis is imaged. It also determines where axis labels are drawn when the O1LabelDisplay property is set to true. The primary ordinal axis is typically imaged at the bottom of the graph in vertical graphs and on the left side in horizontal graphs.

Syntax:

```
setO1AxisSide(value);  
value = getO1AxisSide();
```

where:

value

Can be one of the following:

0 for Vertical: Bottom or Horizontal: Left. 0 is the default value.

1 for Vertical: Top or Horizontal: Right

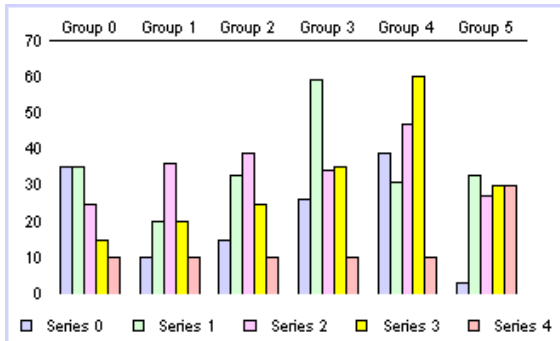
2 for Vertical: Top and Bottom or Horizontal: Right and Left

Example:

```

setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setFrameDisplay(false);
  setO1AxisSide(1);

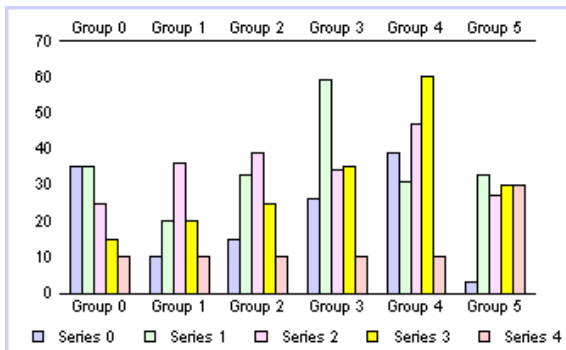
```



```

setO1AxisSide(2);

```

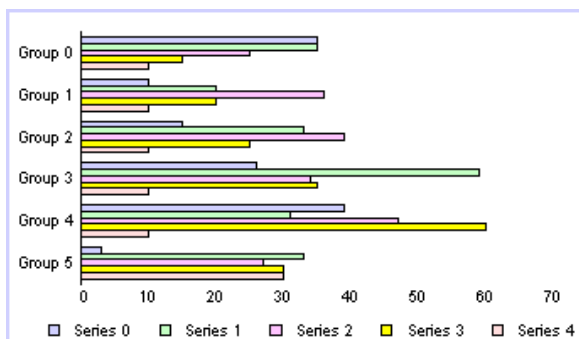


```

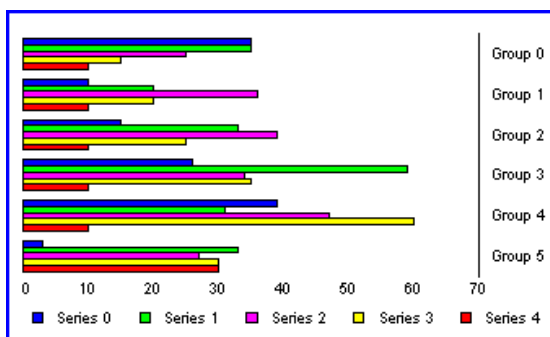
setDepthRadius(0);
setGraphType(24);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setFrameDisplay(false);

```

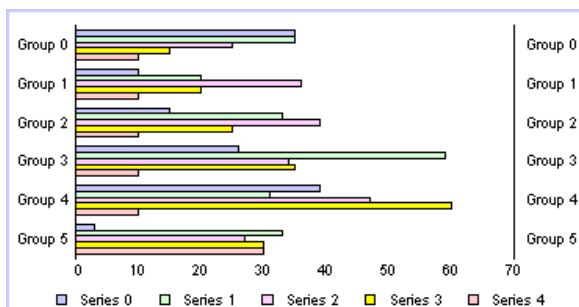
```
setO1AxisSide(0);
```



```
setO1AxisSide(1);
```



```
setO1AxisSide(2);
```



See [O1AxisLineDisplay](#) on page 423, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856.

O1ExcludeMaxLabel

This property enables or disables drawing of the maximum label on the primary ordinal (O1) axis.

Syntax:

```
setO1ExcludeMaxLabel(boolean);  
boolean = getO1ExcludeMaxLabel();
```

where:

boolean

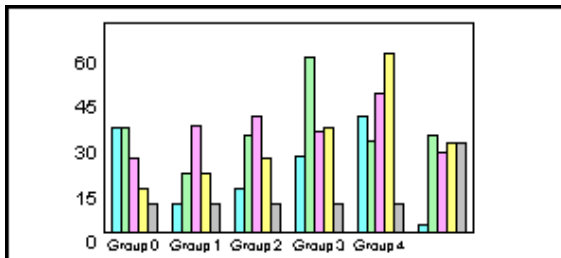
Can be one of the following:

true excludes the maximum label on the O1-axis.

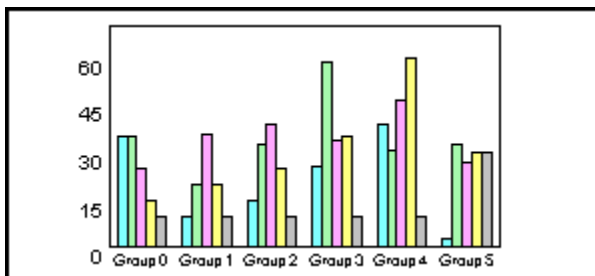
false includes the maximum label on the O1-axis. This value is the default.

Example:

```
setDepthRadius(0);  
setY1MajorGridDisplay(false);  
setY1AxisLineDisplay(false);  
setO1MajorGridDisplay(false); setO1ExcludeMaxLabel(true);
```



```
setO1ExcludeMaxLabel(false);
```



See [O1ExcludeMinLabel](#) on page 429, [getExcludeMaxLabel\(\)](#) on page 744, [setExcludeMaxLabel\(\)](#) on page 888.

O1ExcludeMinLabel

This property enables or disables drawing of the minimum label on the primary ordinal (O1) axis.

Syntax:

```
setO1ExcludeMinLabel(boolean);
boolean = getO1ExcludeMinLabel();
```

where:

boolean

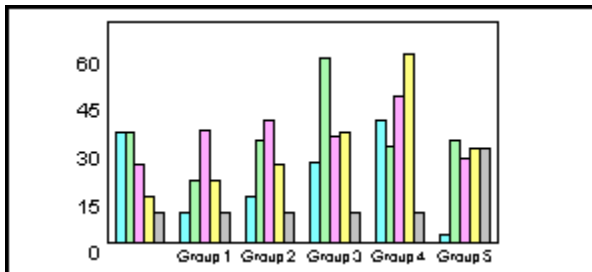
Can be one of the following:

[true](#) excludes the minimum label on the O1-axis.

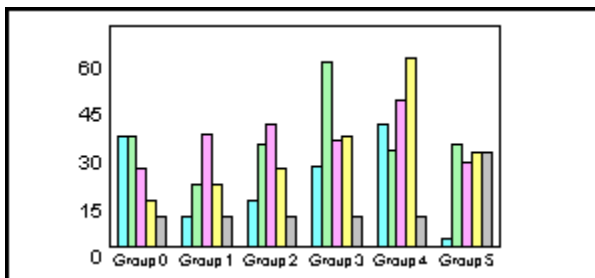
[false](#) includes the minimum label on the O1-axis. This value is the default.

Example:

```
setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false); setO1ExcludeMinLabel(true);
```



[setO1ExcludeMinLabel\(true\);](#)



See [O1ExcludeMaxLabel](#) on page 428, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMinLabel\(\)](#) on page 889.

O1LabelAutoAngle

When O1-Axis or Group labels are long and/or set to a large font size such that they overlap, this property can be used to set the labels to be drawn at an angle.

Syntax:

```
setO1LabelAutoAngle(value);  
integer = getO1LabelAutoAngle();
```

where:

integer

Can be one of the following:

0 for no rotation. 0 is the default value.

1 for 90 degrees rotation.

2 for 270 degrees rotation.

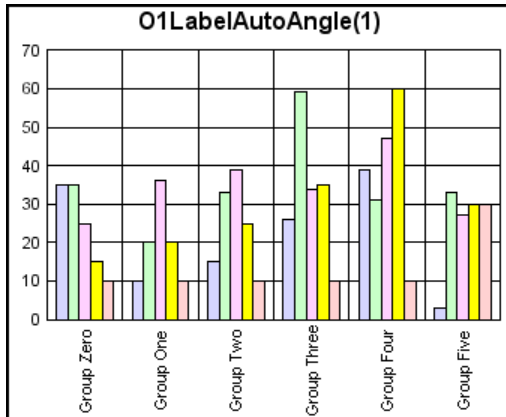
Example:

```
setDepthRadius(0);  
setDepthAngle(0);  
setFootnoteDisplay(false);  
setSubtitleDisplay(false);  
setLegendDisplay(false);
```

```

setGroupLabel(0, "Group Zero");
setGroupLabel(1, "Group One");
setGroupLabel(2, "Group Two");
setGroupLabel(3, "Group Three");
setGroupLabel(4, "Group Four");
setGroupLabel(5, "Group Five");
setO1LabelAutoAngle(1);
setTitleString("O1AutoAngle(1)");

```



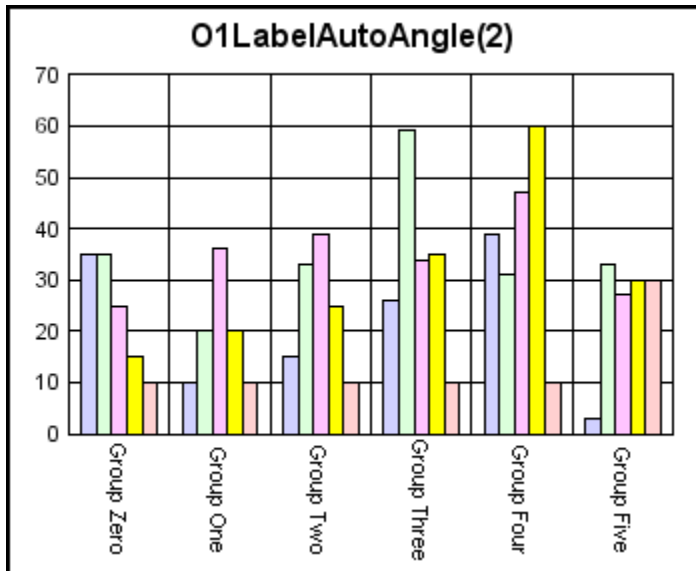
```

setDepthRadius(0);
setDepthAngle(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setLegendDisplay(false);

setGroupLabel(0, "Group Zero");
setGroupLabel(1, "Group One");
setGroupLabel(2, "Group Two");
setGroupLabel(3, "Group Three");
setGroupLabel(4, "Group Four");
setGroupLabel(5, "Group Five");

```

```
setO1LabelAutoAngle(2);
setTitleString("O1AutoAngle(2)");
```



O1LabelAutoSkip

This property sets or disables label skip for labels on the O1 axis.

Syntax:

```
setO1LabelAutoSkip(value);
value = getO1LabelAutoSkip();
```

where:

value

Can be one of the following:

0 does not skip labels. 0 is the default value.

1 automatically omits labels at regular intervals when the graph size is reduced in order to maintain a reasonable, readable font size (8 point or larger).

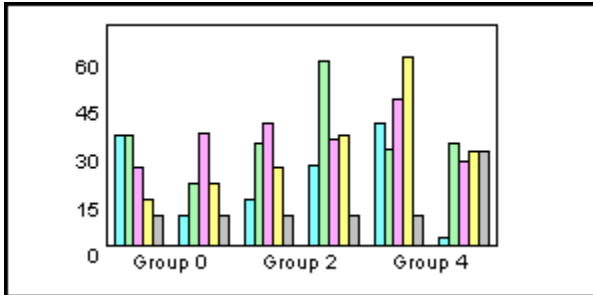
2 sets manual skipping. Use the O1LabelSkipBegin property to define the first label to skip and O1LabelSkipCount to define the skip interval.

Example:

```

setO1LabelSkipBegin(1);
setO1LabelSkipCount(1);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setDepthRadius(0);
setO1LabelAutoSkip(2);

```



See [O1LabelSkipBegin](#) on page 436, [O1LabelSkipCount](#) on page 438.

O1LabelDisplay

This property enables or disables the display of labels and the title on the O1 axis. `O1AxisSide` defines the location where labels are displayed.

Syntax:

```

setO1LabelDisplay(boolean);
boolean = getO1LabelDisplay();

```

where:

boolean

Can be one of the following:

[true](#) displays labels on the O1-axis. This value is the default.

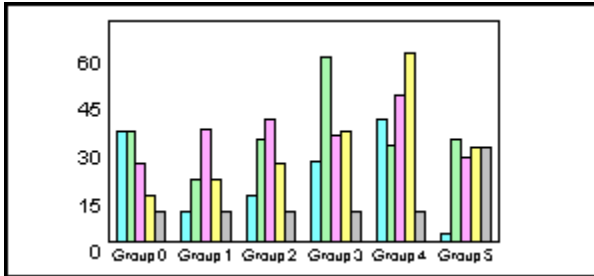
[false](#) does not display labels on the O1-axis.

Example:

```

setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setDepthRadius(0);
setO1LabelDisplay(true);

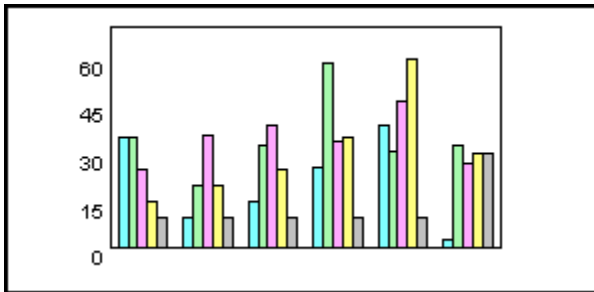
```



```

setO1LabelDisplay(false);

```



See [O1AxisSide](#) on page 425, [O1LabelRotate](#) on page 435, [O1LabelStagger](#) on page 439, [O1LabelWrap](#) on page 441.

O1LabelMargin

This property selects the percentage of the label width that is used as margin between labels on the O1 axis.

Syntax:

```

setO1LabelMargin(value);
value = getO1LabelMargin();

```

where:

value

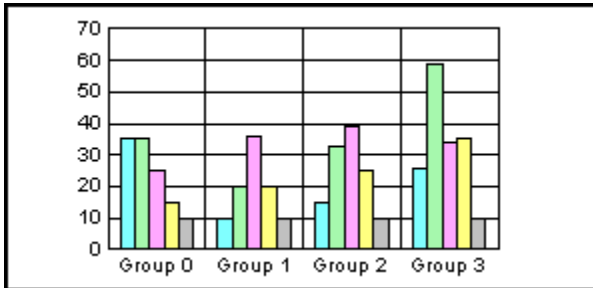
Is any integer between 0 and 100. 0 is the default value.

Example:

```

setDepthRadius(0);
setViewableGroups(4);
setViewableSeries(5);
setDataScrollerPresenceGroups(0);  setO1LabelMargin(10);

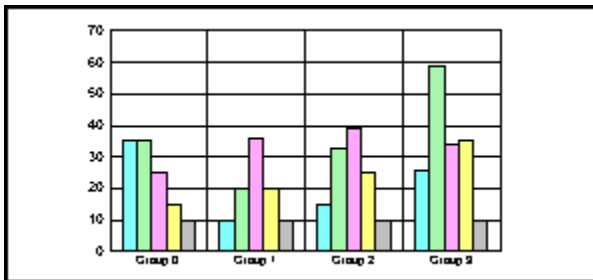
```



```

setO1LabelMargin(50);

```

**O1LabelRotate**

This property selects an angle at which labels on the primary ordinal (O1) axis are rotated.

Syntax:

```

setO1LabelRotate(value);

```

```

value = getO1LabelRotate();

```

where:

value

Can be one of the following:

[0](#) for no text rotation. 0 is the default value.

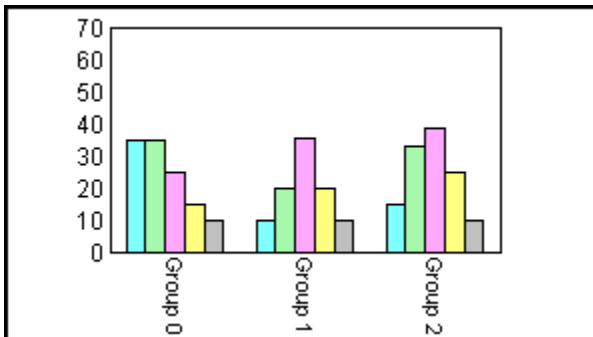
[1](#) for 90 degrees rotation (reads top to bottom).

[2](#) for 270 degrees rotation (reads bottom to top).

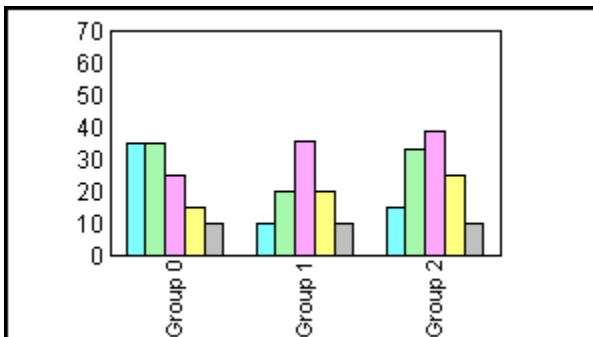
3 for 45 degrees rotation.

Example:

```
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setDepthRadius(0);
setLegendDisplay(false);
setO1LabelRotate(2);
```



setO1LabelRotate(1);



See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [O1LabelStagger](#) on page 439, [O1LabelWrap](#) on page 441.

O1LabelSkipBegin

When O1LabelAutoSkip is set to two (manual), this property defines the first label on the O1-Axis to be omitted. Use the O1LabelSkipCount property to define the label skip interval.

Syntax:

```
setO1LabelSkipBegin(value);
value = getO1LabelSkipBegin();
```

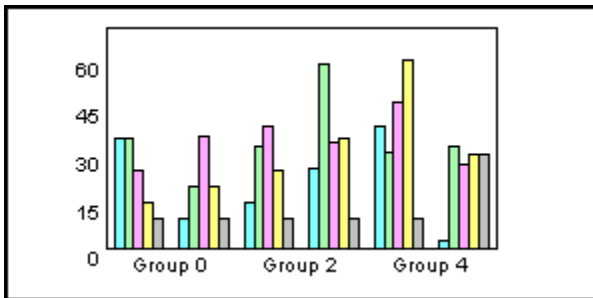
where:

value

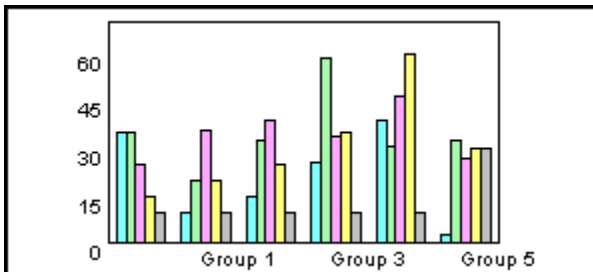
Is any integer between 0 and the number of groups in the graph. 0 is the default value.

Example:

```
setO1LabelSkipCount(1);
setO1LabelAutoSkip(2);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
setDepthRadius(0);
setO1LabelSkipBegin(1);
```



```
setO1LabelSkipCount(1); setO1LabelSkipBegin(0);
```



See [O2LabelAutoSkip](#) on page 457, [O2LabelSkipCount](#) on page 463.

O1LabelSkipCount

When O1LabelAutoSkip is set to manual (2), this property defines the label skip interval. Use O1LabelSkipBegin to define the first label to be omitted.

Syntax:

```
setO1LabelSkipCount(value);  
value = getO1LabelSkipCount();
```

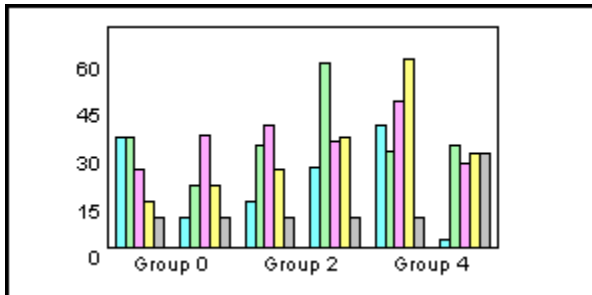
where:

value

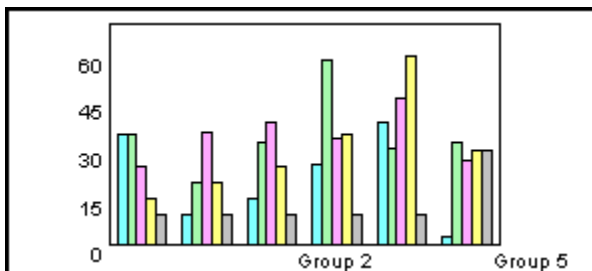
Is any integer between 0 and the number of groups in the graph. 0 is the default value.

Example:

```
setO1LabelSkipBegin(1);  
setO1LabelAutoSkip(2);  
setY1MajorGridDisplay(false);  
setY1AxisLineDisplay(false);  
setO1MajorGridDisplay(false);  
setDepthRadius(0);  
  setO1LabelSkipCount(1);
```



```
setO1LabelSkipBegin(0);  
setO1LabelSkipCount(2);
```



See [O1LabelAutoSkip](#) on page 432, [O1LabelSkipBegin](#) on page 436.

O1LabelStagger

This property enables or disables staggered labels on the O1 axis.

Syntax:

```
setO1LabelStagger(boolean);  
boolean = getO1LabelStagger();
```

where:

boolean

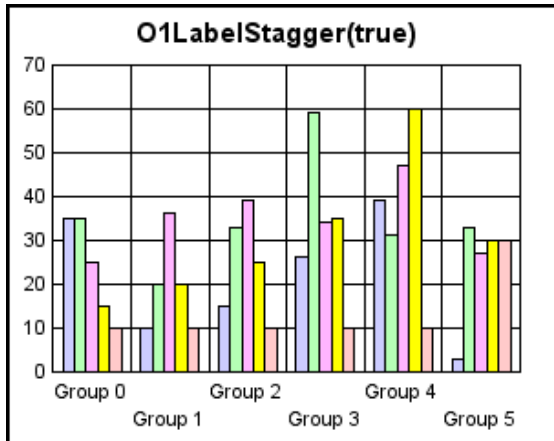
Can be one of the following:

[true](#) staggers labels on the O1-axis.

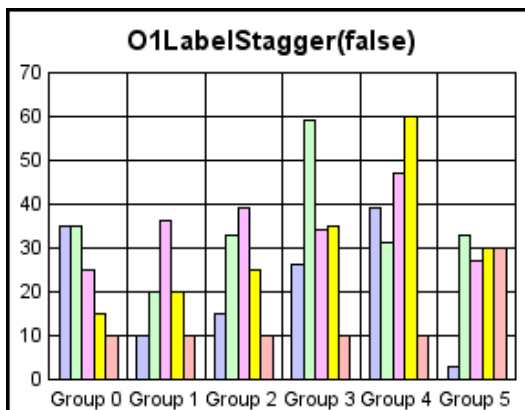
[false](#) does not stagger labels. This value is the default.

Example:

```
setDepthRadius(0);  
setDepthAngle(0);  
setFootnoteDisplay(false);  
setSubtitleDisplay(false);  
setLegendDisplay(false);  
    setO1LabelStagger(true);  
setTitleString("O1LabelStagger(true)");
```



```
setDepthRadius(0);  
setDepthAngle(0);  
setFootnoteDisplay(false);  
setSubtitleDisplay(false);  
setLegendDisplay(false);  
    setO1LabelStagger(false);  
setTitleString("O1LabelStagger(false)");
```



Note: This property only applies to 2D graphs, staggered labels are not supported in 3D graphs.

See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [O1LabelRotate](#) on page 435, [O1LabelWrap](#) on page 441, [getLabelStagger\(\)](#) on page 764, [setLabelStagger\(\)](#) on page 918.

O1LabelWrap

This property controls whether header labels on the primary ordinal (O1) axis are drawn on more than one line.

Syntax:

```
setO1LabelWrap(boolean);
```

```
boolean = getO1LabelWrap();
```

where:

boolean

Can be one of the following:

[true](#) wraps labels on the O1-axis. The graphing engine looks for space characters in the label string and draws the label on multiple lines at space character boundaries.

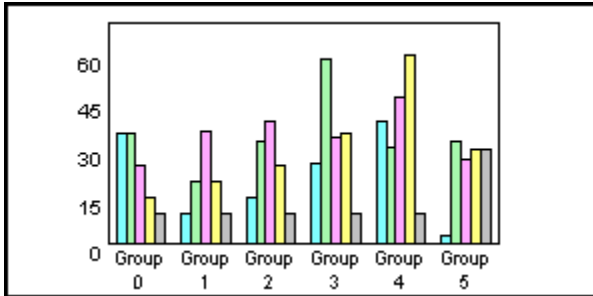
[false](#) draws labels on a single line. This value is the default.

Example:

```

setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setO1MajorGridDisplay(false);
  setO1LabelWrap(true);

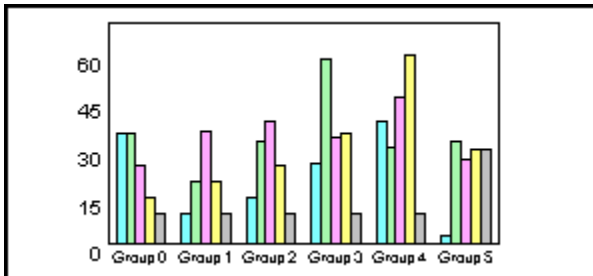
```



```

setO1LabelWrap(false);

```



Note: If a "\n" is encountered in the label string, the string is wrapped even if this property is disabled (setO1LabelWrap(false);).

See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [O1LabelRotate](#) on page 435, [O1LabelStagger](#) on page 439.

O1MajorGridDisplay

This property enables or disables the display of major grid lines on the primary ordinal (O1) axis grid (that is, lines attached to a text-based axis, like the "bottom" axis of a bar graph). The O1AxisSide property determines the location where grids and ticks are displayed. Use O1MajorGridStyle to define the style of major grid lines. The primary ordinal axis can have major and minor grid lines.

Syntax:

```
setO1MajorGridDisplay(boolean);  
boolean = getO1MajorGridDisplay();
```

where:

boolean

Can be one of the following:

[true](#) draws major grid lines on the O1-axis. This value is the default.

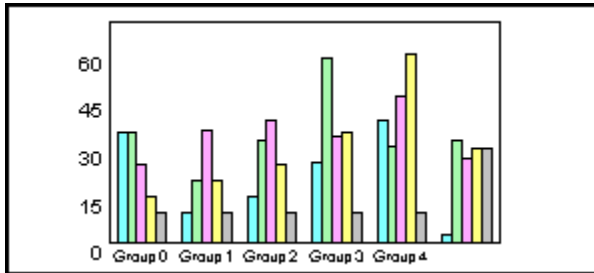
[false](#) does not draw major grid lines.

Example:

```

setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);  setO1MajorGridDisplay(true);

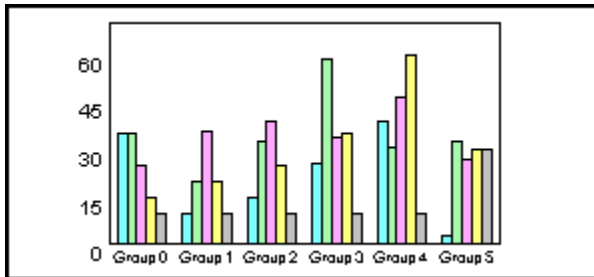
```



```

setO1MajorGridDisplay(false);

```



See [O1AxisSide](#) on page 425, [O1MajorGridStyle](#) on page 444, [O1MinorGridDisplay](#) on page 447, [O1MinorGridStyle](#) on page 449, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

O1MajorGridStyle

This property controls the style of major grid lines that are shown on the primary ordinal (O1) axis (that is, lines attached to a text-based axis, like the "bottom" axis of a bar graph). The primary ordinal axis can have major and minor grid lines. Different styles should be selected for major and minor grid lines so the user can distinguish between the two.

Syntax:

```

setO1MajorGridStyle(value);
value = getO1MajorGridStyle();

```

where:

value

Can be one of the following:

[0](#) for normal grid lines, height of frame. 0 is the default value.

[1](#) for normal grid lines extending beyond the height of frame.

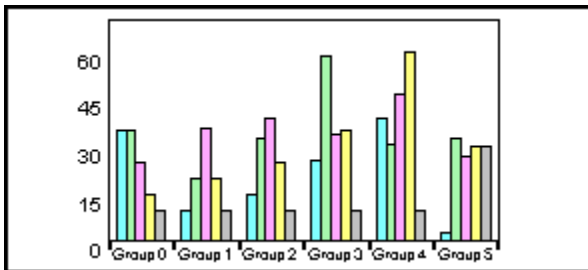
[2](#) for small tick marks from frame edge inward.

[3](#) for small tick marks from frame edge outward.

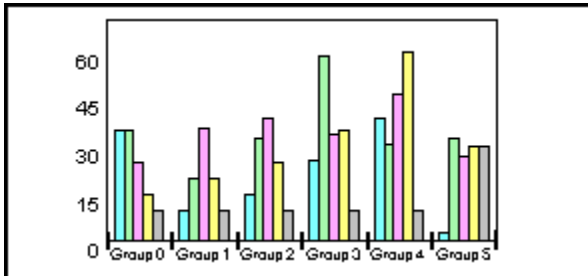
[4](#) for small tick marks spanning across the frame edge.

Example:

```
setDepthRadius(0);
setY1MajorGridDisplay(false);
setY1AxisLineDisplay(false);
setLineWidth(getO1MajorGrid(),4);  setO1MajorGridStyle(3);
```



```
setO1MajorGridStyle(4);
```



See [O1AxisSide](#) on page 425, [O1MajorGridDisplay](#) on page 443, [O1MinorGridDisplay](#) on page 447, [O1MinorGridStyle](#) on page 449, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

O1MajorTickDisplay

This property enables or disables drawing of major tick marks on the primary ordinal (O1) axis.

Syntax:

```
setO1MajorTickDisplay(boolean);
```

```
boolean = getO1MajorTickDisplay ();
```

where:

boolean

Can be one of the following:

[true](#) draws major tick marks on the O1-axis. This value is the default.

[false](#) does not draw major tick marks.

For an example, see [O1MinorTickCount](#) on page 450.

O1MajorTickStyle

This property controls the style of major tick marks that can be drawn on the primary ordinal (O1) axis with `setO1MajorTickDisplay (true)` .

Syntax:

```
setO1MajorTickStyle(value);
```

```
value = getO1MajorTickStyle();
```

where:

value

Can be one of the following:

[0](#) for small tick marks from frame edge outward. This value is the default.

[1](#) for small tick marks from frame edge outward.

[2](#) for small tick marks from frame edge inward.

[3](#) for small tick marks from frame edge outward.

[4](#) for small tick marks span across the frame edge.

Also see [O1MinorTickCount](#) on page 450.

O1MinorGridCount

This property controls the number of minor grid lines between the major grid lines on the O1 axis.

Syntax:

```
setO1MinorGridCount(value);
```

```
value = getO1MinorGridCount();
```

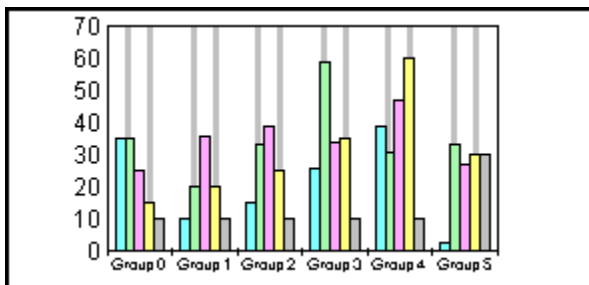
where:

value

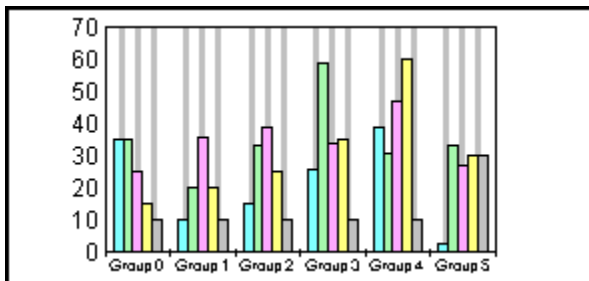
Is any integer between 1 and 50. 1 is the default value.

Example:

```
setDepthRadius(0);
setY1MajorGridDisplay(false);
setO1MajorGridStyle(4);
setO1MinorGridDisplay(true);
setLineWidth(getO1MinorGrid(),4);
setBorderColor(getO1MinorGrid(),new Color(255,0,0));
setO1MinorGridCount(2);
```



`setO1MinorGridCount(3);`



See [O1MinorGridDisplay](#) on page 447, [O1MinorGridStyle](#) on page 449, [getGridCount\(\)](#) on page 757, [setGridCount\(\)](#) on page 909.

O1MinorGridDisplay

This property enables or disables the display of minor grid lines on the primary ordinal (O1) axis grid (that is, lines attached to a text-based axis, like the "bottom" axis of a bar graph). Use `O1MinorGridStyle` to define the style of minor grid lines. The primary ordinal axis can have major and minor grid lines.

Syntax:

```
setO1MinorGridDisplay(boolean);
boolean = getO1MinorGridDisplay();
```

where:

boolean

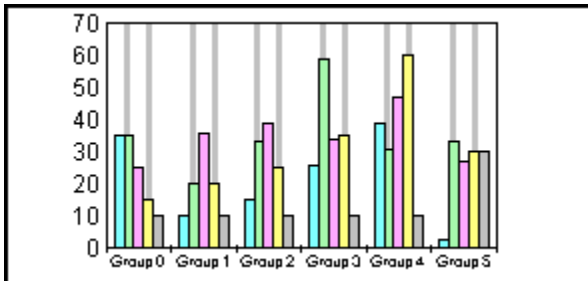
Can be one of the following:

`true` draws minor grid lines on the O1-axis.

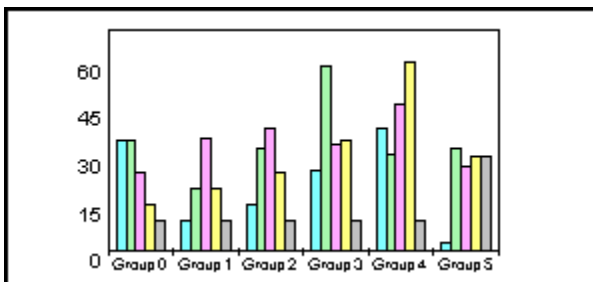
`false` does not draw minor grid lines. This value is the default.

Example:

```
setDepthRadius(0);
setY1MajorGridDisplay(false);
setO1MajorGridStyle(4);
setLineWidth(getO1MinorGrid(),4);
setBorderColor(getO1MinorGrid(),new Color(255,0,0));
setO1MinorGridCount(2);
setO1MinorGridDisplay(true);
```



`setO1MinorGridDisplay(false);`



See [O1AxisSide](#) on page 425, [O1MajorGridDisplay](#) on page 443, [O1MajorGridStyle](#) on page 444, [O1MinorGridStyle](#) on page 449.

O1MinorGridStyle

This property controls the style of the minor grid lines on the primary ordinal (O1) axis. The primary ordinal axis can have major and minor grid lines. Different styles should be selected for major and minor grid lines so the user can distinguish between the two.

Syntax:

```
setO1MinorGridStyle(value);  
value = getO1MinorGridStyle();
```

where:

value

Can be one of the following:

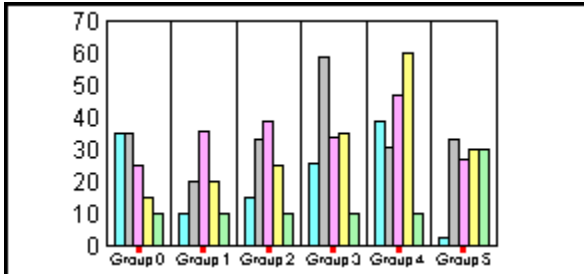
- 0 for normal grid lines, height of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```

setDepthRadius(0);
setY1MajorGridDisplay(false);
setO1MajorGridStyle(0);
setO1MinorGridDisplay(true);
setLineWidth(getO1MinorGrid(),4);
setBorderColor(getO1MinorGrid(),new Color(0,255,0));
setO1MinorGridStyle(3);

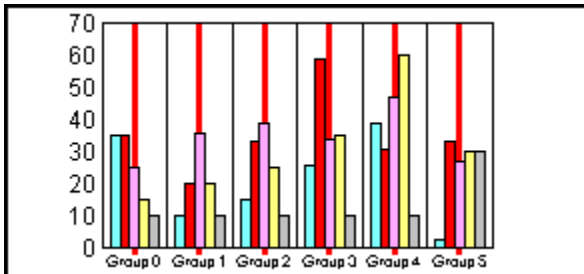
```



```

setO1MinorGridStyle(1);

```



See [O1AxisSide](#) on page 425, [O1MajorGridDisplay](#) on page 443, [O1MinorGridDisplay](#) on page 447, [O1MajorGridStyle](#) on page 444, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

O1MinorTickCount

This property controls the number of minor tick marks to draw between major tick marks on the O1 axis.

Syntax:

```

setO1MinorTickCount(value);
value = getO1MinorTickCount();

```

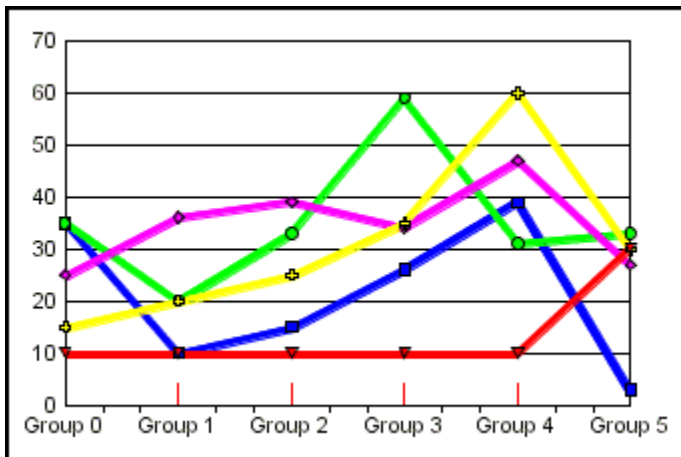
where:

value

Is an integer between 1 and 50. 1 is the default value.

Example:

```
setDepthRadius(0);
setDepthAngle(0);
setGraphType(41);
setO1MajorGridDisplay(false);
setO1MajorTickDisplay(true);
setO1MinorTickDisplay(true);
setTickLength(getO1MajorTick(), 1000);
setBorderColor(getO1MajorTick(), new Color (255,0,0));
setO1MinorTickCount(2);
setO1MajorTickStyle(2);
setO1MinorTickStyle(1);
```



O1MinorTickDisplay

This property enables or disables drawing of minor tick marks on the primary ordinal (O1) axis.

Syntax:

```
setO1MinorTickDisplay (boolean);
boolean = getO1MinorTickDisplay ();
```

where:

boolean

Can be one of the following:

[true](#) draws minor tick marks on the O1-axis. This value is the default.

[false](#) does not draw minor tick marks.

For an example see [O1MinorTickCount](#) on page 450.

O1MinorTickStyle

This property controls the style of minor tick marks that can be drawn on the primary ordinal (O1) axis with `setO1MinorTickDisplay (true)` .

Syntax:

```
setO1MinorTickStyle(value);  
value = getO1MinorTickStyle();
```

where:

value

Can be one of the following:

[0](#) for small tick marks from frame edge outward. This value is the default.

[1](#) for small tick marks from frame edge outward.

[2](#) for small tick marks from frame edge inward.

[3](#) for small tick marks from frame edge outward.

[4](#) for small tick marks span across the frame edge.

For an example, see [O1MinorTickCount](#) on page 450.

O1TitleDisplay

This property determines whether or not the string of characters assigned to the `O1TitleString` property is displayed in the graph.

Note: The `O1LabelDisplay` property must also be set to true for the axis title to display.

Syntax:

```
setO1TitleDisplay(boolean);  
boolean = getO1TitleDisplay();
```


where:

boolean

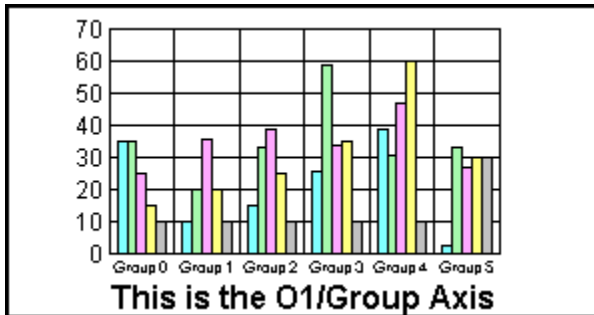
Can be one of the following:

true draws O1-axis title.

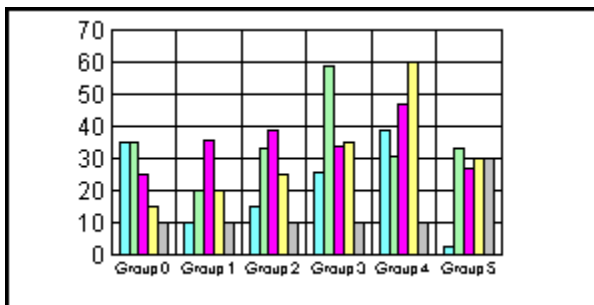
false does not draw title. This value is the default.

Example:

```
setDepthRadius(0);
setLegendDisplay(false);
setO1TitleString("This is the O1/Group Axis");
setFontSize(getO1Title(),8);
setO1TitleAutofit(false);
setO1TitleDisplay(true);
```



```
setO1TitleDisplay(false);
```



Note: If `O1LabelDisplay` is false, the O1-axis title will not be drawn regardless of the `O1TitleDisplay` or `O1TitleString` properties.

See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [O1TitleString](#) on page 454.

O1TitleString

This property defines the O1-axis title string displayed by the O1TitleDisplay property.

Syntax:

```
setO1TitleString(string);
string = getO1TitleString();
```

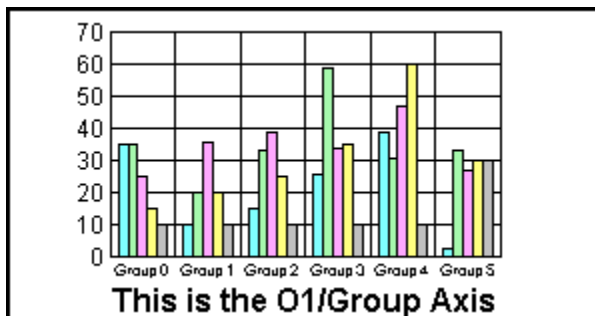
where:

string

Represents any printable characters. Null string is the default value.

Example:

```
setDepthRadius(0);
setLegendDisplay(false);
  setO1TitleString("This is the O1/Group Axis");
setTitleDisplay(true);
setFontSize(getO1Title(),8);
setO1TitleAutofit(true);
```



Note: The O1AxisSide property determines where the O1TitleString is drawn in graph. Also note that setting any string to O1TitleString will automatically set O1TitleDisplay to true.

See [O1AxisSide](#) on page 425, [O1LabelDisplay](#) on page 433, [O1TitleDisplay](#) on page 452.

O2ExcludeMaxLabel

This property enables or disables drawing of the maximum label on the series (O2) axis.

Syntax:

```
setO2ExcludeMaxLabel(boolean);
boolean = getO2ExcludeMaxLabel();
```

where:

boolean

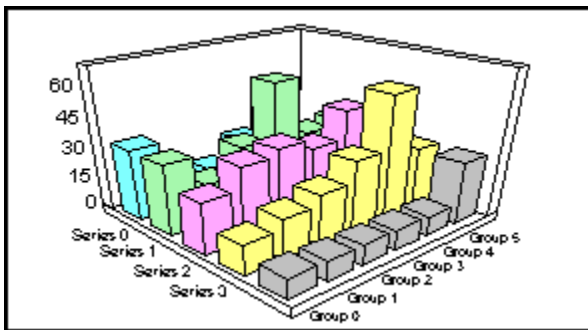
Can be one of the following:

true excludes the maximum label on the O2-axis

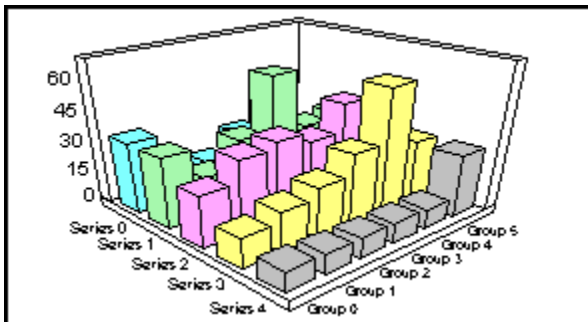
false includes the maximum label on the O2-axis. This value is the default.

Example:

```
setO2ExcludeMaxLabel(true);
setGraphType(0);
setAutoshading(false);
```



```
setO2ExcludeMaxLabel(false);
```



See [O1ExcludeMinLabel](#) on page 429, [getExcludeMaxLabel\(\)](#) on page 744, [setExcludeMaxLabel\(\)](#) on page 888.

O2ExcludeMinLabel

This property enables or disables drawing of the minimum label on the series (O2) axis.

Syntax:

```
setO2ExcludeMinLabel(boolean);  
boolean = getO2ExcludeMinLabel();
```

where:

boolean

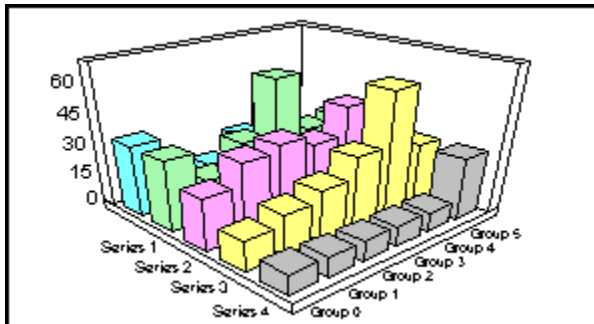
Can be one of the following:

true excludes the minimum label on the O2-axis.

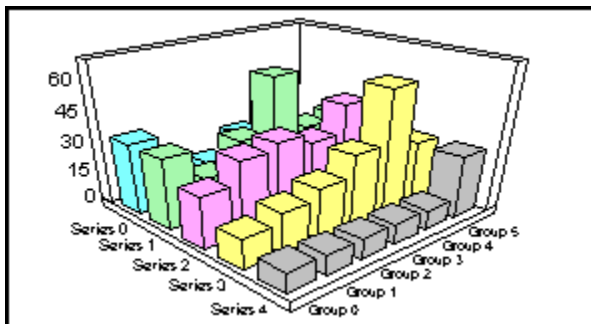
false includes the minimum label on the O2-axis. This value is the default.

Example:

```
setO2ExcludeMinLabel(true);  
setGraphType(0);  
setAutoshading(false);
```



```
setO2ExcludeMinLabel(false);
```



See [O2ExcludeMaxLabel](#) on page 454, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMinLabel\(\)](#) on page 889.

O2LabelAutoSkip

This property sets or disables label skip for labels on the O2/series axis labels in a 3D graph.

Syntax:

```
setO2LabelAutoSkip(value);
```

```
value = getO2LabelAutoSkip();
```

where:

value

Can be one of the following:

[0](#) does not skip labels. 0 is the default value.

[1](#) automatically omits labels at regular intervals when the graph size is reduced in order to maintain a reasonable, readable font size (8 point or larger).

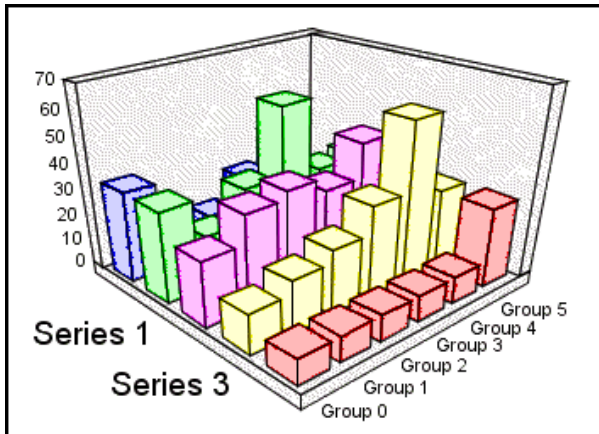
[2](#) sets manual skipping. Use the [O2LabelSkipBegin](#) property to define the first label to skip and [O2LabelSkipCount](#) to define the skip interval.

Example:

```

setTitleDisplay(false);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setGraphType(0);
setO2LabelSkipBegin(1);
setO2LabelSkipCount(1);
    setO2LabelAutoSkip(2);

```



See [O2LabelSkipBegin](#) on page 461, [O2LabelSkipCount](#) on page 463.

O2LabelDisplay

This property enables or disables the display of labels and the O2 title on the series (O2) axis. The O2AxisSide property defines the location where labels are displayed.

Syntax:

```

setO2LabelDisplay(boolean);
boolean = getO2LabelDisplay();

```

where:

boolean

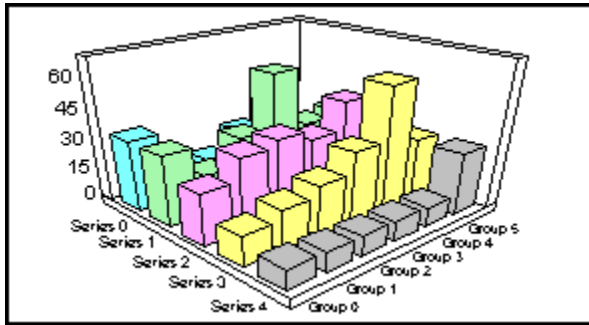
Can be one of the following:

[true](#) displays labels on the O2-axis. This value is the default.

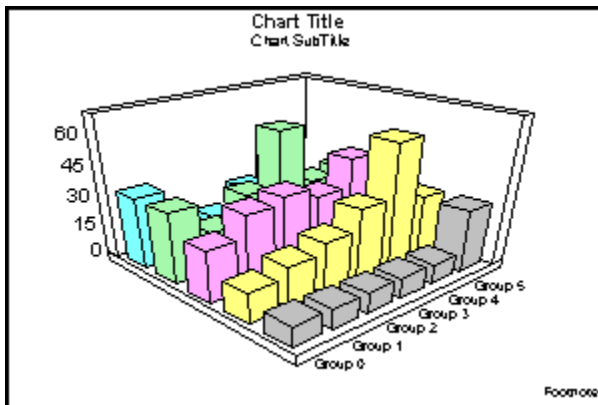
[false](#) does not display labels.

Example:

```
setGraphType(0);
  setO2LabelDisplay(true);
setAutoshading(false);
```



```
setO2LabelDisplay(false);
```



See [O2LabelRotate](#) on page 459.

O2LabelRotate

This property selects an angle at which labels on the series (O2) axis are rotated.

Syntax:

```
setO2LabelRotate(value);
value = getO2LabelRotate();
```

where:

value

Can be one of the following:

[0](#) for no text rotation. 0 is the default value.

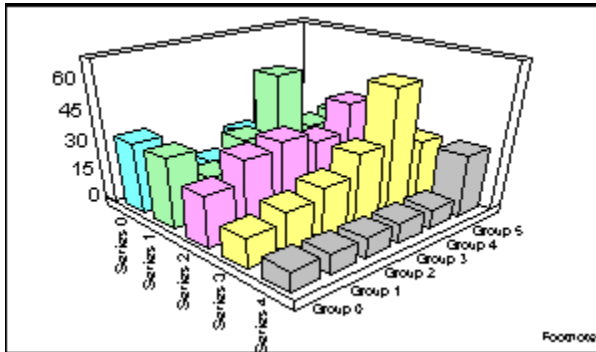
1 rotates text 90 degrees (reads top to bottom).

2 rotates text 270 degrees (reads bottom to top).

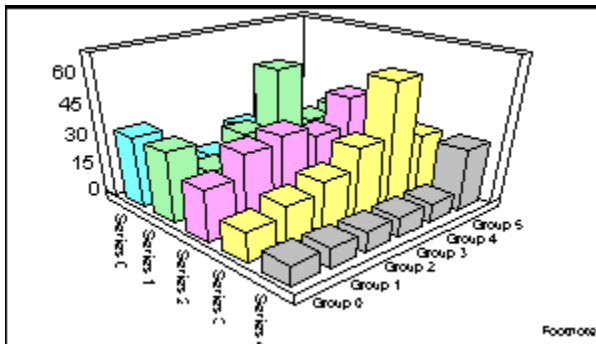
3 rotates text 45 degrees.

Example:

```
setGraphType(0);
setAutoshading(false);
setO2LabelRotate(1);
```



```
setO2LabelRotate(2);
```



See [O2LabelDisplay](#) on page 458, [O2LabelWrap](#) on page 463.

O2LabelSkipBegin

When O2LabelAutoSkip is set to two (manual), this property defines the first label on the O2-Axis to be omitted. Use the O2LabelSkipCount property to define the label skip interval.

Syntax:

```
setO2LabelSkipBegin(value);
```

```
value = getO2LabelSkipBegin();
```

where:

value

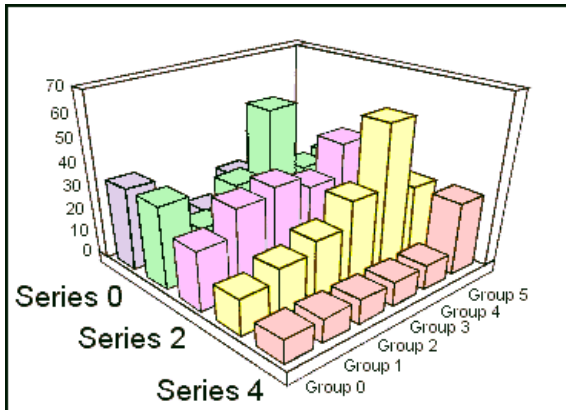
Is any integer between 0 and the number of groups in the graph. 0 is the default value.

Example:

```

setTitleDisplay(false);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setGraphType(0);
    setO2LabelSkipBegin(0);
setO2LabelSkipCount(1);
setO2LabelAutoSkip(2);

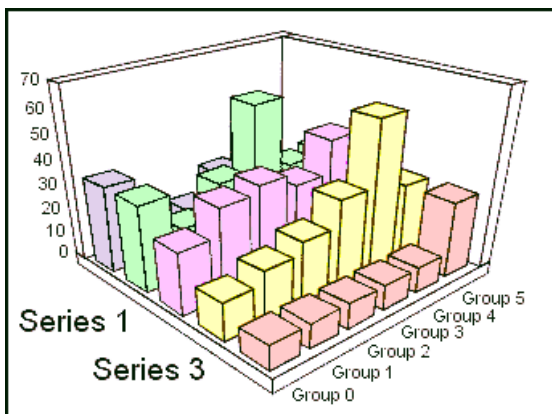
```



```

setTitleDisplay(false);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setGraphType(0);
    setO2LabelSkipBegin(1);
setO2LabelSkipCount(1);
setO2LabelAutoSkip(2);

```



See [O2LabelAutoSkip](#) on page 457, [O2LabelSkipCount](#) on page 463.

O2LabelSkipCount

When O2LabelAutoSkip is set to manual (2), this property defines the label skip interval. Use O2LabelSkipBegin to define the first label to be omitted.

Syntax:

```
setO2LabelSkipCount(value);  
value = getO2LabelSkipCount();
```

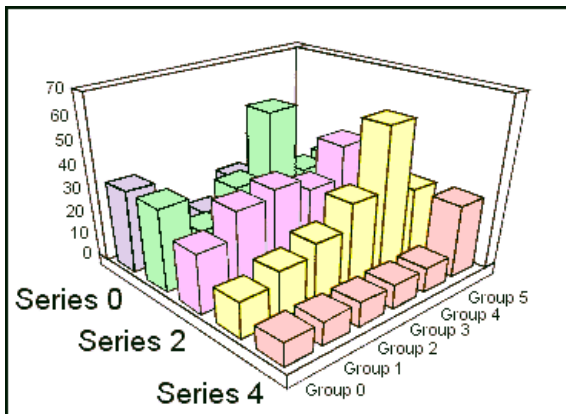
where:

value

Is any integer between 0 and the number of groups in the graph. 0 is the default value.

Example:

```
setTitleDisplay(false);  
setSubtitleDisplay(false);  
setFootnoteDisplay(false);  
setGraphType(0);  
setO2LabelSkipBegin(0);  
    setO2LabelSkipCount(1);  
setO2LabelAutoSkip(2);
```



See [O1LabelAutoSkip](#) on page 432, [O1LabelSkipBegin](#) on page 436.

O2LabelWrap

This property controls whether header labels may have more than 1 line on a series (O2) axis.

Syntax:

```
setO2LabelWrap(boolean);
boolean = getO2LabelWrap();
```

where:

boolean

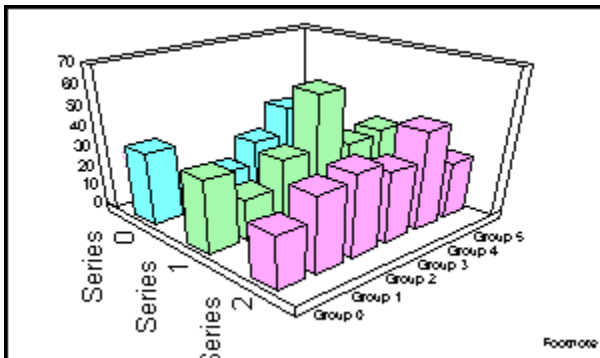
Can be one of the following:

[true](#) wraps labels on the O2-axis. The graphing engine looks for space characters in the label string and draws the label on multiple lines at space character boundaries.

[false](#) draws labels on a single line. This value is the default.

Example:

```
setO2LabelAutofit(false);
setFontSizeVC(getO2Label(),2086);
setGraphType(0);
setO2LabelWrap(true);
```



Note: If a "\n" is encountered in the label string, the string is wrapped even if this property is disabled (setO2LabelWrap(false);).

See [O2LabelDisplay](#) on page 458, [O2LabelRotate](#) on page 459.

O2TitleDisplay

This property determines whether or not the string of characters assigned to the O2TitleString property is displayed in the graph. Note that the O2LabelDisplay property must also be set to true in order for the axis title to display.

Syntax:

```
setO2TitleDisplay(boolean);
boolean = getO2TitleDisplay();
```

where:

boolean

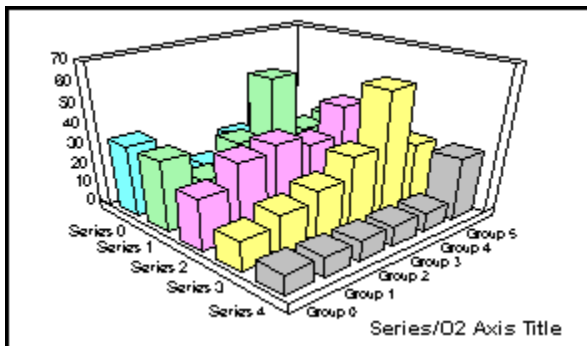
Can be one of the following:

`true` draws O2-axis title.

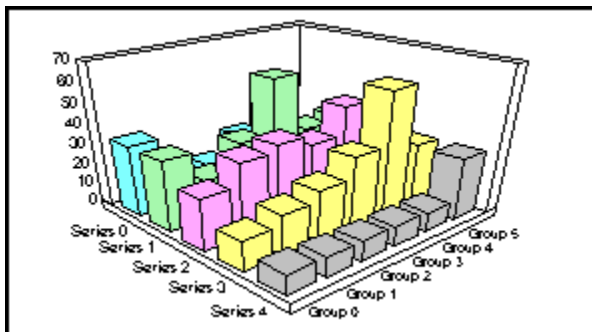
`false` does not draw title. This value is the default.

Example:

```
setO2TitleDisplay(true);
setGraphType(0);
setO2TitleString("Series/O2 Axis Title");
```



```
setO2TitleDisplay(false);
```



Note: If O2LabelDisplay is false, the O2-axis title will not be drawn regardless of the O2TitleDisplay or O2TitleString properties.

O2TitleString

This property defines the O2-axis title string displayed by the O2TitleDisplay property.

Syntax:

```
setO2TitleString("string");
string = getO2TitleString();
```

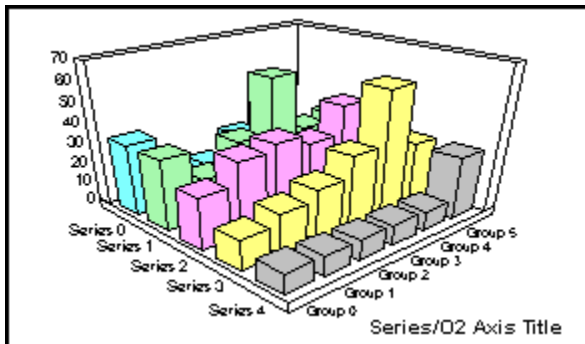
where:

string

Represents any printable characters. Null string is the default.

Example:

```
setFootnoteDisplay(false);
setO2TitleDisplay(true);
setFontStyle(getO2Title(),2);
setGraphType(0);
    setO2TitleString("Series/O2 Axis Title");
```



Note: The O2AxisSide property determines where the O2TitleString is drawn in graph. Also note that setting any string to O2TitleString will automatically set O2TitleDisplay to true.

See [O2LabelDisplay](#) on page 458, [O2TitleDisplay](#) on page 464.

ParetoDisplayThreshold

This property sets the location to split the graph frame and accumulation line in a pareto graph.

Syntax:

```
setParetoDisplayThreshold(value);  
value = getParetoDisplayThreshold();
```

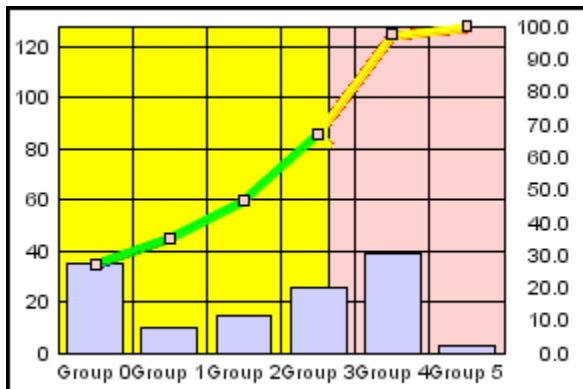
where:

value

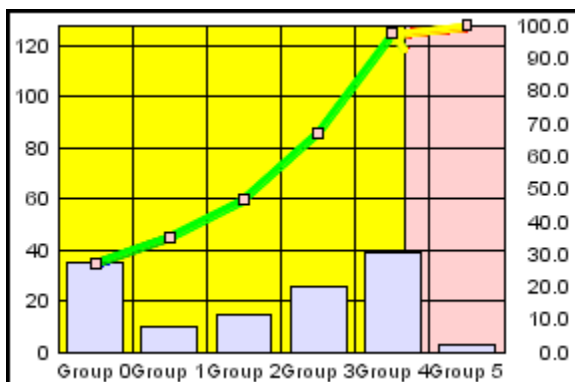
Is any integer between 0 and 100. 1 is the default value.

Example:

```
setTitleDisplay(false);  
setSubtitleDisplay(false);  
setFootnoteDisplay(false);  
setDepthAngle(0);  
setDepthRadius(0);  
setGraphType(102);  
    setParetoDisplayThreshold(50);
```



```
setTitleDisplay(false);  
setSubtitleDisplay(false);  
setFootnoteDisplay(false);  
setDepthAngle(0);  
setDepthRadius(0);  
setGraphType(102);  
    setParetoDisplayThreshold(90);
```



ParetoY2MajorGridStep

This property sets the interval of grid lines and labels on the Y2-axis in a pareto graph.

Syntax:

```
setParetoY2MajorGridStep(value);  
value = getParetoY2MajorGridStep();
```

where:

value

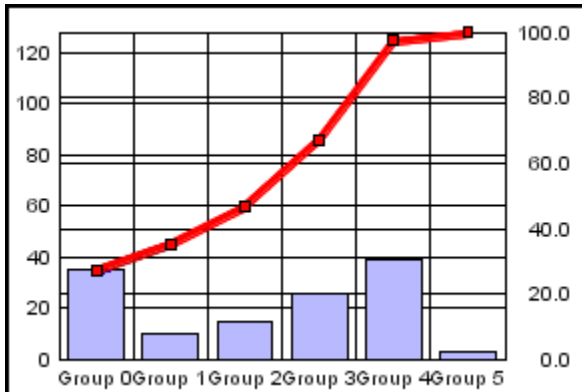
Is a value between 1.0 and 500.0. This value is a Java double-precision number. The default value is 10.0.

Example:

```

setTitleDisplay(false);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setDepthAngle(0);
setDepthRadius(0);
setGraphType(102);
setParetoDisplayThreshold(0);  setParetoY2MajorGridStep(20.0);

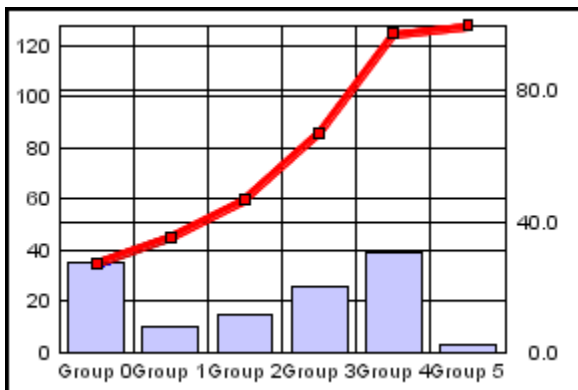
```



```

setTitleDisplay(false);
setSubtitleDisplay(false);
setFootnoteDisplay(false);
setDepthAngle(0);
setDepthRadius(0);
setGraphType(102);
setParetoDisplayThreshold(0);  setParetoY2MajorGridStep(40.0);

```



ParetoY2MajorTickStep

This property sets the interval of tick marks on the Y2-axis in a pareto graph.

Syntax:

```
setParetoY2MajorTickStep(value);
value = getParetoY2MajorTickStep();
```

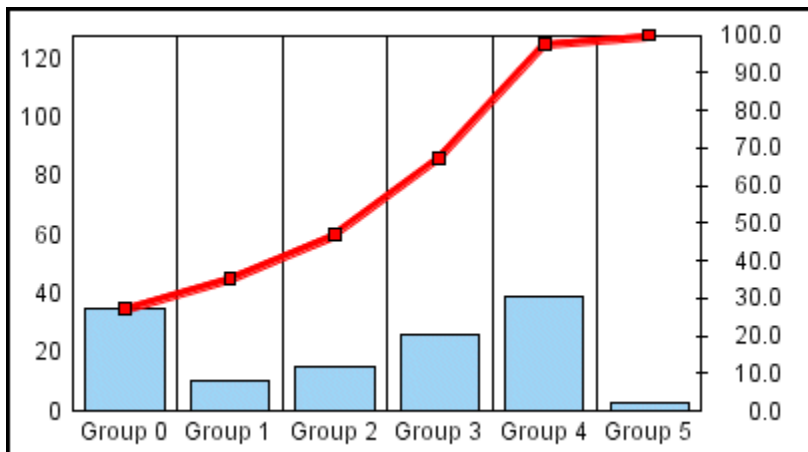
where:

value

Is a value between 1.0 and 500.0. This value is a Java double-precision number. The default value is 10.0.

Example:

```
setGraphType(102);
setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY2MajorGrid(), false);
setDisplay(getY1MajorGrid(), false);  setParetoY2MajorTickStep(10.0);
setDisplay(getY2MajorTick(), true);
```



PieBarLabelDisplay

When the GraphType property selects a Pie-Bar graph (setGraphType(93);), this property enables or disables the pie-bar label display.

Syntax:

```
PieBarLabelDisplay(boolean);
```

```
boolean = getPieBarLabelDisplay();
```

where:

boolean

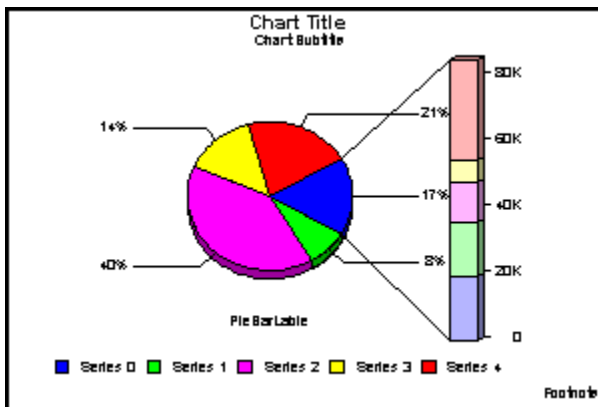
Can be one of the following:

true displays Pie-Bar graph label. This value is the default.

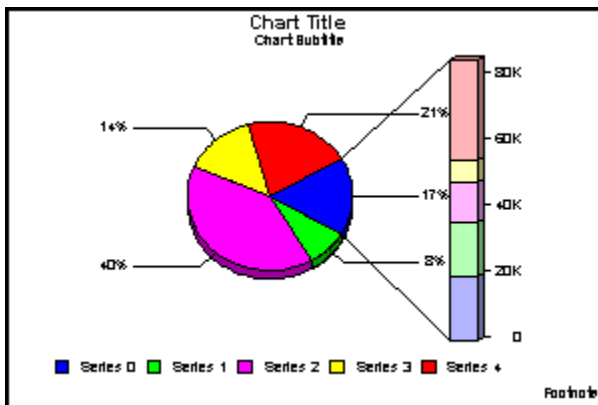
false does not display Pie-Bar graph label.

Example:

```
setPieBarLabelDisplay(true);
setTextString(getPieBarLabel(), "PieBarLabel");
setGraphType(93);
```



```
setPieBarLabelDisplay(false);
setTextString(getPieBarLabel(), "PieBarLabel");
setGraphType(93);
```



See [PieBarLabelString](#) on page 473.

PieBarLabelString

When the GraphType property selects a Pie-Bar graph (setGraphType(93);), this property defines the Pie-Bar label string displayed by the PieBarLabelDisplay property.

Syntax:

```
PieBarLabelString("string");
string = getPieBarLabelString();
```

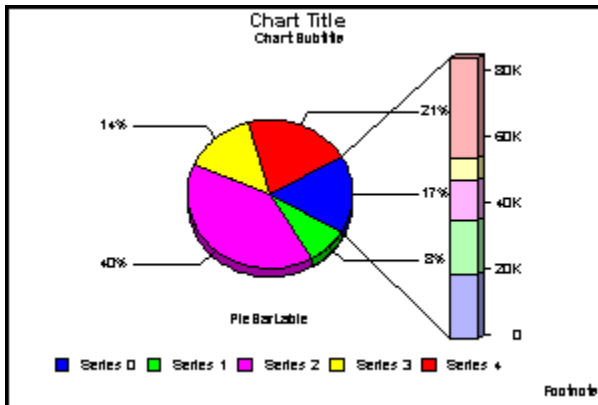
where:

string

Represents any printable characters. Null string is the default.

Example:

```
setPieBarLabelDisplay(true); setPieBarLabelString("PieBarLabel");
setGraphType(93);
```



See [PieBarLabelDisplay](#) on page 471.

PieBarSeries

When the GraphType property selects a Pie-Bar graph (setGraphType(93);), this property selects the series from which the bar is graphed.

Syntax:

```
setPieBarSeries(value);
value = getPieBarSeries();
```

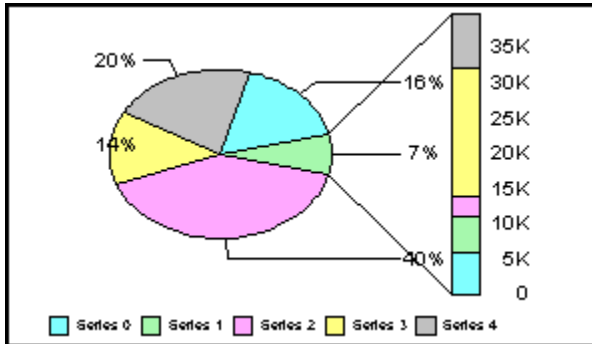
where:

value

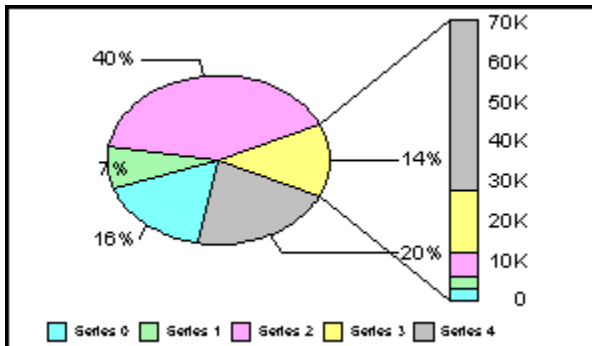
Is any integer between 0 and the number of series in the graph. 0 is the default value.

Example:

```
setGraphType(93);
setPieBarSeries(1);
```



```
setGraphType(93);
setPieBarSeries(3);
```



PieDepth

This property specifies the depth of the crust in a pie graph or a funnel graph. The GraphType property must be set to one of the pie graphs (55...60) or the funnel graph (106).

Syntax:

```
setPieDepth(value);
value = getPieDepth();
```

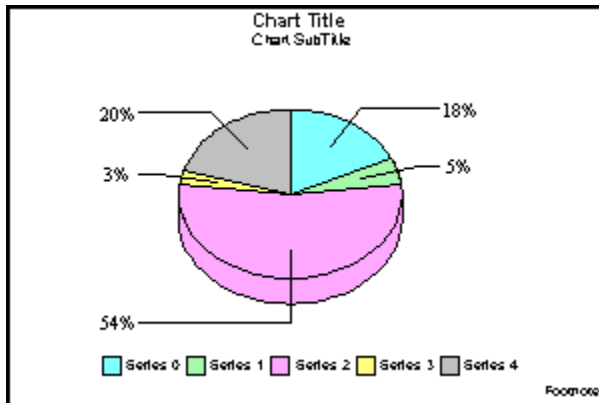
where:

value

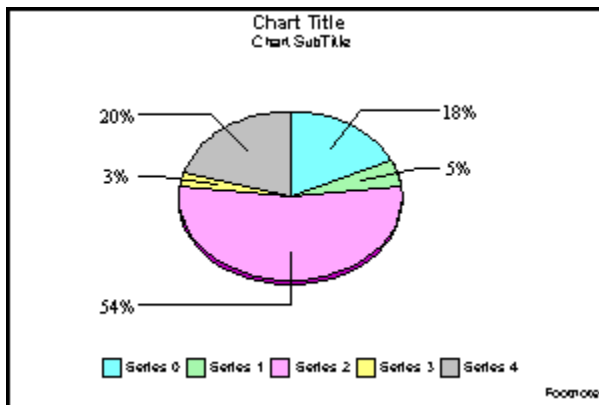
Is any integer between 0 and 100. 30 is the default value.

Pie Graph Example:

```
setGraphType(55);  
setPieDepth(90);
```



```
setPieDepth(15);
```

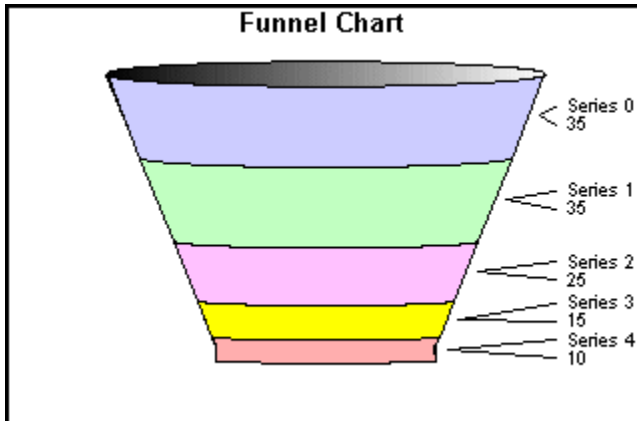


Funnel Graph Example:

```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
setPieRotate(50);
setPieTilt(25);
setPieDepth(100);

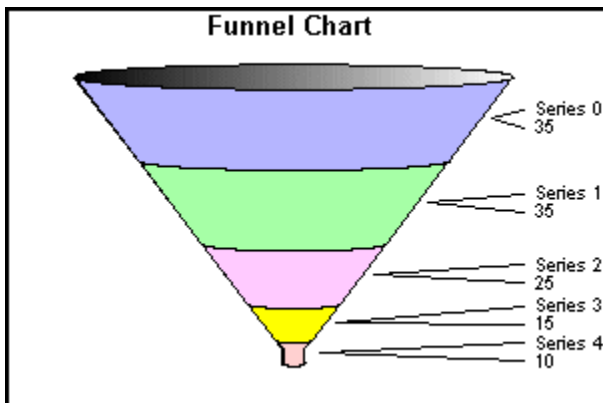
```



```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
setPieRotate(50);
setPieTilt(25);
setPieDepth(10);

```



See [PieRotate](#) on page 488, [PieTilt](#) on page 493.

PieFeelerTextDisplay

This property enables or disables the display of feelers and data text in a pie graph.

Syntax:

```
setPieFeelerTextDisplay(value);  
value = getPieFeelerTextDisplay();
```

where:

value

Can be one of the following:

0

For no labels.

1

To show feeler lines and text. 1 is the default value.

2

For labels only, no feeler lines.

3

For labels on slices.

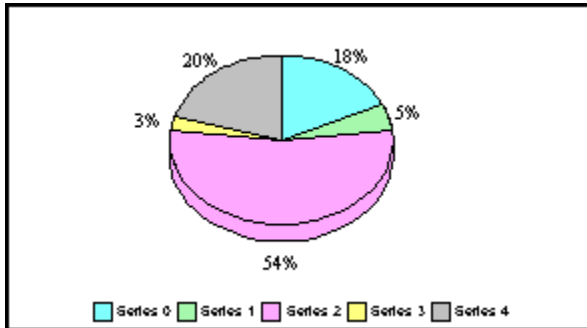
4

For smart positioning. If there is enough space to fit the label inside the slice, it will go inside the slice. If there is not enough space the pie label would be outside with feeler lines.

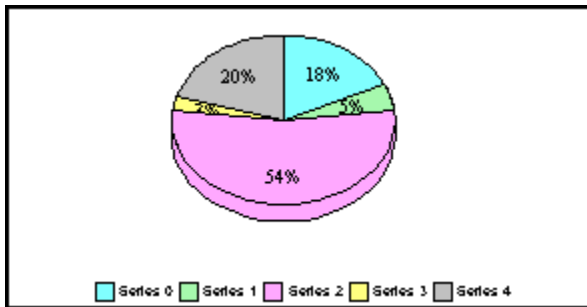
Note: This value is not supported in HTML5 graphs.

Example:

```
setGraphType(55);
  setPieFeelerTextDisplay(2);
setPieDepth(60);
```



```
setPieFeelerTextDisplay(3);
```



Note: The `PieFeelerTextFormat`, `PieFeelerTextFormatPattern`, and `PieLabelDisplay` properties determine the format of data text displayed next to feelers when this property is enabled.

See [PieFeelerTextFormat](#) on page 478, [PieFeelerTextFormatPattern](#) on page 479, [PieLabelDisplay](#) on page 481.

PieFeelerTextFormat

This property is used to select a standard preset format for data values that may be drawn next to feelers in a pie graph. Set this property to minus one and use `PieFeelerTextFormatPattern` if you want to define a standard-Java format pattern. The `PieLabelDisplay` property determines the information that is displayed (for example: percent value, absolute values, series label). When this property is used, the `GraphType` property must be set to one of the pie graphs (55...60).

Syntax:

```
setPieFeelerTextFormat(value);
value = getPieFeelerTextFormat();
```

where:

value

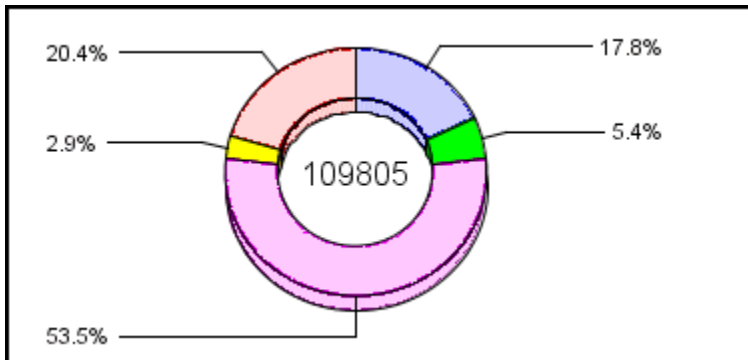
Is any integer between -1 and 21.

-1 uses the pattern defined by `PieFeelerTextFormatPattern`.

0 through 21 selects a preset number format. 2 is the default value (#%). See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setPieTilt(5);
setPieRingTotalFormat(1);
setGraphType(56);
setPieRingSize(60);
setPieFeelerTextFormat(3);
```



See [PieFeelerTextDisplay](#) on page 477, [PieFeelerTextFormatPattern](#) on page 479, [PieLabelDisplay](#) on page 481.

PieFeelerTextFormatPattern

When `PieFeelerTextFormat` is set to minus one (custom), this property is used to specify a standard Java number format pattern for data text values.

Syntax:

```
setPieFeelerTextFormatPattern(pattern);
pattern = getPieFeelerTextFormatPattern();
```

where:

pattern

Is a pattern string in the following format:

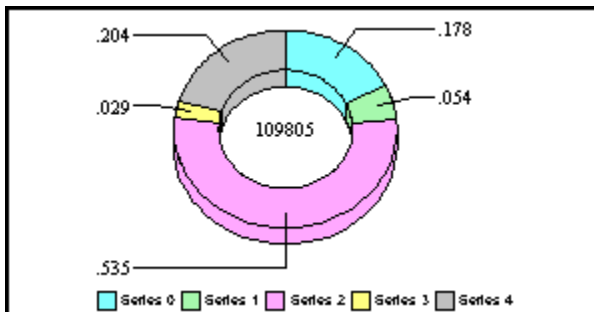
```
pattern:= subpattern{;subpattern}subpattern:=
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFD' -
specialCharacterssuffix:= '\\u0000'..'\\uFFFD' -
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

The default is "#.#"

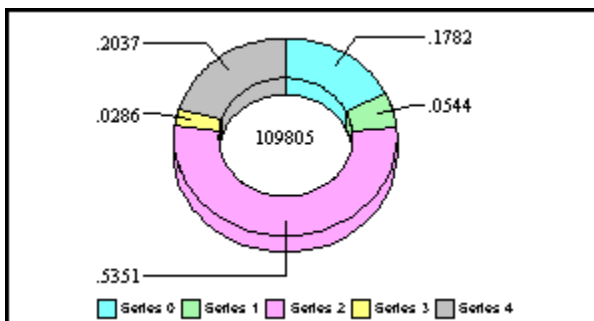
See [Formatting Numbers and Numeric Labels](#) on page 130 for more detailed information about this format pattern.

Example:

```
setPieFeelerTextFormatPattern("##.####");
setGraphType(56);
setPieRingSize(60);
setPieFeelerTextFormat(-1);
```



```
setPieFeelerTextFormatPattern("##.#####");
```



See [PieFeelerTextDisplay](#) on page 477, [PieFeelerTextFormat](#) on page 478, [PieLabelDisplay](#) on page 481.

PieLabelDisplay

This property determines the format of labels displayed next to feelers in a pie graph. The GraphType property must be set to one of the pie graphs (55...60).

Syntax:

```
setPieLabelDisplay(value);
```

```
value = getPieLabelDisplay();
```

where:

value

Can be one of the following:

0 for the absolute (true) value of the slice.

1 for the percent value of the slice. 1 is the default value.

2 for the series label.

3 for the series label and percent value.

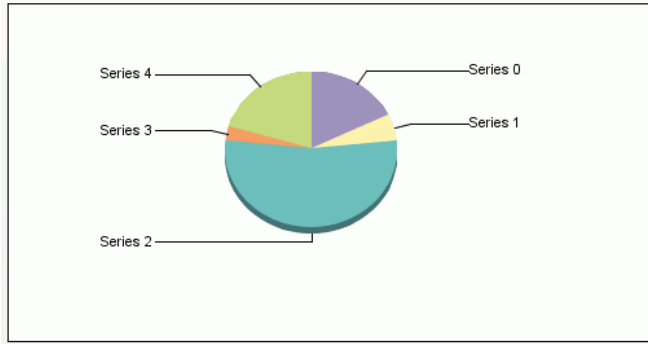
4 for the series label and absolute value.

5 for the absolute value and percent value.

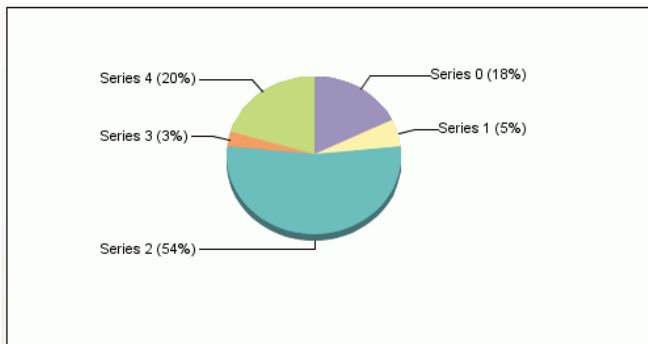
6 for the absolute value, percent value, and series label.

Example:

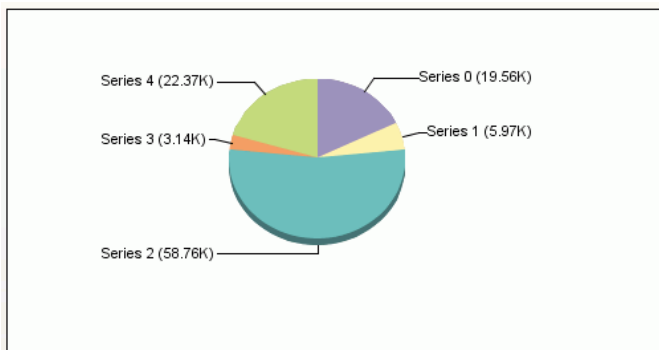
```
setGraphType(55);  
setPieLabelDisplay(2);
```



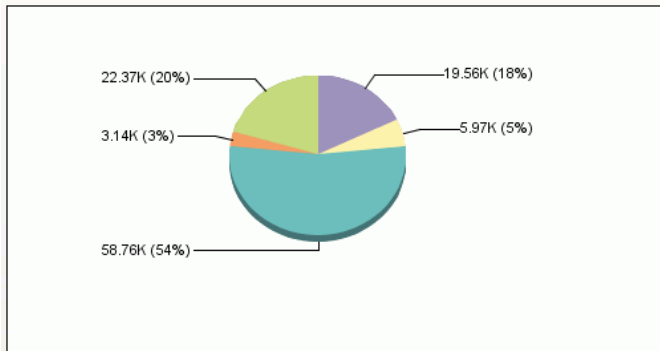
```
setPieLabelDisplay(3);
```



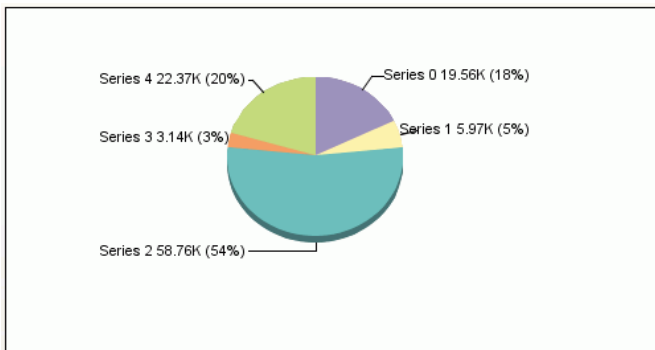
```
setPieLabelDisplay(4);
```



```
setPieLabelDisplay(5);
```



`setPieLabelDisplay(6);`



Note: For values 5 and 6, use objects `getPieSliceFeelerValue()` and `getPieSliceFeelerPercent()`.

See [PieFeelerTextDisplay](#) on page 477, [PieFeelerTextFormat](#) on page 478, [PieFeelerTextFormatPattern](#) on page 479, [getPieSliceFeelerValue\(\)](#) on page 1109, [getPieSliceFeelerPercent\(\)](#) on page 1108.

PieRingSize

This property determines the relative size of the ring (inner circle) in a ring pie. The `GraphType` property must be set to one of the pie ring graphs (56, 58, or 60).

Syntax:

`setPieRingSize(value);`

`value = getPieRingSize();`

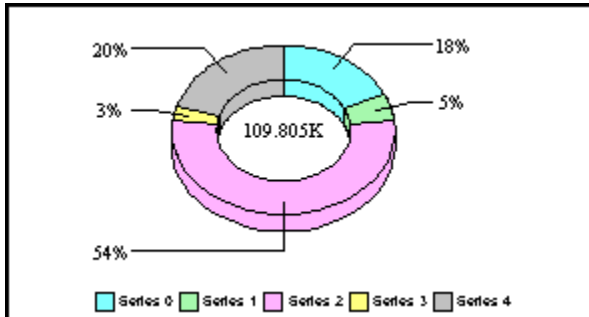
where:

value

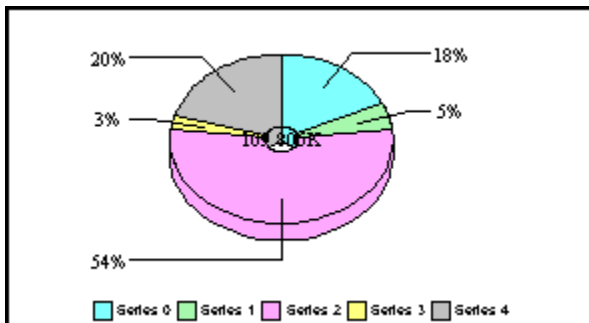
Is any integer between 0 and 100. 30 is the default value.

Example:

```
setGraphType(56);
setPieRingSize(60);
```



```
setPieRingSize(15);
```



See [PieRingTotalDisplay](#) on page 484, [PieRingTotalFormat](#) on page 486, [PieRingTotalFormatPattern](#) on page 487.

PieRingTotalDisplay

This property enables or disables the display of a total value in the center of a pie ring graph. The GraphType property must be set to one of the pie ring graphs (56, 58, or 60). The PieRingTotalFormat property determines the format of the total value that is displayed.

Syntax:

```
setPieRingTotalDisplay(boolean);
```



```
boolean = getPieRingTotalDisplay();
```

where:

boolean

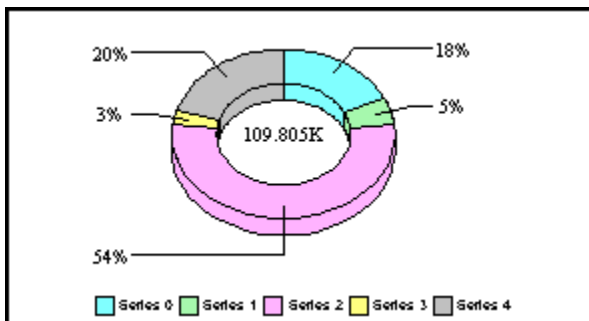
Can be one of the following:

[true](#) displays total. This value is the default.

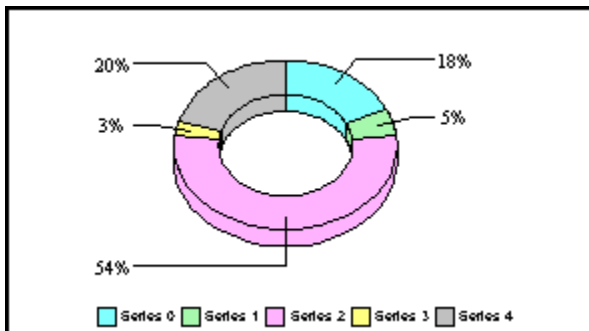
[false](#) does not display total.

Example:

```
setGraphType(56);
  setPieRingTotalDisplay(true);
setPieRingSize(60);
```



```
setPieRingTotalDisplay(false);
```



See [PieRingSize](#) on page 483, [PieRingTotalFormat](#) on page 486, [PieRingTotalFormatPattern](#) on page 487.

PieRingTotalFormat

When the `PieRingTotalDisplay` property is set to true and the total data value is displayed in the center of a ring pie graph, this property can be used to define the format of the data value. This property selects one of the standard preset formats. See the `PieRingTotalFormatPattern` property to specify a standard Java number format pattern.

Syntax:

```
setPieRingTotalFormat(value);
value = getPieRingTotalFormat();
```

where:

value

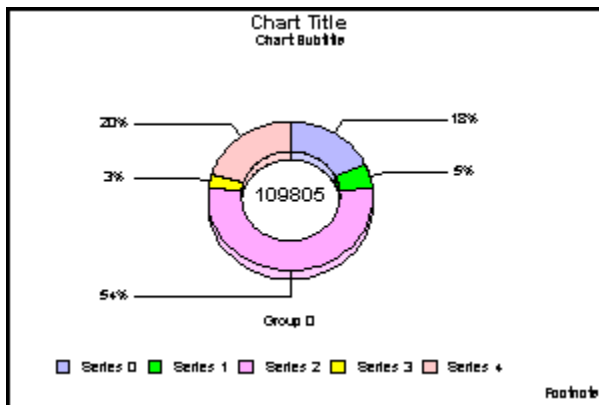
Is any integer between -1 and 21.

-1 uses the pattern defined by `PieRingTotalFormatPattern`.

0 through 21 selects a preset number format. 0 is the default value. See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setPieRingTotalFormat(1);
setGraphType(56);
setPieRingSize(60);
```



See [PieRingSize](#) on page 483, [PieRingTotalDisplay](#) on page 484, [PieRingTotalFormatPattern](#) on page 487.

PieRingTotalFormatPattern

When the `PieRingTotalDisplay` property is set to true and the total data value is displayed in the center of a ring pie graph, this property can be used to define the format of the data value. This property is used to specify a standard Java number format pattern. See the `PieRingTotalFormat` property to select one of the standard preset formats.

Syntax:

```
setPieRingTotalFormatPattern(patternString);  
patternString = getPieRingTotalFormatPattern();
```

where:

patternString

Is a pattern string in the following format:

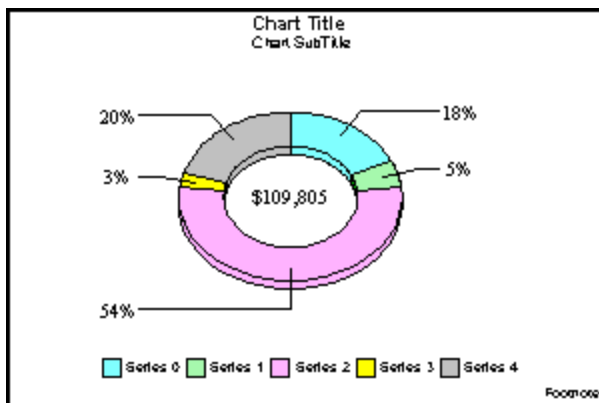
```
pattern:= subpattern{;subpattern}subpattern:=  
{prefix}integer{.fraction}{suffix}  
  prefix:= '\\u0000'..'\\uFFFD'  
  - specialCharacterssuffix:= '\\u0000'..'\\uFFFD'  
  - specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

The default is "#.#"

See [Formatting Numbers and Numeric Labels](#) on page 130 for more detailed information about this format pattern.

Example:

```
setPieRingTotalFormat(-1);  
setGraphType(56);  
setPieRingSize(60); setPieRingTotalFormatPattern("$###,###");
```



See [PieRingSize](#) on page 483, [PieRingTotalDisplay](#) on page 484, [PieRingTotalFormat](#) on page 486.

PieRotate

This property rotates a pie or a funnel graph a specified number of degrees.

Syntax:

```
setPieRotate(value);
```

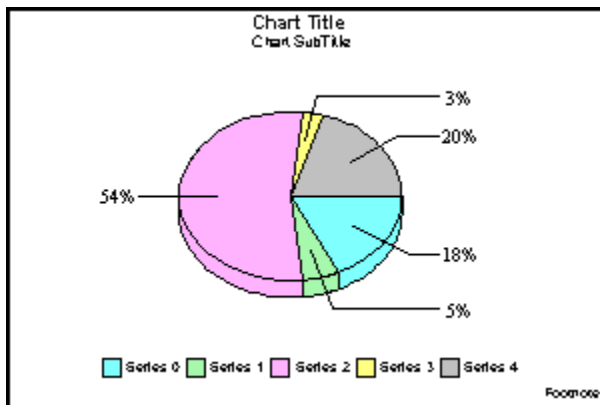
```
value = getPieRotate();
```

value

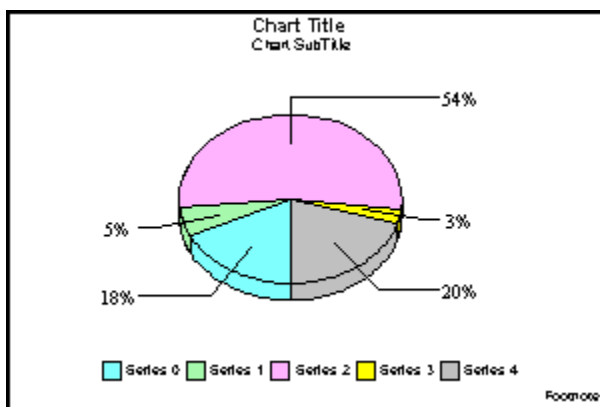
Is any integer between 0 and 359. 0 is the default value.

Pie Graph Example:

```
setGraphType(55);
setPieRotate(90);
```



```
setPieRotate(180);
```

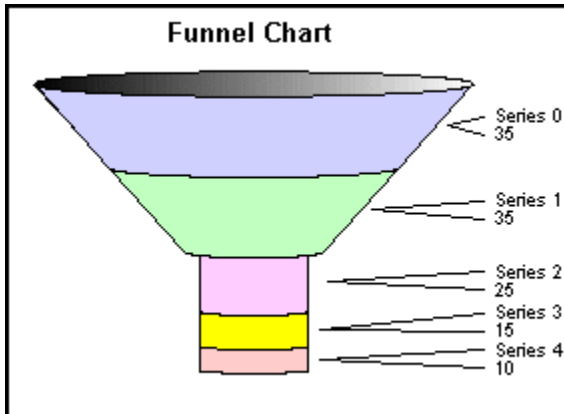


Funnel Graph Example:

```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
    setPieRotate(359);
setPieTilt(25);
setPieDepth(50);

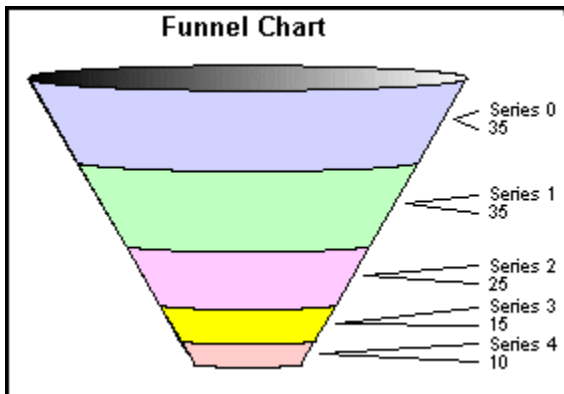
```



```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
    setPieRotate(10);
setPieTilt(25);
setPieDepth(50);

```



See [PieDepth](#) on page 474, [PieTilt](#) on page 493.

PieSorting

This property determines the order in which pie slices are drawn.

Syntax:

```
setPieSorting(value);  
value = getPieSorting();
```

where:

value

Can be one of the following:

[0](#) for no sorting, leave the original order. This value is the default.

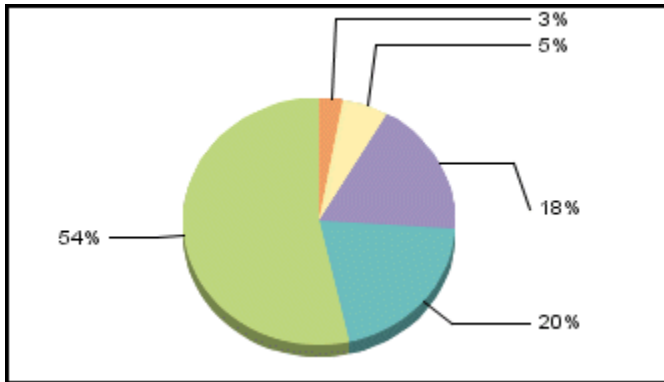
[1](#) for ascending order.

[2](#) for descending order

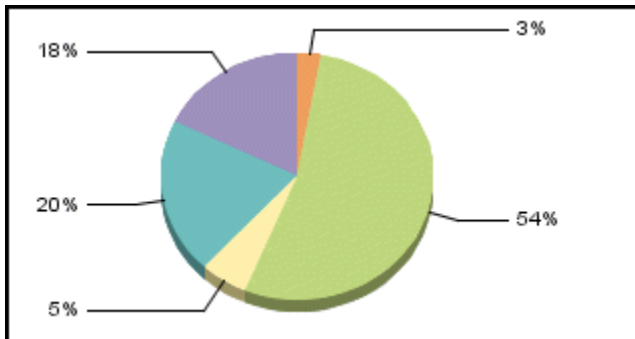
[3](#) for alternate (largest slice, then smallest slice, then 2nd largest slice, then 2nd smallest slice, and so on). This is the best sorting to avoid label overlap when there are a lot of slices.

Example:

```
setGraphType(55);  
setPieSorting(1);
```



```
setGraphType(55);  
setPieSorting(3);
```

**PiesPerRow**

This property specifies the number of pies to be drawn in a row in multi-pie graphs.

Syntax:

```
setPiesPerRow(value);  
value = getPiesPerRow();
```

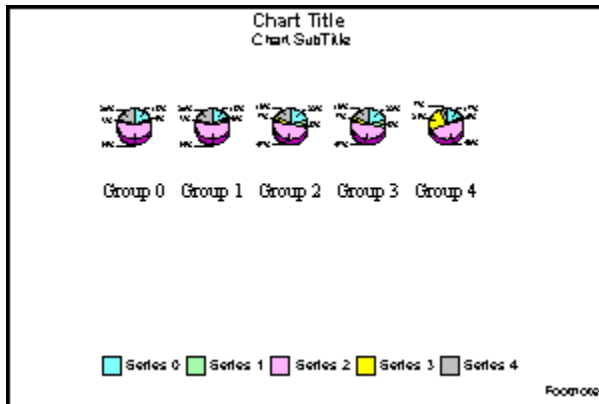
where:

value

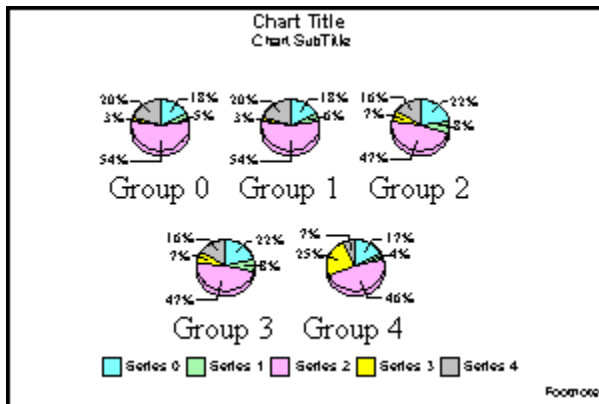
Is any integer between 1 and 32. 2 is the default value.

Example:

```
setPieTilt(5);
setGraphType(57);
  setPiesPerRow(5);
```



```
setPiesPerRow(3);
```

**PieTilt**

This property tilts a pie or a funnel graph a specified number of degrees.

Syntax:

```
setPieTilt(value);
value = getPieTilt();
```

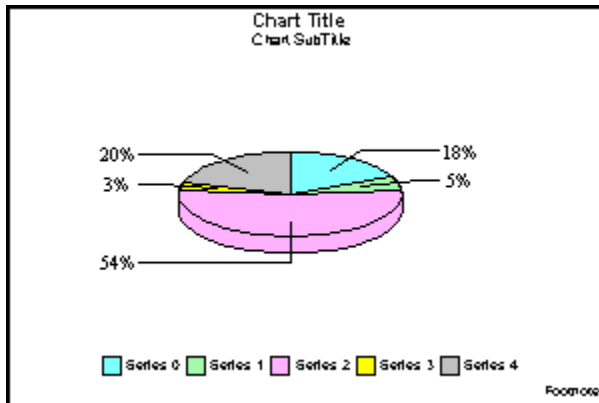
where:

value

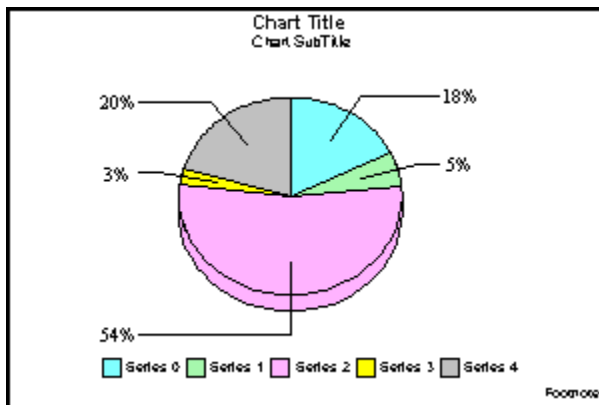
Is any integer between 0 and 100. 10 is the default value.

Pie Graph Example:

```
setPieTilt(45);  
setGraphType(55);
```



```
setPieTilt(5);
```

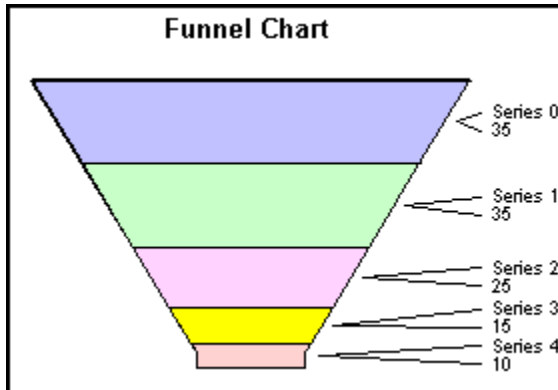


Funnel Graph Example:

```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
setPieRotate(50);
    setPieTilt(1);
setPieDepth(50);

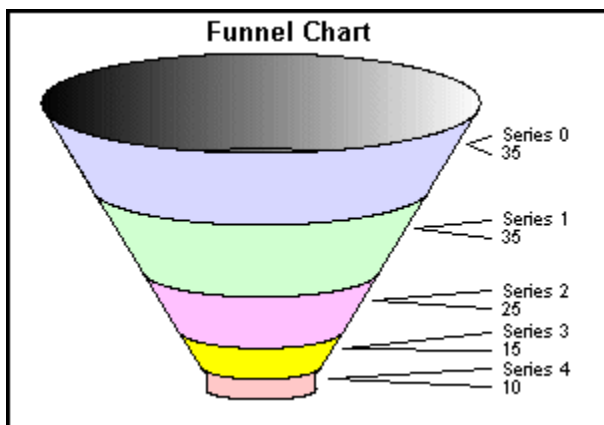
```



```

setTransparentBorderColor(getFrame(), true);
setGraphType(106);
setTitleString("Funnel Chart");
setSubtitleDisplay(false);
setPieRotate(50);
    setPieTilt(100);
setPieDepth(50);

```



See [PieDepth](#) on page 474, [PieRotate](#) on page 488.

Place

The Place property enables or disables the automatic graph layout manager. This functionality allows you to automatically set up the spacing of various elements of a graph. Prior to drawing a graph, certain methods may be set depending on what the automatic graph layout manager determines the optimum graph to be.

Syntax:

```
setPlace(boolean);
```

```
boolean = getPlace();
```

where:

boolean

Can be one of the following:

[true](#) enables automatic graph layout manager.

[false](#) disables automatic graph layout manager. This value is the default.

When setPlace (true) is used, you can use the following methods to further control automatic placement of graph elements:

- ❑ setPlaceAlign(); For any text object, allow or disallow horizontal alignment. By default, horizontal alignment is allowed.
- ❑ setPlaceMargin(); Define the margin to draw around the entire graph. The default value is default new Rectangle (-15500, -15500, 31000, 31000).
- ❑ setPlaceMaxNumWordWrapLines(); For any text object, define the maximum number of lines that can be wrapped when setPlaceWordWrap() is enabled. The default value is 2.
- ❑ setPlaceMinFontSize(); For any text object, define the minimum font size that can be used by the automatic graph layout manager to resize a text object. The default value is 4 point.
- ❑ setPlacePosition(); Define how the automatic graph layout manager can position objects. In the current implementation, only the legend object can be repositioned. The default value is zero.
- ❑ setPlaceRect(); Set the rectangle to be used for initial placement of objects. In the current implementation, only object IDs return by getPlaceVertO1LabelsTitle() or getPlaceHorzO1LabelsTitle() can be used.
- ❑ setPlaceResize(); For any graph object, allow or disallow the automatic graph layout manager to move or resize the object. By default, resizing of all graph objects is allowed.

- ❑ `setPlaceSkip()`; For primary ordinal axis (O1) labels, allow or disallow the automatic graph layout manager to skip labels. By default, label skipping is allowed.
- ❑ `setPlaceTruncateCount()`; For any text object, set the number of characters that can be truncated.
- ❑ `setPlaceWordWrap()`; For any text object, allow or disallow word wrapping. By default, word wrapping is allowed.

Note:

- ❑ This property is automatically set to false after the graph layout manager has been activated. If you want to use the graph layout manager when a container is resized, your container resize routine should include `setPlace(true)`.
- ❑ In most cases, `setPlace(true)` is all that is required for the graph layout manager to produce an acceptable graph layout. The `setPlace...()` methods can be used if `setPlace(true)` alone does not produce acceptable results.
- ❑ In the current implementation, this property and the corresponding `setPlace...()` methods only work with O1-Axis labels and title and the legend area in horizontal and vertical bar, line, and area graphs. See the individual `setPlace()` methods for more detailed information.

See All `get/setPlace...` methods in [Methods](#) on page 715 and `getPlaceVertO1LabelsTitle()` and `getPlaceHorzO1LabelsTitle()` objects in [Methods That Return an Object ID](#) on page 1033.

PrimitiveAntialiasing

This property enables or disables smoother drawing of graph objects using Java 2D. Your development environment must include JRE 1.2 (or later) in order to use this property.

Syntax:

```
setPrimitiveAntialiasing(boolean);
boolean = getPrimitiveAntialiasing();
```

where:

boolean

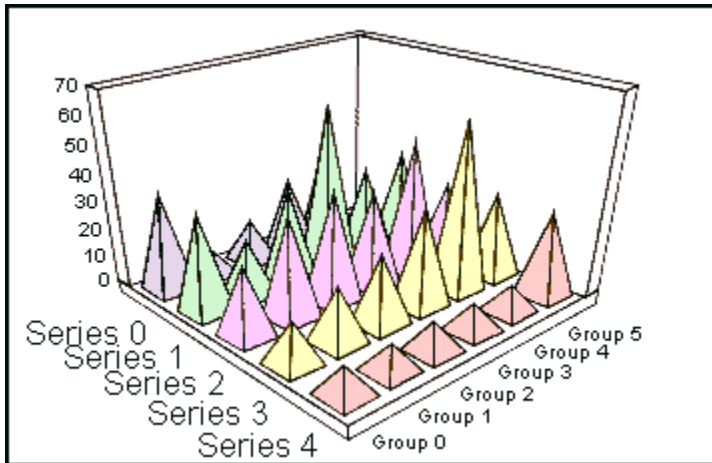
Can be one of the following:

[true](#) enables smoother drawing of graph objects.

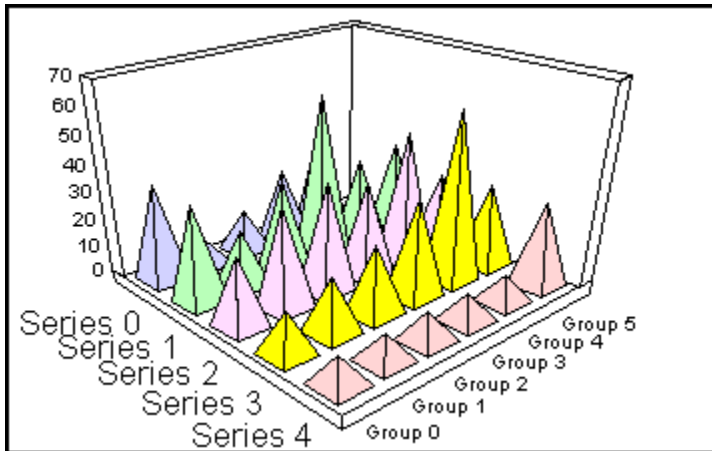
[false](#) disables smoother drawing of graph objects. This value is the default.

Example:

```
setGraphType(1>true;  
setAutoshading(false); setPrimitiveAntialiasing(true);
```



```
setGraphType(1>true;  
setAutoshading(false); setPrimitiveAntialiasing(false);
```



See [TextAntialiasing](#) on page 551 for information about improving the quality of text objects.

QuadrantLineCountX

This property sets the number of vertical quadrant lines on the X-axis in a bubble graph.

Syntax:

```
setQuadrantLineCountX(value);  
value = getQuadrantLineCountX();
```

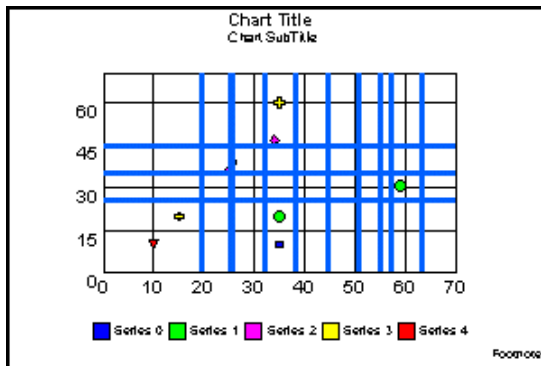
where:

value

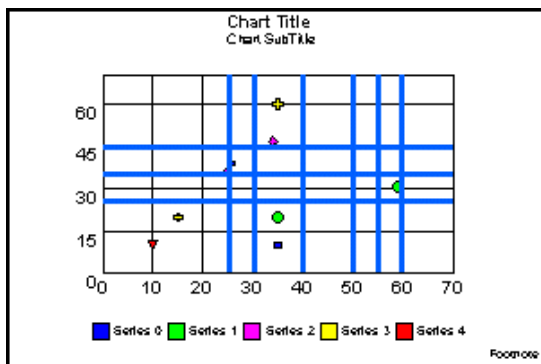
Is any integer. 1 is the default value.

Example:

```
setGraphType(89); setQuadrantLineCountX(10);
```



```
setQuadrantLineCountX(6);
```



Note: This property is only applicable to bubble graphs. It is ignored if any other graph type is selected.

QuadrantLineCountY

This property sets the number of horizontal quadrant lines in a bubble graph.

Syntax:

```
setQuadrantLineCountY(value);
```

```
value = getQuadrantLineCountY();
```

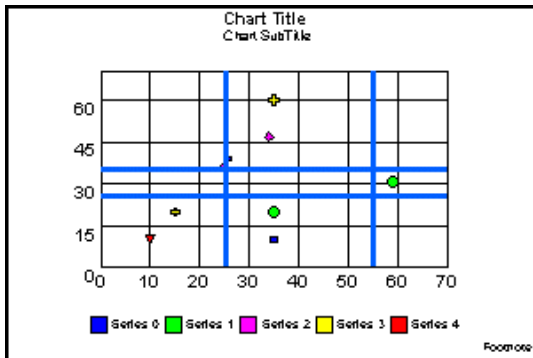
where:

value

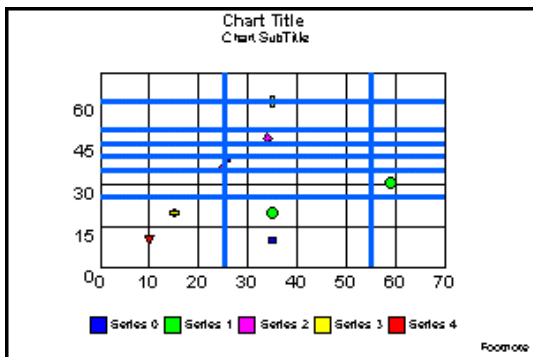
Is any integer. 1 is the default value.

Example:

```
setGraphType(89); setQuadrantLineCountY(2);
```



```
setQuadrantLineCountY(6);
```



Note: This property is only applicable to bubble graphs. It is ignored if any other graph type is selected.

See [QuadrantLineCountX](#) on page 499.

ReferencePercentageMode

This property determines whether the XY coordinates used for a reference will be the absolute value, or the percentage value, of the frame.

Syntax:

```
setReferencePercentageMode (IdentObj idobj, boolean);
```

where:

idObj

Is the object ID of a reference object (for example, `getReferenceLineXY()`, `getReferenceMarker()`, and so on).

boolean

Can be one of the following:

`true` is the percentage value of the frame. For the `getReferenceMarker()`, `getReferenceRect()`, and `getReferenceLineXY` methods, the percentage value is 0 to 1. For the `getReferenceLineXYRatio()` method, the percentage value is 0 to 100.

`false` is the absolute value of the X and Y-axis. This is the default value.

ReferenceTextPosition

This property defines the position where text is displayed relevant to the particular reference.

Syntax:

```
setReferenceTextPosition(value);
```

where:

value

Can be one of the following:

`0` for center above. This is the default value.

`1` for left above.

`2` for right above.

`3` for center below.

`4` for left below.

`5` for right below.

`6` for center.

RemoveDuplicateDataText

This property controls whether data text values that are the same for two or more series are represented by a single data text value or are repeated.

Syntax:

```
setRemoveDuplicateDataText(boolean);
```

```
boolean = getRemoveDuplicateDataText();
```

where:

boolean

Can be one of the following:

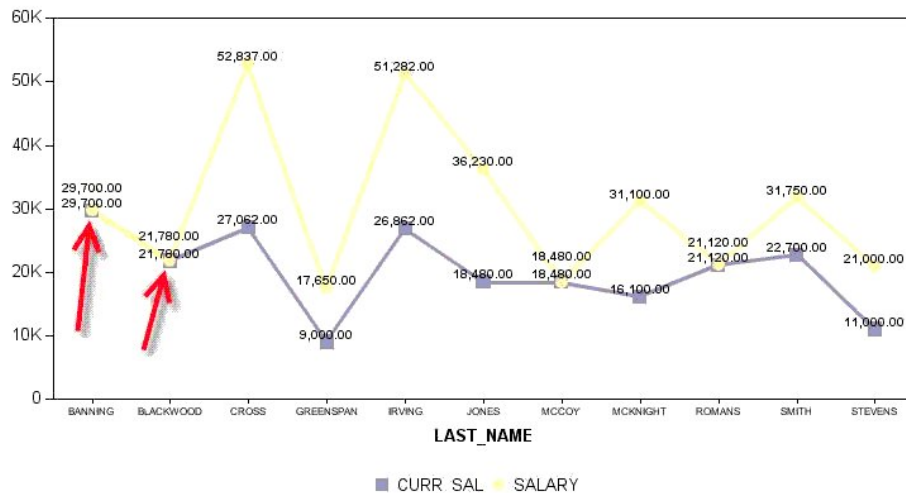
true removes duplicate text values.

false does not remove duplicate text values. This value is the default.

Example:

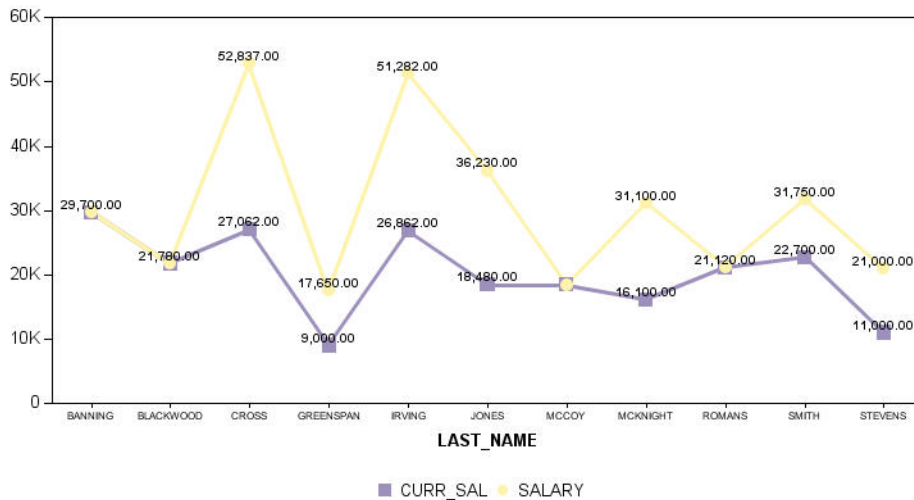
With `setRemoveDuplicateDataText(false)`, the points where the two series have the same data value show that value twice:

```
setRemoveDuplicateDataText(false);
```



With `setRemoveDuplicateDataText(true)`, the points where the two series have the same data value show that value once:

`setRemoveDuplicateDataText(true);`



RepositionDataTextCollision

This property repositions the data text value if multiple data text values are colliding.

Syntax:

`setRepositionDataTextCollision(value);`

where:

value

Is an integer from 1 to 10 that specifies the number of times to recursively go through and attempt to move data text around until the best solution is found. The default value is 4. The value 0 turns this property off.

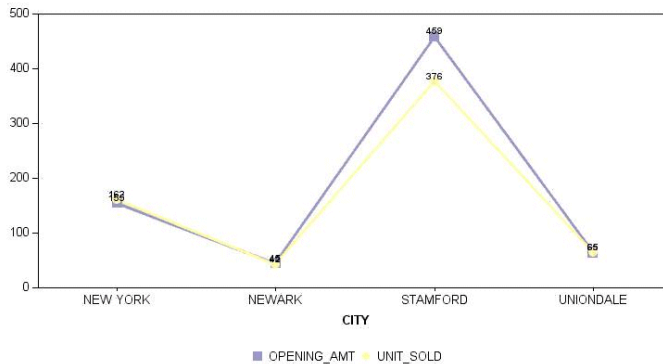
Note:

- ❑ This property is part of `setPlace(true)`. To override the default, `setPlace(false)` is required.
- ❑ This API call does not work with stacked charts. For stacked charts, you can use `setRepositionStackedDataTextCollision`.

Example:

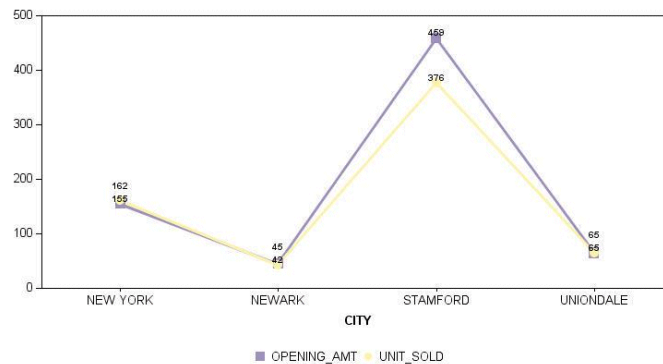
With `setRepositionDataTextCollision(0)`, the data text overlaps for points where the two series have the same value:

`setRepositionDataTextCollision(0);`



With `setRepositionDataTextCollision(4)`, the data text has been moved so it does not overlap:

`setRepositionDataTextCollision(4);`



See [RepositionStackedDataTextCollision](#) on page 505 and [Place](#) on page 496.

RepositionStackedDataTextCollision

This property controls whether to stagger data text values that are colliding in stacked bars.

Syntax:

`setRepositionStackedDataTextCollision(boolean);`

```
boolean = getRepositionStackedDataTextCollision();
```

where:

boolean

Can be one of the following:

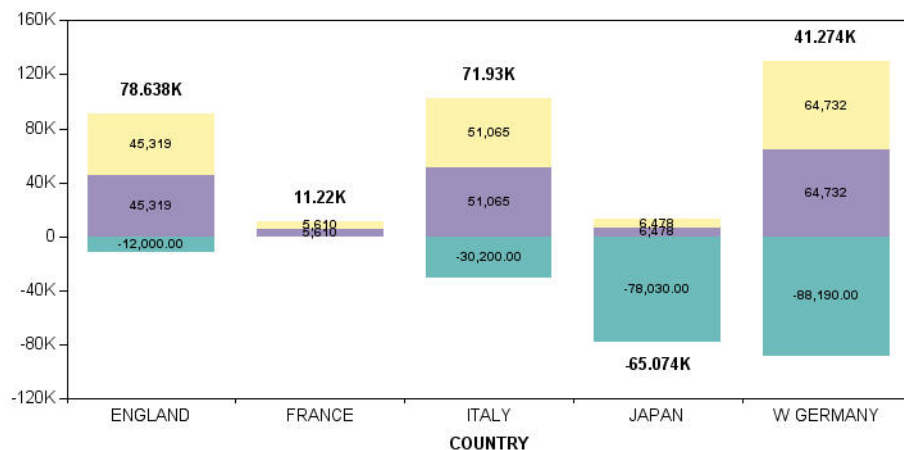
true staggers colliding data values.

false does not stagger colliding data values. This value is the default.

Example:

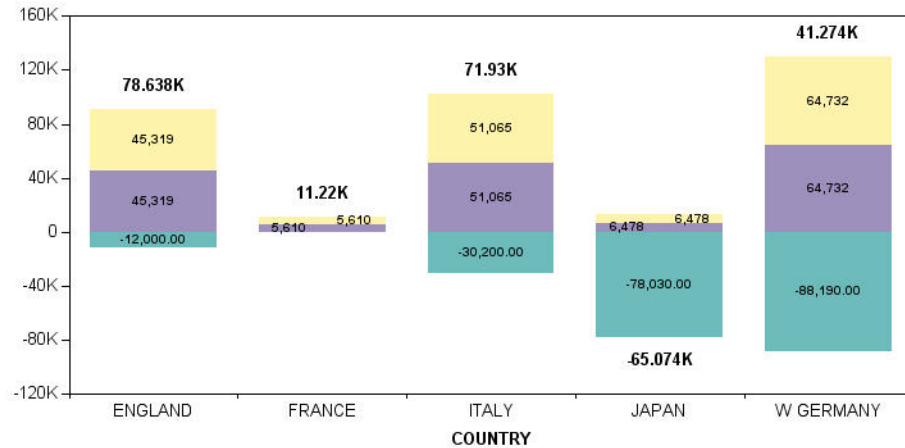
With `setRepositionStackedDataTextCollision(false)`, colliding data values are not staggered:

```
setRepositionStackedDataTextCollision(false);
```



With `setRepositionStackedDataTextCollision(true)`, colliding data values are staggered:

```
setRepositionStackedDataTextCollision(true);
```



See [RepositionDataTextCollision](#) on page 504.

ReshapeEnable

This property enables or disables the ability of the user to move and resize title, subtitle, footnote, graph frame, and legend box objects in a graph.

Syntax:

```
setReshapeEnable(boolean);
```

```
boolean = getReshapeEnable();
```

where:

boolean

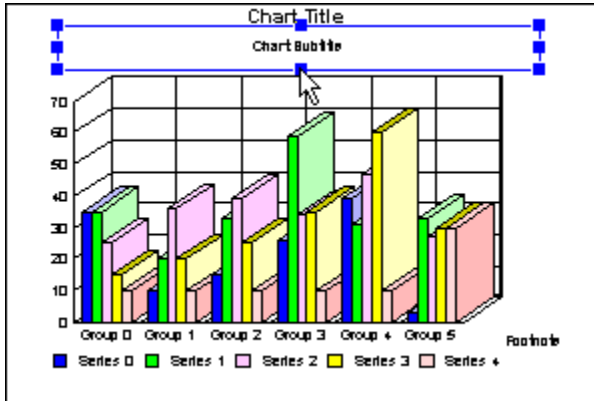
Can be one of the following:

[true](#) allows user to move or resize objects. This value is the default.

[false](#) does not allow user to move or resize objects.

Example:

```
setReshapeEnable(true);
```



See [ResizeBarMode](#) on page 508, [SelectionEnable](#) on page 522, [SelectionEnableMove](#) on page 523.

ResizeBarMode

This property enables or disables resize bar mode. When it is enabled, the user interface will allow resizing of bars. This property is only applicable to 2D and 2.5D bar graphs, it is not supported for 3D risers.

Syntax:

```
setResizeBarMode(boolean);
```

```
boolean = getResizeBarMode();
```

where:

boolean

Can be one of the following:

[true](#) allows user to resize bars. The user can change the size of bars by using the control key and the mouse to stretch or reduce their size. This action will also change the internal data value assigned to the bar. The `getDataValue()` method can be used to get the new data value.

If data callbacks are used to provide data input to the graph, any resize operation is reset to the value provided by the data call back. This value is the default.

[false](#) does not allow resizing of bars.

Note: The `DepthRadius` property must be set to zero in order for this property to enable the user interface to resize bars.

See [ReshapeEnable](#) on page 507, [SelectionEnable](#) on page 522, [SelectionEnableMove](#) on page 523, [DepthRadius](#) on page 334.

ReverseGroups

This property enables or disables drawing of groups in reverse order.

Syntax:

```
setReverseGroups(boolean);
```

```
boolean = getReverseGroups();
```

where:

boolean

Can be one of the following:

[true](#) draws groups in reverse order.

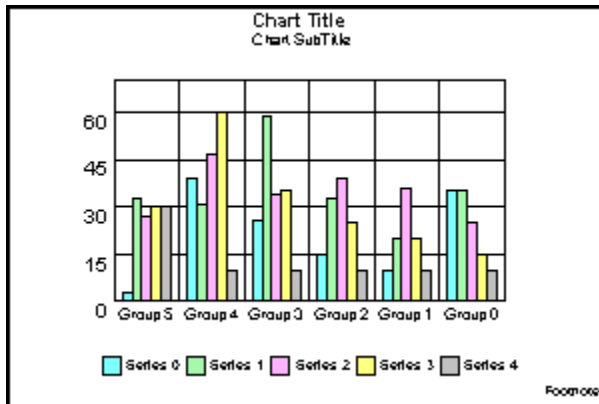
[false](#) draws groups in normal order. This value is the default.

Example:

```

setDepthAngle(0);
  setReverseGroups(true);
setDepthRadius(0);

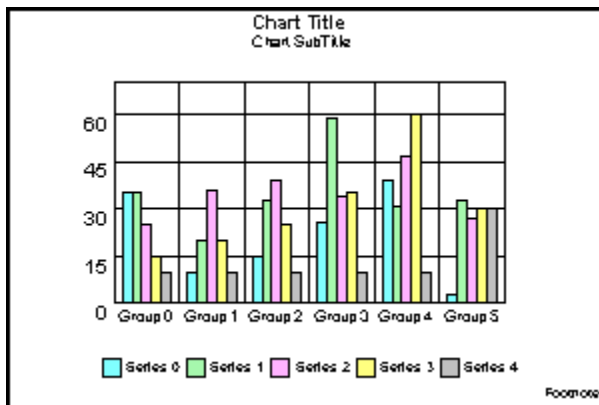
```



```

setReverseGroups(false);

```



See [ReverseSeries](#) on page 510.

ReverseSeries

This property enables or disables drawing of series in reverse order.

Syntax:

```

setReverseSeries(boolean);
boolean = getReverseSeries();

```

where:

boolean

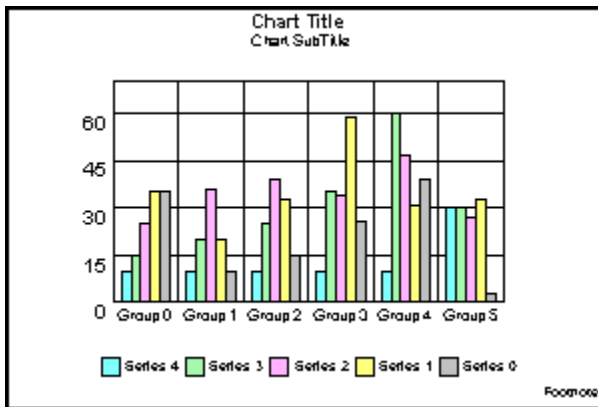
Can be one of the following:

true draws series in reverse order.

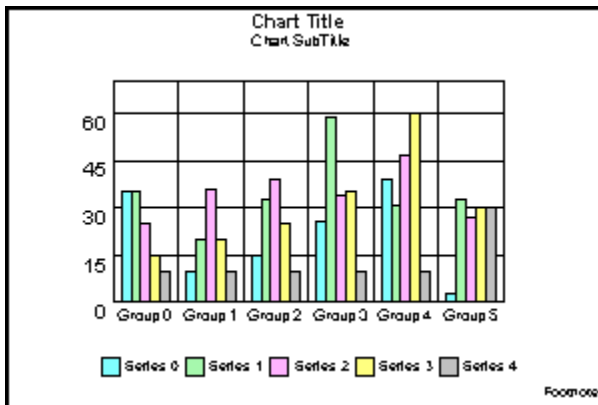
false draws series in normal order. This value is the default.

Example:

```
setReverseSeries(true);
setDepthAngle(0);
setDepthRadius(0);
```



```
setReverseSeries(false);
```



See [ReverseGroups](#) on page 509.

Riser3DThicknessY

This property determines the thickness of risers in 3D ribbon, floating cube, and floating pyramid graphs. It is used in combination with CubeSquareRisers and RiserWidth to set the X-, Z-, and Y-thickness of the risers (ribbons, cube, or pyramids).

Syntax:

```
setRiser3DThicknessY(value);
```

```
value = getRiser3DThicknessY();
```

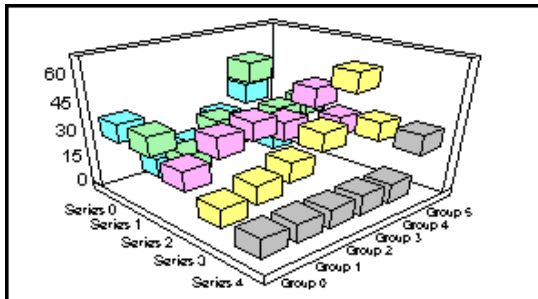
where:

value

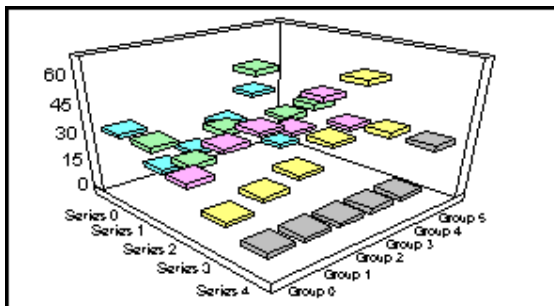
Is any integer between 1 and 100. 75 is the default value.

Example:

```
setGraphType(4);  
setRiser3DThicknessY(100);  
setAutoshading(false);  
setO2LabelAutofit(true);
```



```
setRiser3DThicknessY(25);
```



Note: A 3D Ribbon, Floating Cube, or Floating Pyramid graph type must be selected.

See [RiserWidth](#) on page 516, [CubeSquareRisers](#) on page 284.

RiserBarGroupSpacing

For bar graphs only, this property sets the spacing between bars within a group (clusters) in a side-by-side bar graph. A larger number creates more space between bars within a group. When this property is set to zero, there is no space between the bars and all bars in the group will touch each other. When this property is set to 100, bars in a group are as far apart as possible.

Syntax:

```
setRiserBarGroupSpacing(value);  
value = getRiserBarGroupSpacing();
```

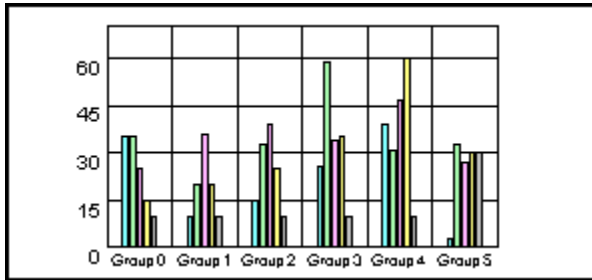
where:

value

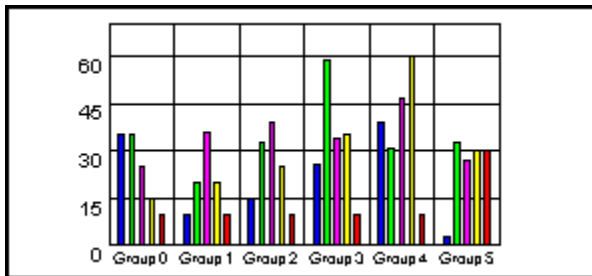
Is any integer between -1 and 100. 0 is the default value.

Example:

```
setRiserWidth(40);
setDepthRadius(0);
setRiserBarGroupSpacing(20);
```



```
setRiserBarGroupSpacing(50);
```

**Note:**

- ❑ This property is interactive with RiserWidth. If bars are set to maximum width, space cannot be allocated between bars within a group (that is, group spacing works by rearranging the space available between the Risers after the size set by RiserWidth). RiserWidth affects the amount of space available to RiserBarGroupSpacing.
- ❑ This property affects all riser objects, it cannot be applied to an individual series or group.

See [RiserWidth](#) on page 516.

RiserBorderMode

This property defines when riser borders are drawn. Automatic mode (0) only draws borders for small data sets that consist of less than 900 risers. You can also set the property so that riser borders are always drawn or never drawn.

Syntax:

```
setRiserBorderMode(value);
value = getRiserBorderMode();
```

where:

value

Can be one of the following:

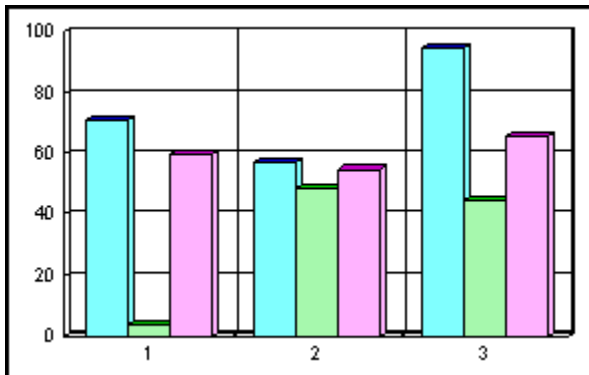
0 for automatic, borders drawn only for small data sets (fewer than 900 risers).

1 for borders never drawn.

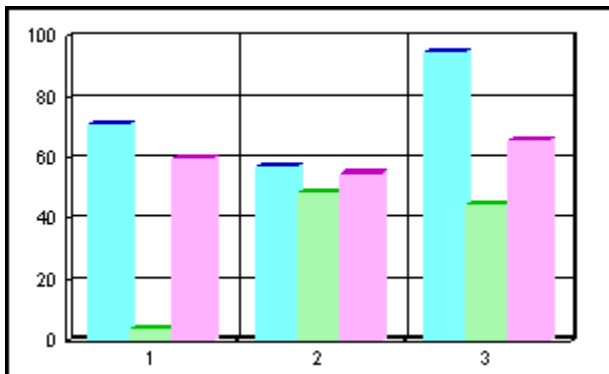
2 for borders are always drawn. 2 is the default value.

Example:

```
setRiserBorderMode(0);
```



```
setRiserBorderMode(1);
```



See [SeriesDefaultBorderColor](#) on page 524, [SeriesDefaultTransparentBorderColor](#) on page 526, [UseSeriesBorderDefaults](#) on page 562.

RiserWidth

For bar graphs, stock high/low graph, and histograms, this property controls the width of bars in a graph. The riser width is expressed as a percentage of the space available for each group of bars. When this property is set to 100, all bars touch each other and there is no space in between them. When this property is set to zero, the bars are displayed as thin lines with no space detectable.

Syntax:

```
setRiserWidth(value);  
value = getRiserWidth();
```

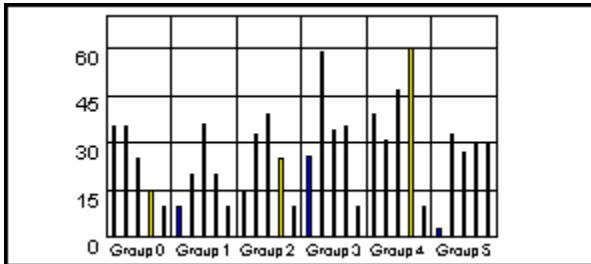
where:

value

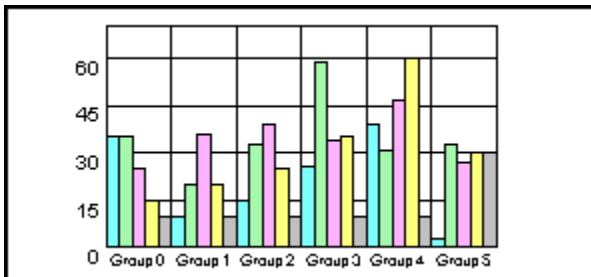
Is any integer between 0 and 100. 75 is the default value.

Example:

```
setRiserWidth(20);
setDepthRadius(0);
setRiserBarGroupSpacing(75);
```



```
setRiserWidth(100);
```

**Note:**

- ❑ This property can interact with `RiserBarGroupSpacing`, since Group Spacing can only adjust the space as a percentage of what is left over after group spacing. For example, if there is a lot of space between the groups, there will not be much visual difference between large and small values in riser bar width.
- ❑ This property affects all riser objects; it cannot be applied to an individual series or group.

See [RiserBarGroupSpacing](#) on page 513.

ScaleFromZero

In applications where the data range can include negative numbers, this property determines whether the risers are drawn pointing up and down from a zero line or whether all risers will draw straight up from the "bottom" of the graph. It affects risers on 3D graphs and all 2D graphs except Pies graphs.

Syntax:

```
setScaleFromZero(boolean);
boolean = getScaleFromZero();
```

where:

boolean

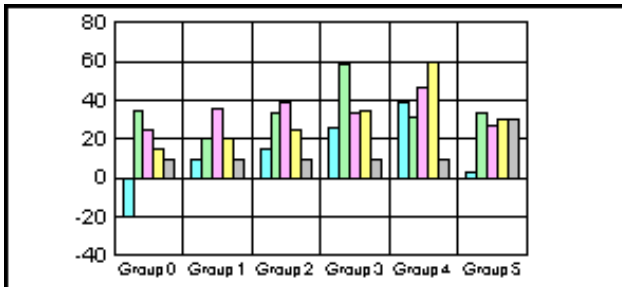
Can be one of the following:

true draws from the zero line. This value is the default.

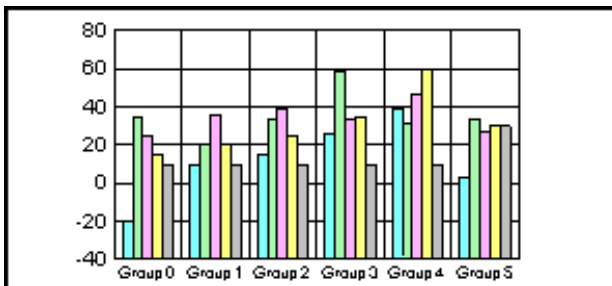
false draws from the base of the frame.

Example:

```
setY1MajorGridStep(20.0);
  setScaleFromZero(true);
setY1ScaleMax(80.0);
setY1ScaleMaxAuto(false);
setY1ScaleMin(-40.0);
setY1ScaleMinAuto(false);
setDepthRadius(0);
```



```
setScaleFromZero(false);
```



ScrollOffsetGroup

When the ViewableGroups property is set to a value greater than zero, this property selects the first group to appear in the graph.

Syntax:

```
setScrollOffsetGroup(value);
```

```
value = getScrollOffsetGroup();
```

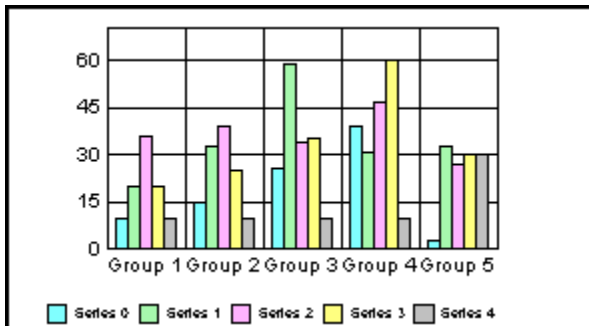
where:

value

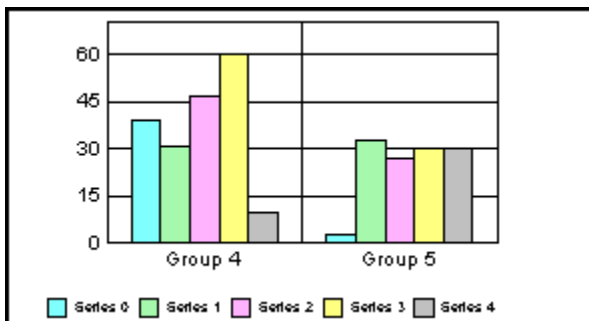
Is any integer between 0 and the number of viewable groups in the graph. 0 is the default value.

Example:

```
setViewableGroups(5);
  setScrollOffsetGroup(1);
setDataScrollerPresenceGroups(0);
setDepthRadius(0);
```



```
setViewableGroups(2); setScrollOffsetGroup(4);
```



Note:

- ❑ This property will only function if ViewableGroups is set to a value greater than zero.
- ❑ The DataScrollerPresenceGroups property can be used to display a data scrolling bar on the group axis. The data scrolling bar can be used to adjust the ScrollOffsetGroups via the user interface.

See [DataScrollerPresenceGroups](#) on page 310, [ViewableGroups](#) on page 566, [ScrollOffsetSeries](#) on page 520.

ScrollOffsetSeries

When ViewableSeries is set to a value greater than zero, this property selects the first series to appear in the graph.

Syntax:

```
setScrollOffsetSeries(value);  
value = getScrollOffsetSeries();
```

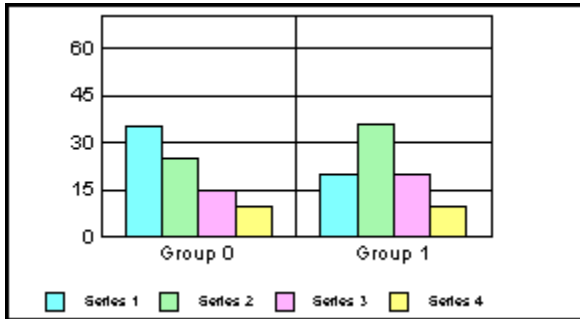
where:

value

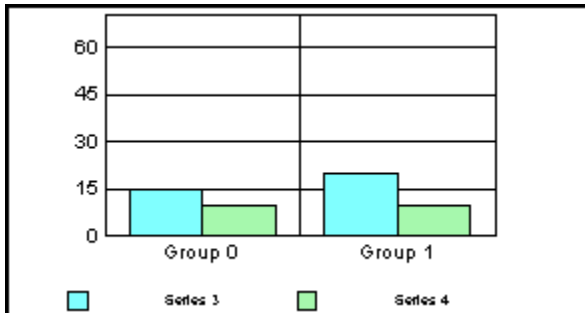
Is any integer between 0 and the number of viewable groups in the graph. 0 is the default value.

Example:

```
setViewableSeries(4);
setViewableGroups(2);
setDepthRadius(0);
setDataScrollerPresenceGroups(0); setScrollOffsetSeries(1);
```



```
setViewableSeries(2); setScrollOffsetSeries(2);
```



Since there are only five series in the data that defines this graph, `setViewableSeries(3)` and `setScrollOffsetSeries(3)` results in only two series remaining in the graph.

Note:

- ❑ This property will only function if `ViewableSeries` is set to a value greater than zero.
- ❑ The `DataScrollerPresenceSeries` property can be used to display a data scrolling bar for the series. The data scrolling bar can be used to adjust the `ScrollOffsetSeries` via the user interface.

See [DataScrollerPresenceSeries](#) on page 311, [ViewableSeries](#) on page 568, [ScrollOffsetGroup](#) on page 519.

SelectionEnable

This property defines the user's ability to select objects in a graph and the operations that can be performed when one or more objects are selected.

- ❑ When `setSelectionEnable(1)` is used, the user can select individual objects (risers, markers, labels, and so on) in a graph. The graphing engine highlights each object that is selected.
- ❑ When `setSelectionEnable(2)` is used, the user can only select data-related objects (bars, legends, and so on). The graphing engine will highlight the selected data objects and all related objects.
- ❑ When `setSelectionEnable(3)` is used, the user cannot select individual objects. When one object is selected, all related objects will also be selected and highlighted.
- ❑ When `setSelectionEnable(4)` is used, the user can click on a single point and have the graph zoom at the rate of 50% on both axes with the selected point used as the center of the zoom rectangle.
- ❑ When `setSelectionEnable(5)` is used, it allows the user to drag out a rectangle and zoom in on the data within the rectangle area. When this mode is used, a single mouse click will back up one data-zoom level. Double click will restore the original zoomed data state. Note that user interface-based zooming and scrolling is mutually exclusive with selecting graph items. When `setSelectionEnable(5)` is used, graph objects cannot be selected.

Syntax:

```
setSelectionEnable(value);
```

```
value = getSelectionEnable();
```

where:

value

Can be one of the following:

0 for no selection.

1 selects a single element, only one object is highlighted.

2 selects only data related objects (bar, legend) and select all related objects when clicking on one.

3 selects all related objects when one is clicked. 3 is the default value.

4 selects 50% data zooming on user-selected object.

5 selects drill-down/data zooming mode.

See [SelectionEnableMove](#) on page 523.

SelectionEnableMove

This property enables or disables a graph user's ability to select and move individual moveable objects (title, subtitle, footnote, graph frame, and legend box) in a graph.

Syntax:

```
setSelectionEnableMove(boolean);  
boolean = getSelectionEnableMove();
```

where:

boolean

Can be one of the following:

[true](#) allows user to select and move objects. This value is the default.

[false](#) allows user to select but not move objects.

See [SelectionEnable](#) on page 522.

SeriesAreRows

This property stores series values as rows or columns.

Syntax:

```
setSeriesAreRows(boolean);  
boolean = setSeriesAreRows();
```

where:

boolean

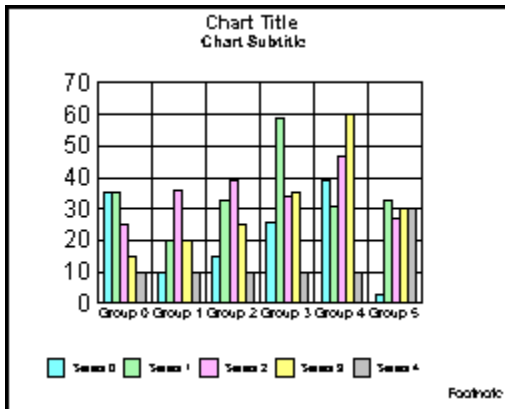
Can be one of the following:

[true](#) stores series as rows. This value is the default.

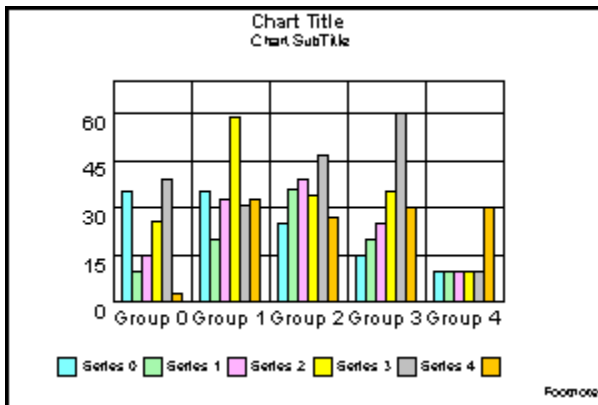
[false](#) stores series as columns.

Example:

```
setDepthAngle(0);
  setSeriesAreRows(true);
setDepthRadius(0);
```



```
setSeriesAreRows(false);
```



See [ReverseSeries](#) on page 510, [ReverseGroups](#) on page 509.

SeriesDefaultBorderColor

This property sets the default color to use for riser edges when the `UseSeriesBorderDefaults` property is enabled. This property is only used when `UseSeriesBorderDefaults` is enabled. When `UseSeriesBorderDefaults` is disabled, this property is ignored.

Syntax:

```
setSeriesDefaultBorderColor(new Color(newValue));
```



```
newValue = setSeriesDefaultBorderColor();
```

where:

newValue

Is the default color value in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

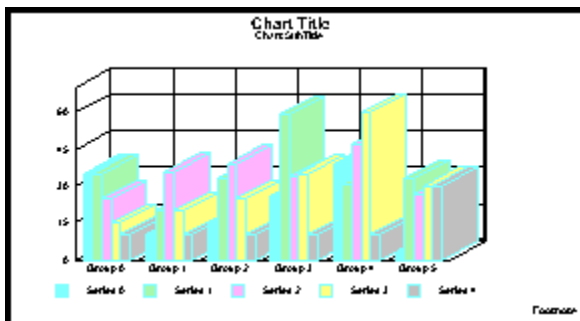
#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

Example:

```
setUseSeriesBorderDefaults(true); setSeriesDefaultBorderColor(new  
Color(128,255,255));
```

or, using a hexadecimal color code:

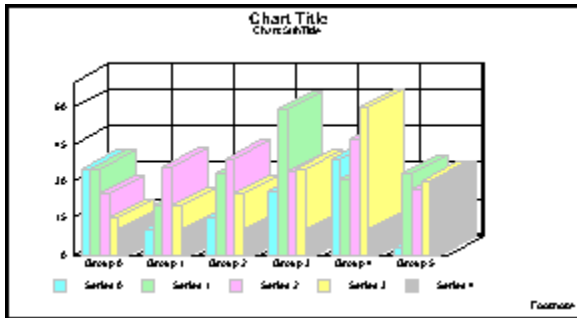
```
setUseSeriesBorderDefaults(true); setSeriesDefaultBorderColor(new  
Color(#80FFFF));
```



```
setUseSeriesBorderDefaults(true); setSeriesDefaultBorderColor(new  
Color(192,192,192));
```

or, using a hexadecimal color code:

```
setUseSeriesBorderDefaults(true); setSeriesDefaultBorderColor(new  
Color(#C0C0C0));
```



See [SeriesDefaultTransparentColor](#) on page 526, [UseSeriesBorderDefaults](#) on page 562.

SeriesDefaultTransparentColor

When UseSeriesBorderDefaults is enabled, this property enables or disables the use of a transparent border color around the edges of risers. This property is only used when UseSeriesBorderDefaults is enabled and is ignored when UseSeriesBorderDefaults is disabled.

Syntax:

```
setSeriesDefaultTransparentColor(boolean);  
boolean = setSeriesDefaultTransparentColor();
```

where:

boolean

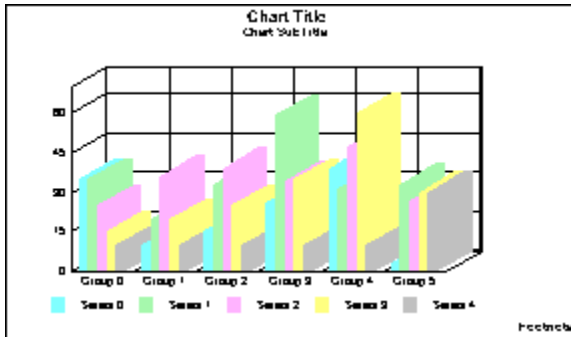
Can be one of the following:

[true](#) uses transparent edges around risers. This value is the default.

[false](#) uses the color assigned to SeriesDefaultBorderColor.

Example:

```
setUseSeriesBorderDefaults(true);
setSeriesDefaultTransparentColor(true);
```



See [SeriesDefaultBorderColor](#) on page 524, [UseSeriesBorderDefaults](#) on page 562.

SeriesLineWidthDefault

This property sets the default line width for all series in a line graph.

Syntax:

```
setSeriesLineWidthDefault(value);
value = getSeriesLineWidthDefault();
```

where:

value

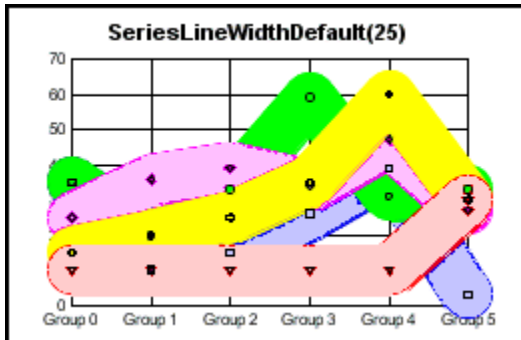
Is any integer between 1 and 100. 3 is the default value.

Example:

```

setGraphType(41);
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleString("SeriesLineWidthDefault(25)");
setSeriesLineWidthDefault(25);

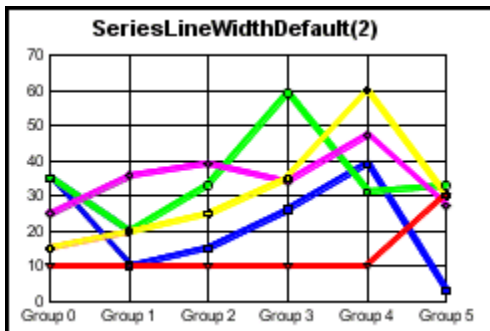
```



```

setGraphType(41);
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleString("SeriesLineWidthDefault(2)"); setSeriesLineWidthDefault(2);

```



Note: DepthRadius must be zero in order for this property to work.

SeriesLooping

This property is used to select the interval at which series color values are repeated.

Syntax:

```
setSeriesLooping(value);
value = getSeriesLooping();
```

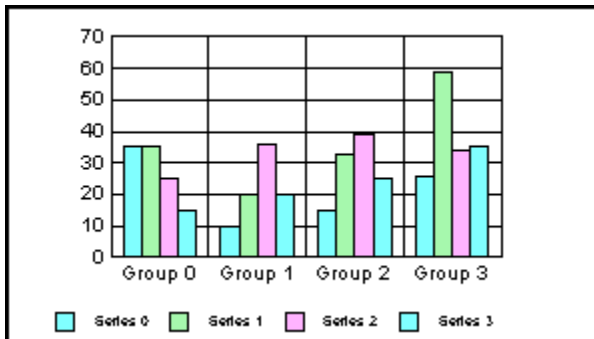
where:

value

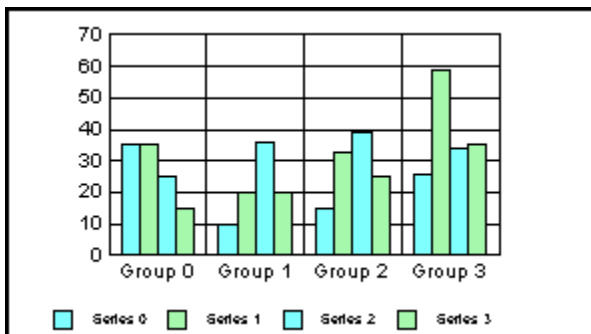
Is any integer between 1 and the number of series in data that defines the graph. A value of 1 defines the same color for all series. The default value is 1023 and the maximum is 2048. Starting with series 1024, the colors are repeated (that is, series 1024...2048 have the same colors and 0...1023, respectively).

Example:

```
setViewableGroups(4);
setViewableSeries(4);
setDepthRadius(0);
setDataScrollerPresenceGroups(0); setSeriesLooping(3);
```



`setSeriesLooping(2);`



ShadowXOffsetDefault

This property sets the default value for the X-Offset of a drop shadow.

Syntax:

```
setShadowXOffsetDefault(value);  
value = getShadowXOffsetDefault();
```

where:

value

Is any integer between -20 and 20. 2 is the default value.

See [getShadowXOffset\(\)](#) on page 814, [getShadowYOffset\(\)](#) on page 814, [setShadowColor\(\)](#) on page 984, [setShadowDisplay\(\)](#) on page 985, [setShadowXOffset\(\)](#) on page 989, [setShadowYOffset\(\)](#) on page 990.

ShadowYOffsetDefault

This property sets the default value for the Y-Offset of a drop shadow.

Syntax:

```
setShadowYOffsetDefault(value);  
value = getShadowYOffsetDefault();
```

where:

value

Is any integer between -20 and 20. 2 is the default value.

See [getShadowXOffset\(\)](#) on page 814, [getShadowYOffset\(\)](#) on page 814, [setShadowColor\(\)](#) on page 984, [setShadowDisplay\(\)](#) on page 985, [setShadowXOffset\(\)](#) on page 989, [setShadowYOffset\(\)](#) on page 990.

SmoothLines

This property enables or disables drawing of smooth lines in line charts.

Syntax:

```
setSmoothLines (boolean);  
boolean = SmoothLines();
```

where:

boolean

Can be one of the following:

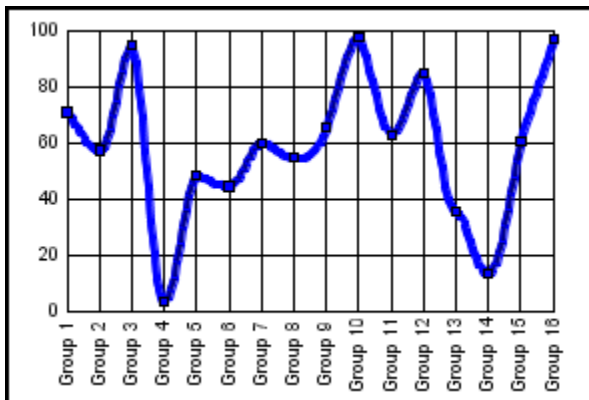
true draws smooth lines.

false draws lines normally. This value is the default.

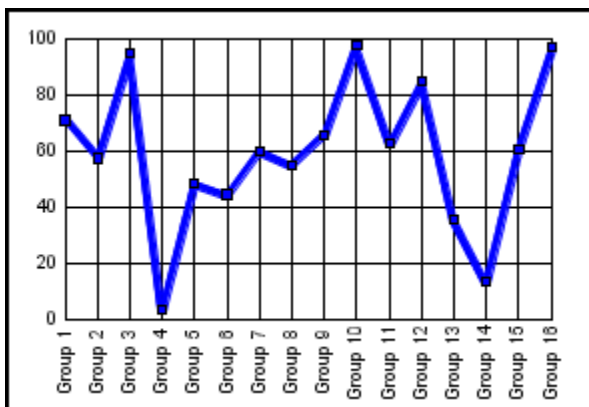
Note: The smooth line effect can only be seen on a chart with more than 3 groups.

Example:

`setSmoothLines(true);`



`setSmoothLines(false);`



SquareMarkers

This property enables or disables square markers in the legend area. If different marker shapes (circle, plus, triangle, and so on) are used for each series by UseSeriesShapes, the marker shape is redrawn within a square rectangle area. The legend marker position must not be set for the legend labels to appear on of the legend markers.

Syntax:

```
setSquareMarkers(boolean);
```

```
boolean = getSquareMarkers();
```

where:

boolean

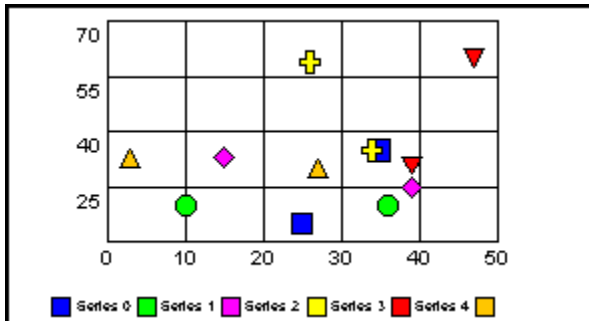
Can be one of the following:

[true](#) forces markers to be square. This value is the default.

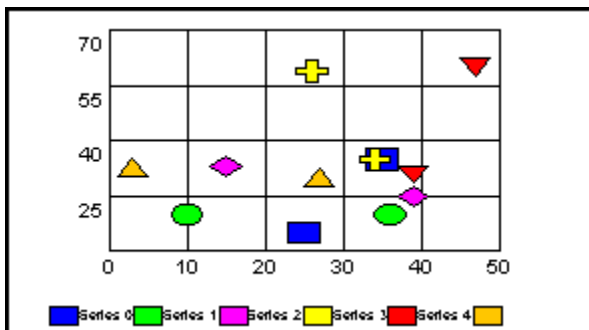
[false](#) does not force markers to be square.

Example:

```
setGraphType(61);
setSquareMarkers(true);
```



```
setSquareMarkers(false);
```



Note: This property is ignored if `LegendMarkerPosition(4)` is used.

See [LegendMarkerPosition](#) on page 400, [UseSeriesShapes](#) on page 562.

StackedDataValueSum

When a stacked graph type is selected and data text is displayed with `DataTextDisplay` set to true, the value assigned to this property determines how data values are displayed.

Syntax:

```
setStackedDataValueSum(boolean);
boolean = getStackedDataValueSum();
```

where:

boolean

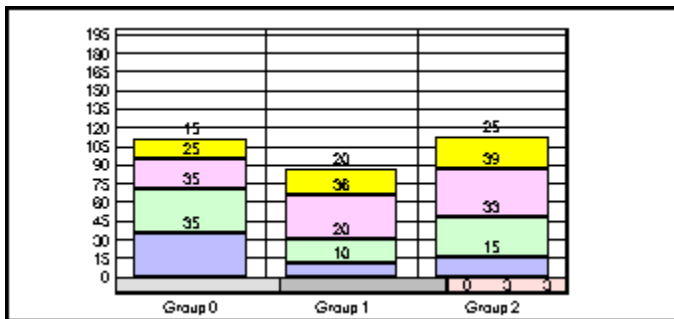
Can be one of the following:

[true](#) shows cumulative sums of riser segments. This value is the default.

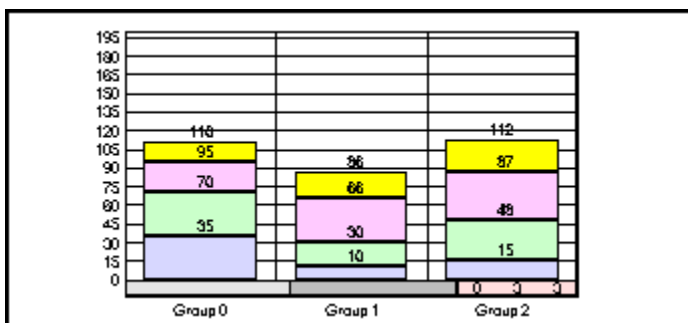
[false](#) shows values reflecting individual segments of the risers.

Example:

```
setDataTextDisplay(true);
setDepthAngle(0);
setViewableSeries(4);
  setStackedDataValueSum(false);
setGraphType(18);
setViewableGroups(3);
setDepthRadius(0);
```



[setStackedDataValueSum\(true\);](#)



See [DataTextDisplay](#) on page 316.

Stock52WeekHighDisplay

This property enables or disables the display of a 52-week high line in a stock graph. The GraphType property must be set to one of the stock graphs (73...83). However, note that a 52-week high line is not available in bi-polar stock graphs. The Stock52WeekHighValue property must define the 52-week high value.

Syntax:

```
setStock52WeekHighDisplay(boolean);  
boolean = getStock52WeekHighDisplay();
```

where:

boolean

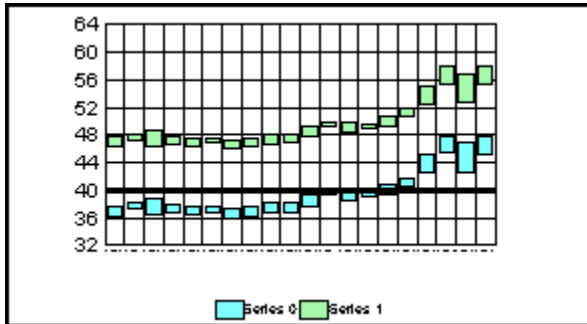
Can be one of the following:

`true` draws 52-week high line.

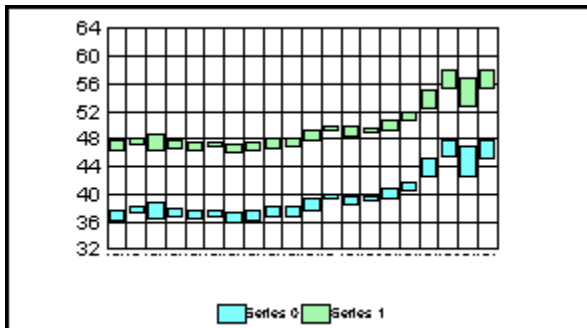
`false` does not draw 52-week high line. This value is the default.

Example:

```
setGraphType(73);  setStock52WeekHighDisplay(true);
```



```
setStock52WeekHighDisplay(false);
```



See [Stock52WeekHighValue](#) on page 536, [Stock52WeekLowValue](#) on page 539, [Stock52WeekLowDisplay](#) on page 538.

Stock52WeekHighValue

This property is used to set a 52-week high value in a stock graph. The GraphType property must be set to one of the stock graphs (73...83). However, note that a 52-week high line is not available in bi-polar stock graphs. The Stock52WeekHighDisplay property must be set to true in order to display a line in the graph identifying the 52-week high value. Only one line per graph is available.

Syntax:

```
setStock52WeekHighValue(value);  
value = getStock52WeekHighValue();
```

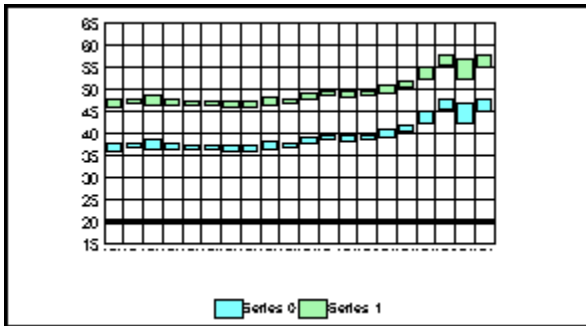
where:

value

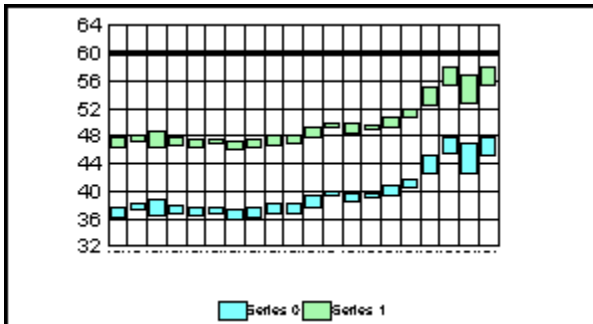
Is a value that falls within the data range of the graph. This value is a Java double-precision number. 40.0 is the default.

Example:

```
setGraphType(73);
  setStock52WeekHighValue(20.0);
setStock52WeekHighDisplay(true);
```



```
setStock52WeekHighValue(60.0);
```



See [Stock52WeekHighDisplay](#) on page 535.

Stock52WeekLowDisplay

This property enables or disables the display of a 52-week low line in a stock graph. The GraphType property must be set to one of the stock graphs (73...83). However, note that a 52-week low line is not available in bi-polar stock graphs. The Stock52WeekLowValue property must define the 52-week low value. Only one line per graph is available.

Syntax:

```
setStock52WeekLowDisplay(boolean);  
boolean = getStock52WeekLowDisplay();
```

where:

boolean

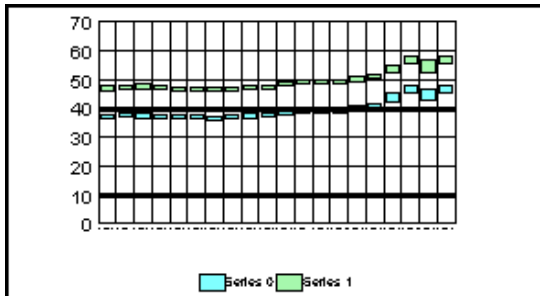
Can be one of the following:

`true` draws 52-week low line

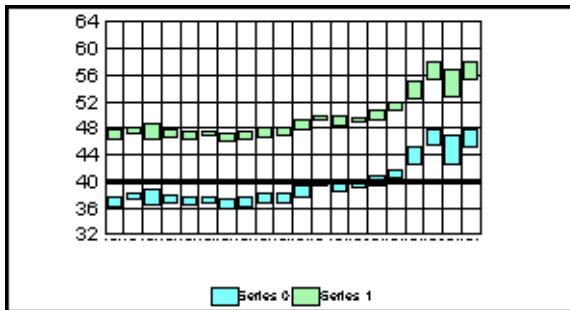
`false` does not draw 52-week low line. This value is the default.

Example:

```
setGraphType(73);
setStock52WeekHighDisplay(true); setStock52WeekLowDisplay(true);
```



```
setStock52WeekLowDisplay(false);
```



See [Stock52WeekLowValue](#) on page 539, [Stock52WeekHighValue](#) on page 536, [Stock52WeekHighDisplay](#) on page 535.

Stock52WeekLowValue

This property is used to set a 52-week low value in a stock graph. The GraphType property must be set to one of the stock graphs (73...83). However note that a 52-week low line is not available for bi-polar stock graphs. The Stock52WeekLowDisplay property must be set to true in order to display a line in the graph identifying the 52-week low value. Only one line per graph is available.

Syntax:

```
setStock52WeekLowValue(value);
value = getStock52WeekLowValue();
```

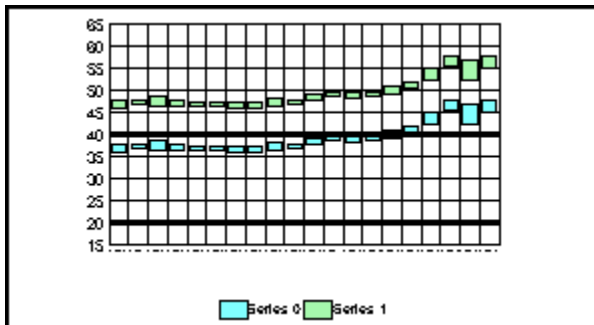
where:

value

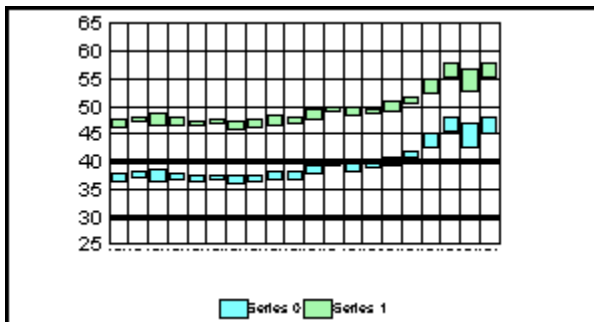
Is a value within the data range of the graph. This value is a Java double-precision number. 10.0 is the default.

Example:

```
setStock52WeekLowValue(20.0);
setGraphType(73);
setStock52WeekHighDisplay(true);
setStock52WeekLowDisplay(true);
```



```
setStock52WeekLowValue(30.0);
```



See [Stock52WeekLowDisplay](#) on page 538.

StockCandleForVolumeColor

This property controls the color of Volume Riser for Stock charts. When enabled, the Volume Riser color is the same as the Candle Color. When disabled, the user has the capability of specifying custom color for Volume Riser color.

Syntax:

```
setStockCandleForVolumeColor(boolean);
```

where:

boolean

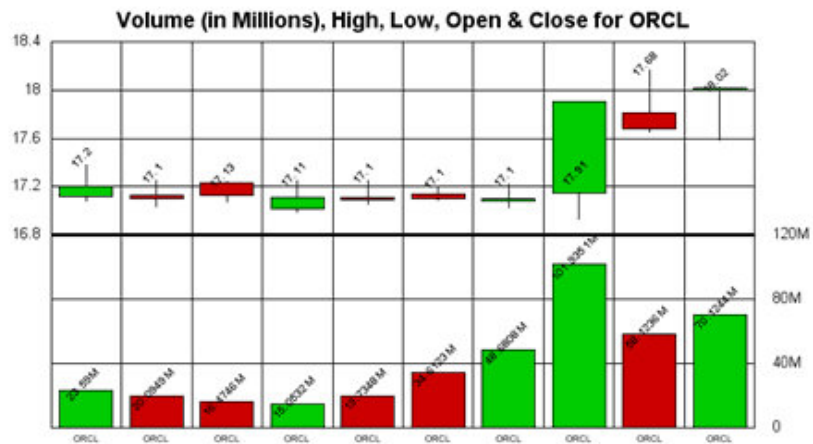
Can be one of the following:

[true](#) sets the Volume Riser color the same as the Candle color. This value is the default.

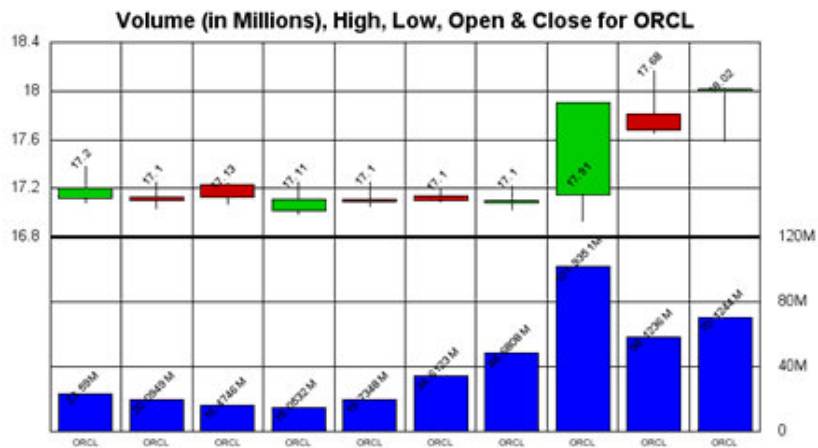
[false](#) allows the Volume Riser color to be customizable.

Example:

```
setStockCandleForVolumeColor(true);
```



```
setStockCandleForVolumeColor(false);  
setFillColor(getStockVolumeRiser(), new Color(0,0,255));
```



StockCloseSplitDisplay

This property enables or disables the display of split risers at the stock close value. The `GraphType` property must be set to one of the stock graphs (73...83). If a close value is not available, this property is ignored.

Syntax:

```
setStockCloseSplitDisplay(boolean);  
boolean = getStockCloseSplitDisplay();
```

where:

boolean

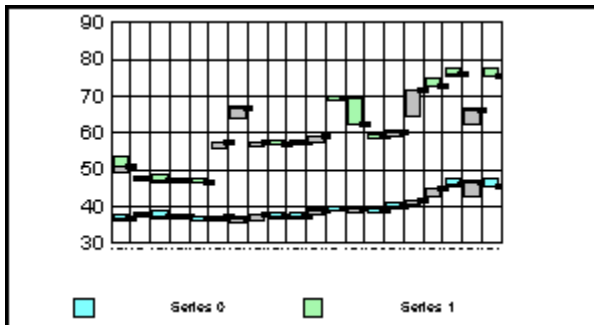
Can be one of the following:

`true` splits risers at close value. This value is the default.

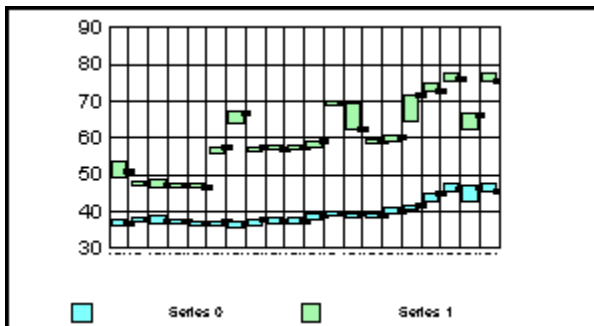
`false` does not split risers at close value.

Example:

```
setGraphType(76); setStockCloseSplitDisplay(true);
```



```
setStockCloseSplitDisplay(false);
```



See [StockCloseTicksDisplay](#) on page 544, [StockOpenTicksDisplay](#) on page 546.

StockCloseTicksDisplay

This property enables or disables the tick marks at the stock close value. The GraphType property must be set to one of the stock graphs (73...83). If a close value is not available, this property is ignored. The StockTickLength property determines the size of the tick marks.

Syntax:

```
setStockCloseTicksDisplay(boolean);  
boolean = getStockCloseTicksDisplay();
```

where:

boolean

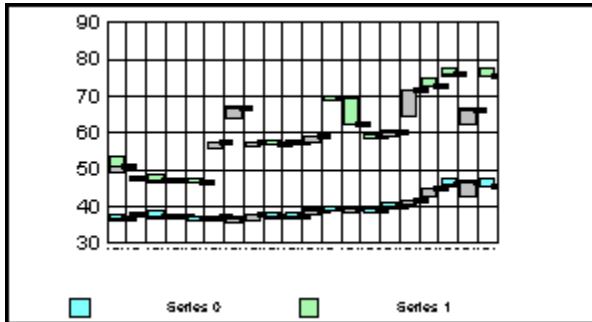
Can be one of the following:

[true](#) displays close value tick marks. This value is the default.

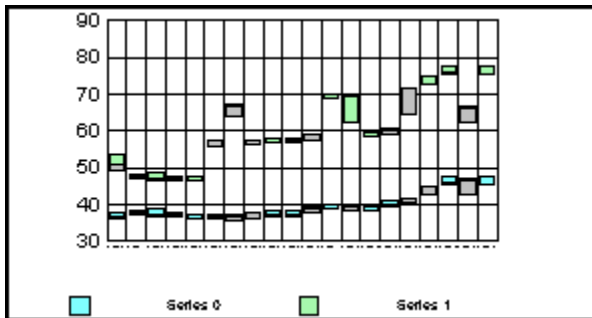
[false](#) does not display close value tick marks.

Example:

```
setGraphType(76);  setStockCloseTicksDisplay(true);
```



```
setStockCloseTicksDisplay(false);
```



See [StockCloseSplitDisplay](#) on page 543, [StockOpenTicksDisplay](#) on page 546, [StockTickLength](#) on page 547.

StockMovingAverageDisplay

This property enables or disables the display of a moving average line in a stock graph. The `GraphType` property must be set to one of the stock graphs (73...83).

Syntax:

```
setStockMovingAverageDisplay(boolean);
```

```
boolean = getStockMovingAverageDisplay();
```

where:

boolean

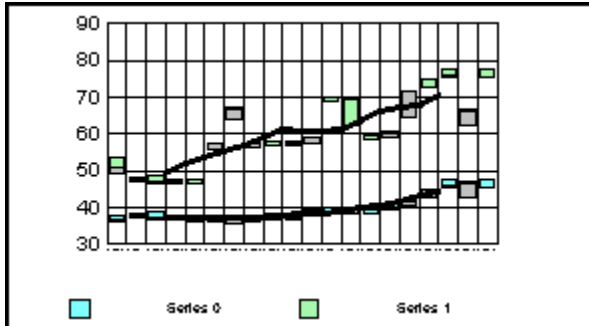
Can be one of the following:

`true` draws moving average line

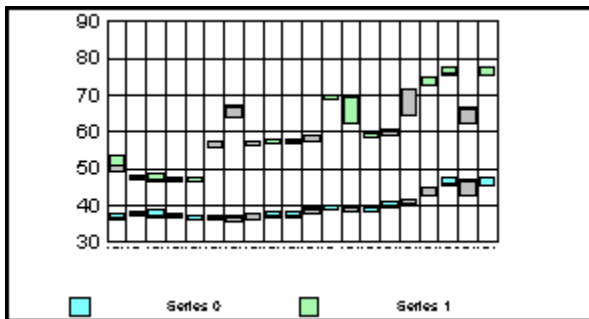
`false` does not draw moving average line. This value is the default.

Example:

```
setGraphType(76); setStockMovingAverageDisplay(true);
```



```
setStockMovingAverageDisplay(false);
```



See [Stock52WeekHighDisplay](#) on page 535, [Stock52WeekLowDisplay](#) on page 538, [StockCloseSplitDisplay](#) on page 543, [StockCloseTicksDisplay](#) on page 544, [StockOpenTicksDisplay](#) on page 546, [setLineWidth\(\)](#) on page 937.

StockOpenTicksDisplay

This property enables or disables the tick marks at the stock open values. The GraphType property must be set to one of the stock graphs (73...83). If an open value is not available, this property is ignored. The StockTickLength property determines the size of the tick marks.

Syntax:

```
setStockOpenTicksDisplay(boolean);  
boolean = getStockOpenTicksDisplay();
```

where:

boolean

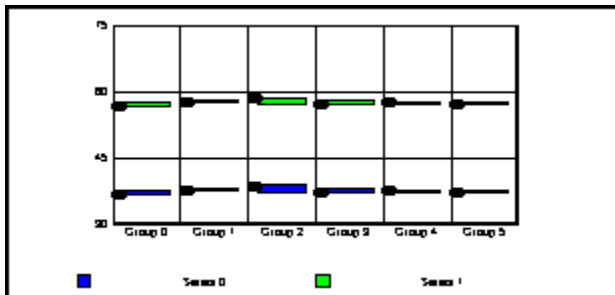
Can be one of the following:

[true](#) displays open value tick marks. This value is the default.

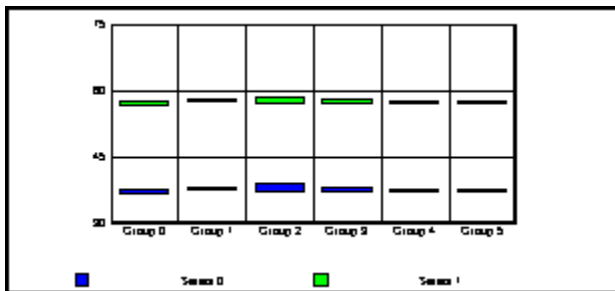
[false](#) does not draw open value tick marks.

Example:

```
setViewableGroupsStock(6);
setDataScrollerPresenceGroups(0);
setGraphType(79);
setStockOpenTicksDisplay(true);
```



```
setStockOpenTicksDisplay(false);
```



See [StockCloseTicksDisplay](#) on page 544, [StockTickLength](#) on page 547.

StockTickLength

This property determines the length of tick marks in a stock graph. The GraphType property must be set to one of the stock graphs (73...83). The StockCloseTicksDisplay and/or StockOpenTicksDisplay property must be enabled. If close and/or open values are not available, the tick mark(s) will not display.

Syntax:

```
setStockTickLength(value);  
value = getStockTickLength();
```

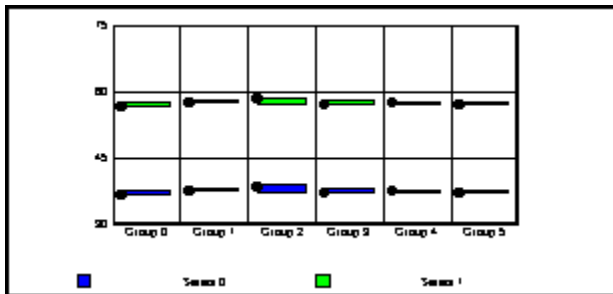
where:

value

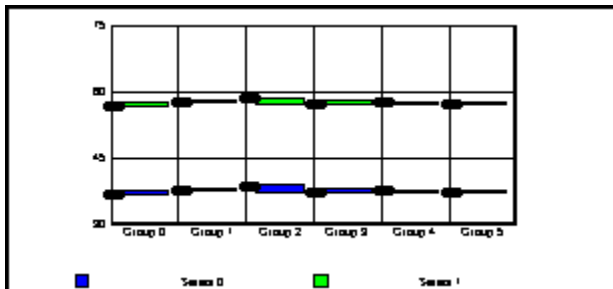
Is any integer between 0 and 100. 50 is the default value.

Example:

```
setViewableSeries(12);  
setViewableGroupsStock(6);  
setDataScrollerPresenceGroups(0);  
setGraphType(79);  
setStockCloseTicksDisplay(false);  
    setStockTickLength(10);  
setViewableSeriesStock(8);
```



```
setStockTickLength(100);
```



See [StockCloseTicksDisplay](#) on page 544, [StockOpenTicksDisplay](#) on page 546.

SubtitleDisplay

This property enables or disables display of the graph subtitle defined by the SubtitleString property.

Syntax:

```
setSubtitleDisplay(boolean);
```

```
boolean = setSubtitleDisplay();
```

where:

boolean

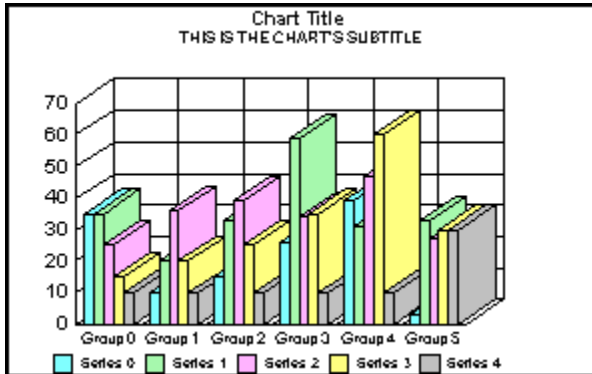
Can be one of the following:

true displays graph subtitle. This value is the default.

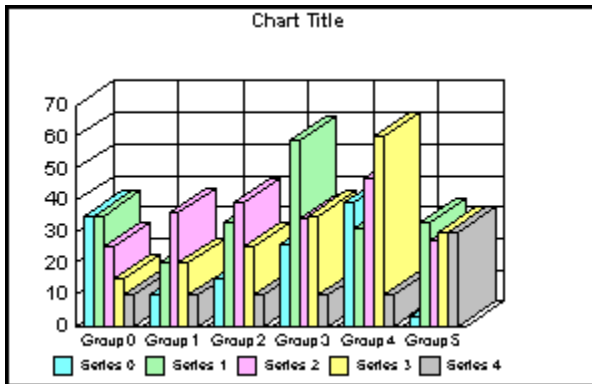
false does not display graph subtitle.

Example:

```
setSubtitleAutofit(true);
setFontSizeVC(getSubtitle(),2782);
setSubtitleString("THIS IS THE CHART'S SUBTITLE");
setSubtitleDisplay(true);
```



```
setSubtitleDisplay(false);
```



See [SubtitleString](#) on page 550.

SubtitleString

This property is used to define the subtitle text that is drawn in the graph when the SubtitleDisplay property is set to true. When a subtitle string is defined with this property, the SubtitleDisplay property is automatically set to true.

Syntax:

```
setSubtitleString("string");
```

```
string = getSubtitleString();
```

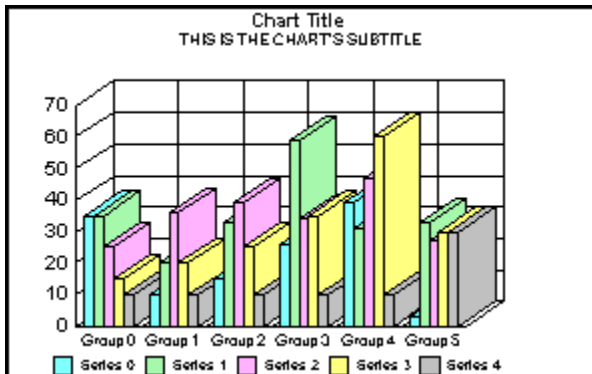
where:

string

Represents any printable characters. "Chart Subtitle" is the default value.

Example:

```
setSubtitleAutofit(true);
setFontSizeVC(getSubtitle(),2782);
setSubtitleString("THIS IS THE CHART'S SUBTITLE");
```



See [SubtitleDisplay](#) on page 549.

TextAntialiasing

This property enables or disables smoother drawing of graph text objects using Java 2D. Your development environment must include JRE 1.2 (or later) in order to use this property.

Syntax:

```
setTextAntialiasing(boolean);
boolean = getTextAntialiasing();
```

where:

boolean

Can be one of the following:

`true` enables text antialiasing.

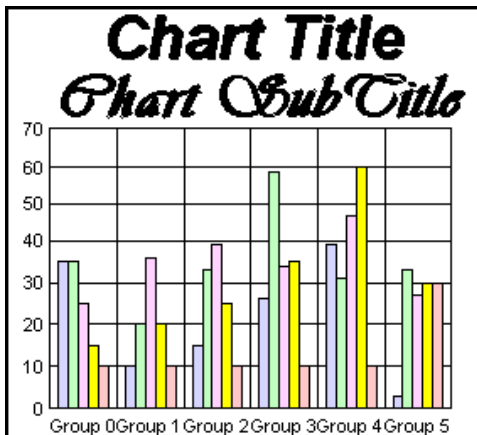
`false` disables text antialiasing.

Example:

```

setAutofit(getSubtitle(), false);
setAutofit(getTitle(), false);
setFontName(getSubtitle(), "Vivaldi");
setFontSize(getSubtitle(), 48);
setFontSize(getTitle(), 48);
setFontSizeAbsolute(getSubtitle(), true);
setFontSizeAbsolute(getTitle(), true);
setFontStyle(getTitle(), 3);
setRect(getSubtitle(), new Rectangle(-13471, 10992, 25806, 1539));
setShadowDisplay(getSubtitle(), true);
setShadowDisplay(getTitle(), true); setTextAntialiasing(false);

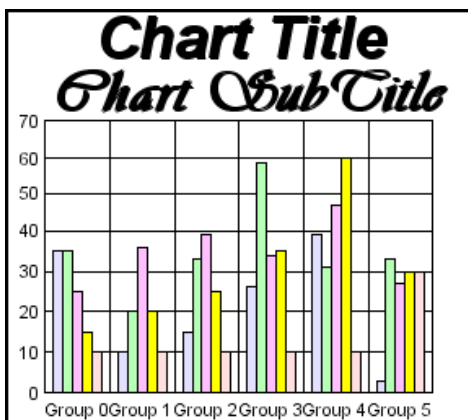
```



```

setTextAntialiasing(true);

```



See [PrimitiveAntialiasing](#) on page 497.

TitleDisplay

This property enables or disables drawing of the graph's title string that is defined by the TitleString property.

Syntax:

```
setTitleDisplay(boolean);
```

```
boolean = getTitleDisplay();
```

where:

boolean

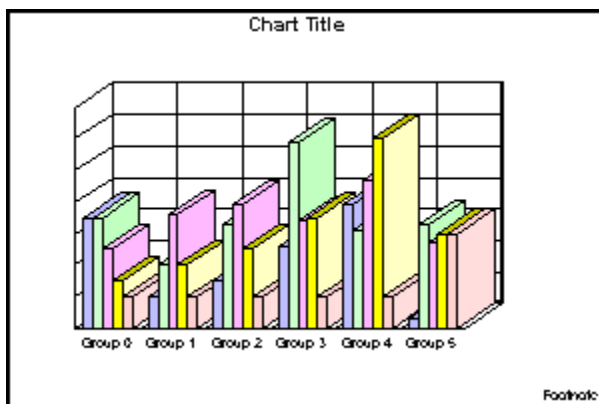
Can be one of the following:

[true](#) displays title string. This value is the default.

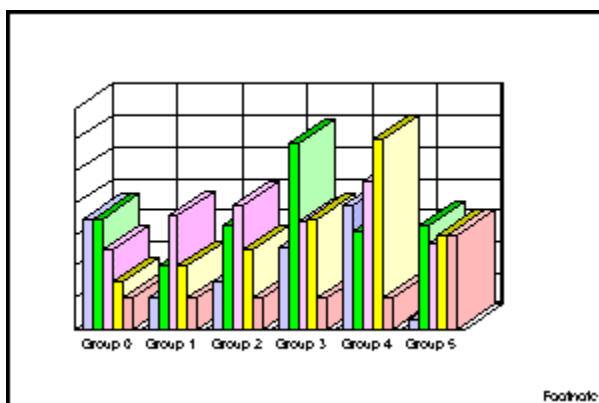
[false](#) does not display title string.

Example:

```
setTitleDisplay(true);  
setSubtitleDisplay(false);  
setLegendDisplay(false);  
setY1LabelDisplay(false);
```



```
setTitleDisplay(false);  
setSubtitleDisplay(false);  
setLegendDisplay(false);  
setY1LabelDisplay(false);
```



See [TitleString](#) on page 555.

TitleString

This property is used to define the title text that is drawn in a graph when the TitleDisplay property is set to true. When a title string is defined with this property, the TitleDisplay property is automatically set to true.

Syntax:

```
setTitleString("string");
string = getTitleString();
```

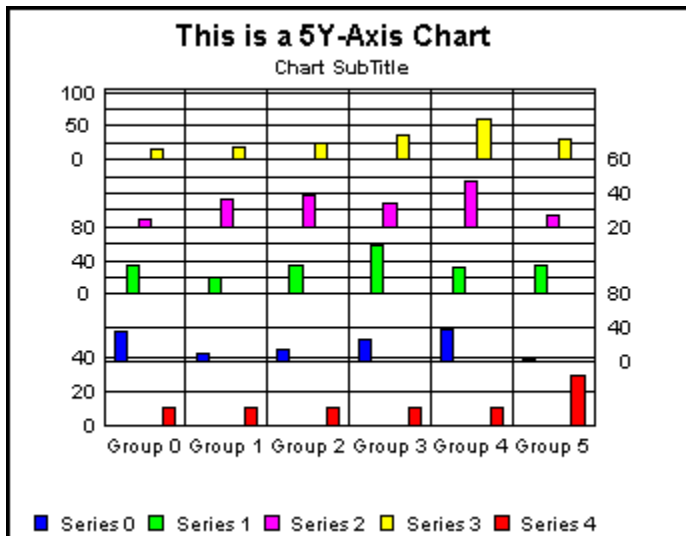
where:

string

Represents any printable characters. "Chart Title" is the default value.

Example:

```
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("This is a 5Y-Axis Chart");
```



See [TitleDisplay](#) on page 553.

ToolTipDelay

This property gets or sets the delay (in milliseconds) at which tool tips display when the ToolTipDisplay property is set to true. The mouse will need to hover over a graph object for this amount of time before the tooltip is displayed.

Syntax:

```
setToolTipDelay(value);  
value = getToolTipDelay();
```

where:

value

Is any integer. 500 is the default value.

See [ToolTipDisplay](#) on page 556, [ToolTipMode](#) on page 556.

ToolTipDisplay

This property enables or disables the display of tool tips.

Syntax:

```
setToolTipDisplay(boolean);  
boolean = getToolTipDisplay();
```

where:

boolean

Can be one of the following:

[true](#) displays tool tips.

[false](#) does not display tool tips. This value is the default.

See [ToolTipDelay](#) on page 556, [ToolTipMode](#) on page 556.

ToolTipMode

This property toggles tool tips between explicit developer information and user level information.

Syntax:

```
setToolTipMode(boolean);  
boolean = getToolTipMode();
```


where:

boolean

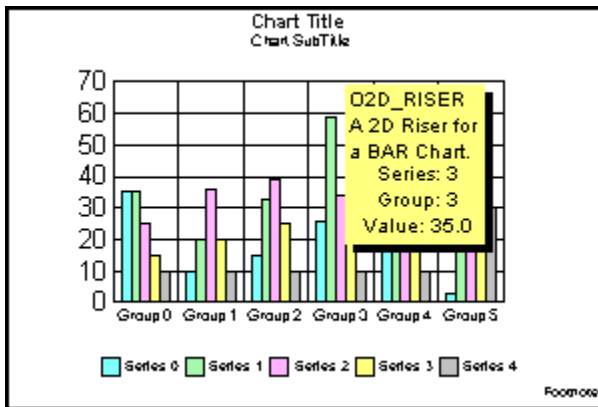
Can be one of the following:

true displays developer information

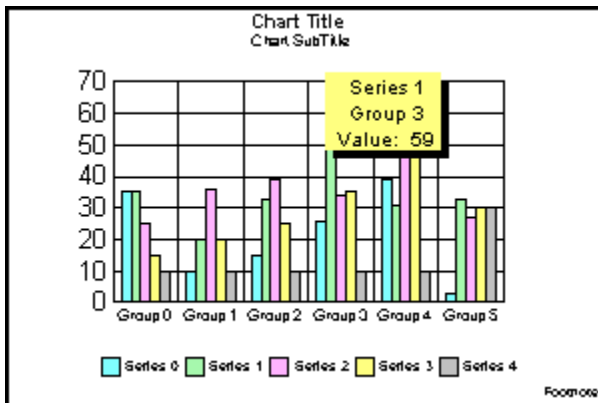
false displays user information. This value is the default.

Example:

```
setToolTipMode(true);
setToolTipDisplay(true);
setDepthRadius(0);
```



```
setToolTipMode(false);
```



See [ToolTipDelay](#) on page 556, [ToolTipDisplay](#) on page 556, [setUserToolTip\(\)](#) on page 1023.

ToolTipTimeoutEnabled

This property enables or disables the tooltip timeout set by the `setToolTipDisplayTimeOut()` method. When enabled, the tooltip will automatically disappear when the time set by `setToolTipDisplayTimeOut()` expires. When disabled, the tooltip will show as long as the mouse hovers over the object.

Syntax:

```
setToolTipTimeoutEnabled(boolean);  
boolean = getToolTipTimeoutEnabled();
```

where:

boolean

Can be one of the following:

`true` enables the tooltip display timer.

`false` disables the tooltip display timer. This value is the default.

TransparentMarkers

This property enables or disables transparent markers in a line graph. When markers are not shown with `setMarkerDisplay(false)`, this property can be used to enable transparent markers so that a tooltip will show when the mouse hovers over the area. Use `TransparentMarkerSize` to make the transparent markers large enough to be found by the mouse.

Syntax:

```
setTransparentMarkers(boolean);  
boolean = getTransparentMarkers();
```

where:

boolean

Can be one of the following:

`true` enables transparent markers

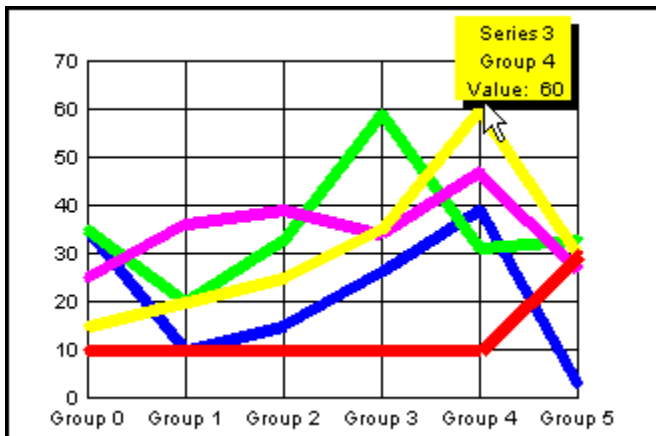
`false` disables transparent markers. This value is the default.

Example:

```

setGraphType(41);
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setSeriesLineWidthDefault(25);
    setTransparentMarkers(true);
setTransparentMarkerSize(100);
setToolTipDisplay(true);

```



See [TransparentMarkerSize](#) on page 559.

TransparentMarkerSize

When transparent markers are enabled with `TransparentMarkers(true)`, this property sets the size of the markers.

Syntax:

```

setTransparentMarkerSize(value);
integer = getTransparentMarkerSize();

```

where:

integer

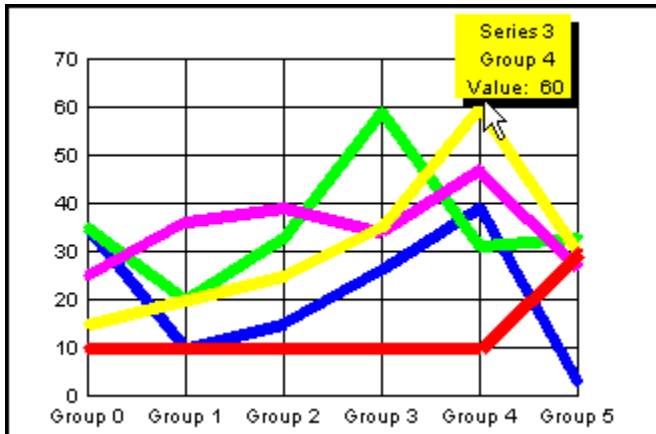
Is any integer between 0 and 100. 0 sets the size of the markers to the value set by `MarkerSizeDefault`. 0 is the default value.

Example:

```

setGraphType(41);
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setSeriesLineWidthDefault(25);
setTransparentMarkers(true);
    setTransparentMarkerSize(100);
setToolTipDisplay(true);

```



See [TransparentMarkers](#) on page 558.

UseDefaultBubbleMarker

This property controls the use of the default circle marker in bubble graphs.

Syntax:

```

setUseDefaultBubbleMarker(boolean);
boolean = getUseDefaultBubbleMarker();

```

where:

boolean

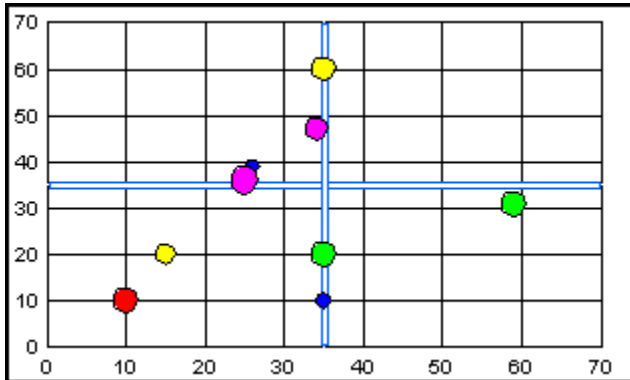
Can be one of the following:

[true](#) draws different shapes for markers. This value is the default.

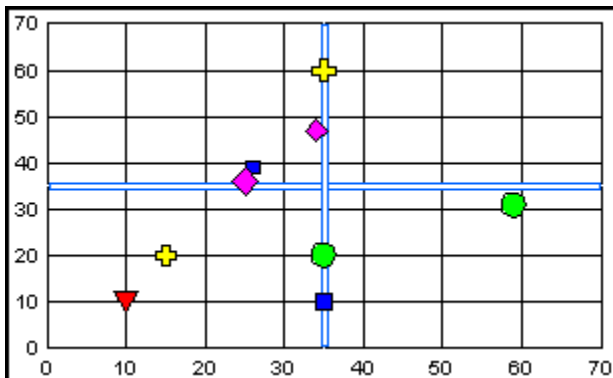
[false](#) draws all circles for markers in a bubble graph.

Example:

```
setGraphType(89);
setDepthRadius(0);
  setUseDefaultBubbleMarker(false);
setMarkerSizeDefault(75);
```



```
setGraphType(89);
setDepthRadius(0);
setMarkerSizeDefault(75);  setUseDefaultBubbleMarker(true);
```

**UseOffScreen**

This property controls whether the graph is drawn directly to the graphics device or using an off-screen image buffer. Set this property to false when the drawing speed of the graph seems very slow. When this property is set to false, it can increase the drawing speed by a factor of three in some situations.

Syntax:

```
setUseOffScreen(boolean);  
boolean = getUseOffScreen();
```

where:

boolean

Can be one of the following:

[true](#) draws directly to graphics device. This value is the default.

[false](#) uses off screen image buffer.

UseSeriesBorderDefaults

This property enables or disables the use of the riser border color set by the SeriesDefaultBorderColor property or transparent borders set by SeriesDefaultTransparentBorderColor.

Syntax:

```
setUseSeriesBorderDefaults(boolean);  
boolean = getUseSeriesBorderDefaults();
```

where:

boolean

Can be one of the following:

[true](#) uses the SeriesDefaultBorderColor for riser borders if SeriesDefaultTransparentBorderColor is false. If SeriesDefaultTransparentBorderColor is true, the property makes the riser borders transparent (that is, they are drawn with the same color as the riser's fill color).

[false](#) draws black riser borders or color defined by other methods (for example: setSelectionBorderColor(), setSeriesBorderColor(), and so on). This value is the default.

See [SeriesDefaultBorderColor](#) on page 524 and [SeriesDefaultTransparentBorderColor](#) on page 526.

UseSeriesShapes

This property enables or disables the use of different shapes for each series in the legend area.

Syntax:

```
setUseSeriesShapes(boolean);  
boolean = getUseSeriesShapes();
```

where:

boolean

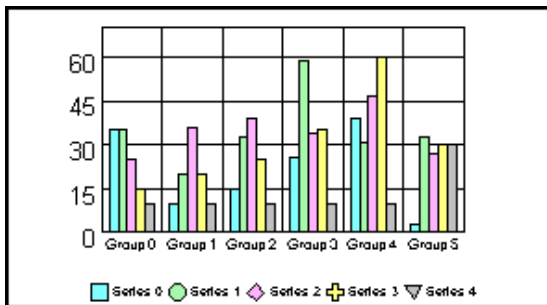
Can be one of the following:

true uses a different shape for each series in the legend area.

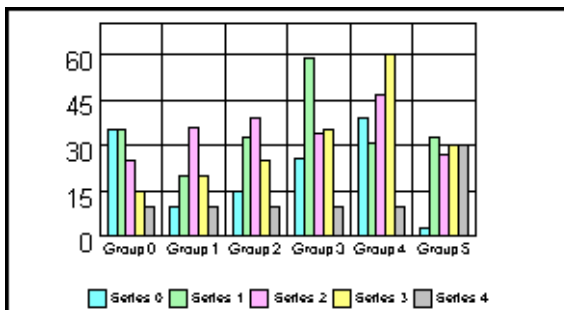
false uses a square box for each legend icon. This value is the default.

Example:

```
setUseSeriesShapes(true);
setDepthRadius(0);
```



```
setUseSeriesShapes(false);
```



See [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400, [LegendReverse](#) on page 404, [getMarkerShape\(\)](#) on page 774, [getMarkerSize\(\)](#) on page 776, [setMarkerShape\(\)](#) on page 940, [setMarkerSize\(\)](#) on page 942.

UseTimeScaleAxis

This property enables or disables use of the time scale axis. If this property is enabled, you must set the `UseSampleData` property to false and use the `setData()` methods to define the data to be graphed. Additionally, the `setLevelInterval()` and `setLevelTextFormat()` methods must be used to define the number of time intervals and the format of data that is shown at each level.

Syntax:

```
setUseTimeScaleAxis(boolean);  
boolean = getUseTimeScaleAxis();
```

where:

boolean

Can be one of the following:

`true` uses time scale axis.

`false` uses normal axis. This value is the default.

Example:

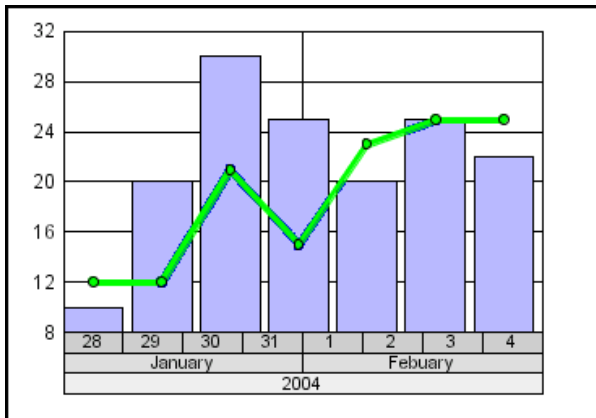
```
restoreDefaults();  
    setUseTimeScaleAxis(true);  
clearDataLabels();  
clearGroupLabels();  
clearSeriesLabels();  
setLegendDisplay(false);  
setDepthRadius(0);  
clearDataStorage();  
setUseSampleData(false);  
setData(0,0,12,28,2003);  
setData(0,1,10.0);  
setData(0,2,12,29,2003);  
setData(0,3,20.0);  
setData(0,4,12,30,2003);  
setData(0,5,30.0);  
setData(0,6,12,31,2003);  
setData(0,7,25.0);  
setData(0,8,01,01,2004);  
setData(0,9,20.0);  
setData(0,10,01,2,2004);  
setData(0,11,25.0);  
setData(0,12,01,3,2004);  
setData(0,13,22.0);  
setData(0,14,01,4,2004);  
setData(0,15,23.0);  
setData(1,0,12,28,2003);
```



```

setData(1,1,12.0);
setData(1,2,12,29,2003);
setData(1,3,12.0);
setData(1,4,12,30,2003);
setData(1,5,21.0);
setData(1,6,12,31,2003);
setData(1,7,15.0);
setData(1,8,01,01,2004);
setData(1,9,23.0);
setData(1,10,1,2,2004);
setData(1,11,25.0);
setData(1,12,1,3,2004);
setData(1,13,25.0);
setData(1,14,1,4,2004);
setStartDate(12,28,2003);
setEndDate(1,4,2004);
setDataRangeToExtent();
setGraphType(17);
setSeriesType(getSeries(0), 1);
setSeriesType(getSeries(1), 2);
setUseTimeScaleAxis(true);
setLevelInterval(0,4);
setLevelInterval(1,6);
setLevelInterval(2,8);
setLevelTextFormat(0,0);
setLevelTextFormat(1,2);
setLevelTextFormat(2,2);

```



Note: This property can only be used with clustered bar, stacked bar, line, and scatter charts.

See [setLevelInterval\(\)](#) on page 925, [setLevelTextFormat\(\)](#) on page 927, and [Time Scale Axis Graphs](#) on page 221.

ViewableGroups

This property selects the number of groups to be made visible in a graph. When this property is set to a value greater than zero, you may use the `ScrollOffsetGroups` property to define the first group to be made visible.

Syntax:

```
setVisibleGroups(value);
```

```
value = getViewableGroups();
```

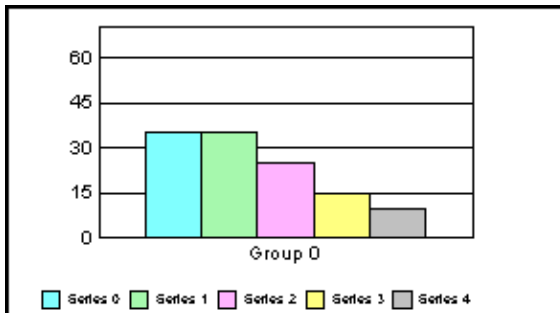
where:

value

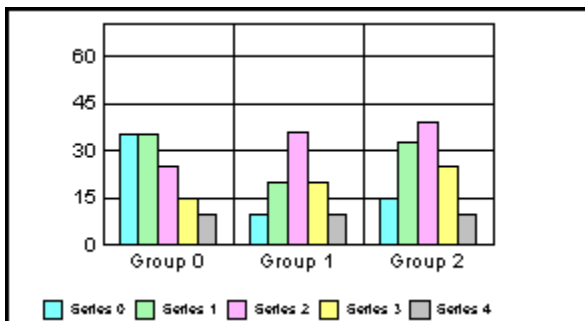
Is any integer between 0 and the number of groups in the data that defines the graph. 0 (show all groups) is the default.

Example:

```
setVisibleGroups(1);  
setDataScrollerPresenceGroups(0);  
setDepthAngle(0);  
setDepthRadius(0);
```



```
setVisibleGroups(3);
```



Note: The `setDataRange()` method can also be used to set the number of groups to be made visible in a graph. It establishes the range of data that is used in a graph.

See [ScrollOffsetGroup](#) on page 519, [ViewableGroupsStock](#) on page 567, [ViewableSeries](#) on page 568, [ViewableSeriesStock](#) on page 569, [setDataRange\(\)](#) on page 877.

ViewableGroupsStock

This property selects the number of groups to be made visible in a stock hi-lo graph.

Syntax:

```
setViewableGroupsStock(value);  
value = getViewableGroupsStock();
```

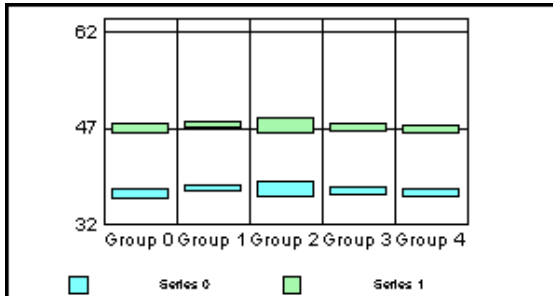
where:

value

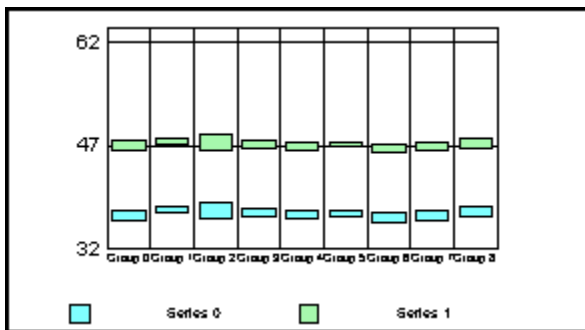
Is any integer between 0 and the number of groups in the data that defines the graph. 0 (show all groups) is the default.

Example:

```
setGraphType(73);
  setViewableGroupsStock(5);
setDataScrollerPresenceGroups(0);
```



```
setViewableGroupsStock(9)
```



Note: The `setDataRange()` method can also be used to set the number of groups to be made visible in a graph. It establishes the range of data that is used in a graph.

See [ViewableGroups](#) on page 566, [ViewableSeriesStock](#) on page 569, [setDataRange\(\)](#) on page 877.

ViewableSeries

This property selects the number of series to be made visible in a graph. When this property is set to a value greater than zero, you may use the `ScrollOffsetSeries` property to define the first series to be made visible.

Syntax:

```
setViewableSeries(value);
value = getViewableSeries();
```

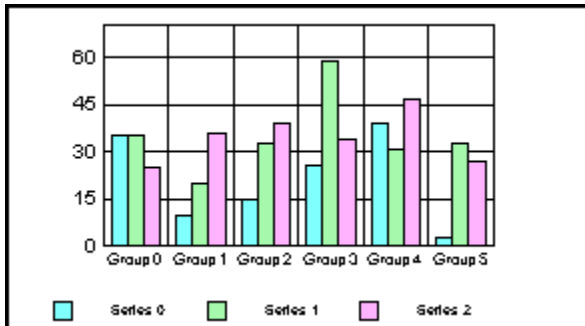
where:

value

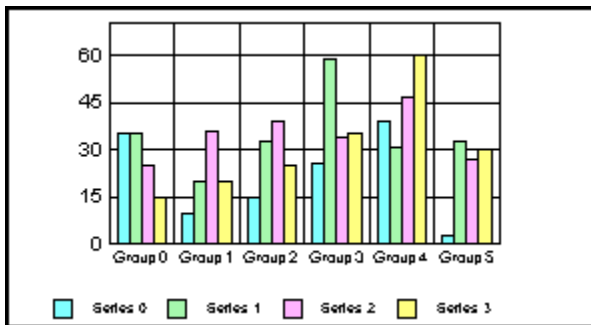
Is any integer between 0 and the number of series in the data that defines the graph. 0 (show all series) is the default value.

Example:

```
setVisibleSeries(3);
setDepthRadius(0);
```



```
setVisibleSeries(4);
```



Note: The `setDataRange()` method can also be used to set the number of series to be made visible in a graph. It establishes the range of data that is used in a graph.

See [ScrollOffsetSeries](#) on page 520, [ViewableGroups](#) on page 566, [ViewableSeriesStock](#) on page 569, [setDataRange\(\)](#) on page 877.

ViewableSeriesStock

This property selects the number of series to be made visible in a stock graph.

Syntax:

```
setViewableSeriesStock(value);
value = getViewableViewableSeriesStock();
```

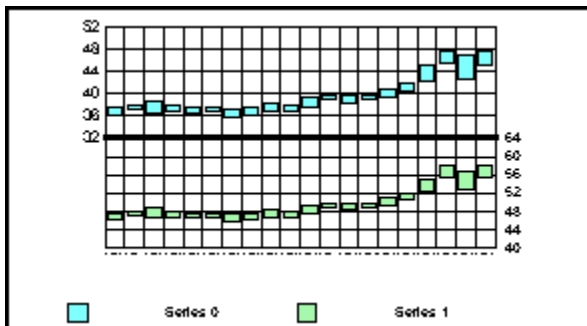
where:

value

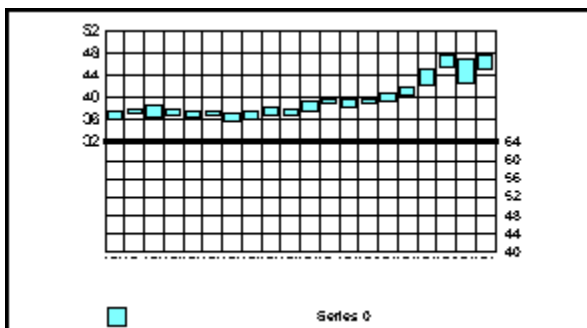
Is any integer between 0 and the number of series in the data that defines the graph. 0 (show all series) is the default value.

Example:

```
setGraphType(73); setViewableSeriesStock(2);
```



```
setViewableSeriesStock(1);
```



Note:

- ❑ This property does not apply to the following graph types: Open-Hi-Lo-Close Candle Stock Graph (Graph Type = 71), Open-Hi-Lo-Close Candle Stock Graph with Volume (Graph Type = 72), and Candle Stock Hi-Lo Open-Close Graph with Volume (Graph Type = 84).
- ❑ The `setDataRange()` method can also be used to set the number of series to be made visible in a graph. It establishes the range of data that is used in a graph.

See [ScrollOffsetSeries](#) on page 520, [ViewableGroupsStock](#) on page 567, [setDataRange\(\)](#) on page 877.

Viewing3DAnglePreset

This property selects an entry in the table of preset viewing angles for 3D graphs.

Syntax:

```
setViewing3DAnglePreset(value);
value = getViewing3DAnglePreset();
```

where:

value

Can be one of the following:

[-1](#) for Custom (as set by CubeViewerX/Y/Z, CubeWallThickX/Y/Z, and so on). 1 is the default value.

[0](#) for Standard

[1](#) for Tall and Skinny

[2](#) for From the Top

[3](#) for Distorted

[4](#) for Short and Fat

[5](#) for Groups Eye

[6](#) for Group Emphasis

[7](#) for Few Series

[8](#) for Few Groups

[9](#) for Distorted Standard

[10](#) for Shorter and Fatter

[11](#) for Thick Wall for Series

[12](#) for Thick Wall Standard

[13](#) for California Special

[14](#) for Blast-O-Vision

[15](#) for Special

Note: This property only affects 3D graphs. This property must be set to -1 (custom) in order for most Cube properties to be applied to the graph.

See [CubeFocusFactor](#) on page 267, [CubeIsometricProjection](#) on page 269, [CubeLightSourceX](#) on page 271, [CubeLightSourceY](#) on page 272, [CubeLightSourceZ](#) on page 274, [CubePanX](#) on page 275, [CubePanY](#) on page 277, [CubeSizeX](#) on page 280, [CubeSizeY](#) on page 281, [CubeSizeZ](#) on page 283, [CubeSquareRisers](#) on page 284, [CubeTranslationX](#) on page 286, [CubeTranslationY](#) on page 288, [CubeTranslationZ](#) on page 290, [CubeViewerX](#) on page 292, [CubeViewerY](#) on page 293, [CubeViewerZ](#) on page 294, [CubeWallThickX](#) on page 296, [CubeWallThickY](#) on page 298, [CubeWallThickZ](#) on page 300, [CubeZoomFactor](#) on page 302.

See [3D Preset Viewing Angles](#) on page 1270 for an illustration of each preset viewing angle.

WaterfallLastGroupTotal

This property makes the last riser in a Waterfall chart into a total that touches the x-axis.

Syntax:

```
setWaterfallLastGroupTotal(boolean);  
boolean = getWaterfallLastGroupTotal();
```

where:

boolean

Can be one of the following:

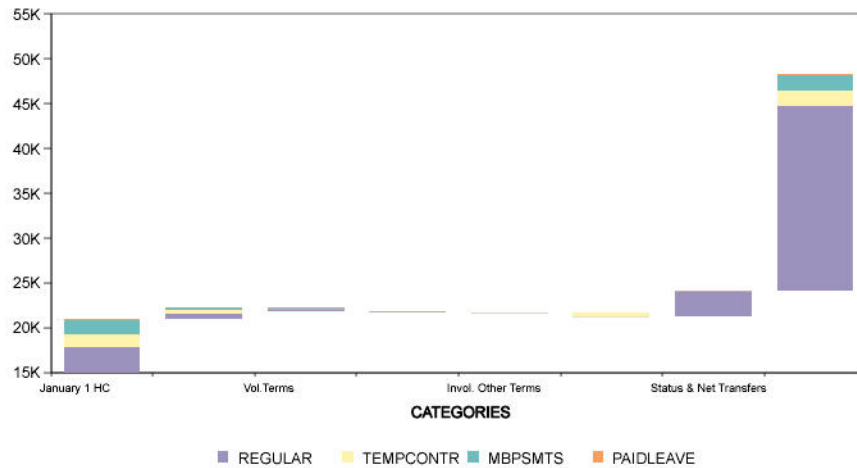
[true](#) makes the last riser into a total touching the x-axis.

[false](#) does not make the last riser into a total. This value is the default.

Example:

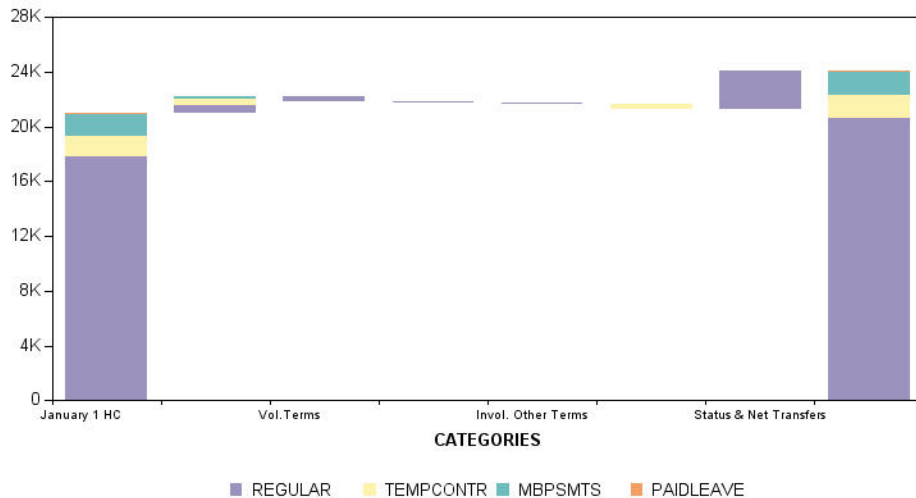
With `setWaterfallLastGroupTotal(false)`, all of the risers in the Waterfall chart are normal risers:

`setWaterfallLastGroupTotal(false);`



With `setWaterfallLastGroupTotal(true)`, the last riser represents the total and touches the x-axis:

`setWaterfallLastGroupTotal(true);`



WaterfallStackColorMode

This property is used to set color mode to distinguish positive and negative numbers by color in Waterfall charts.

Syntax:

```
setWaterfallStackColorMode(boolean);  
boolean = getWaterfallStackColorMode();
```

where:

boolean

Can be one of the following:

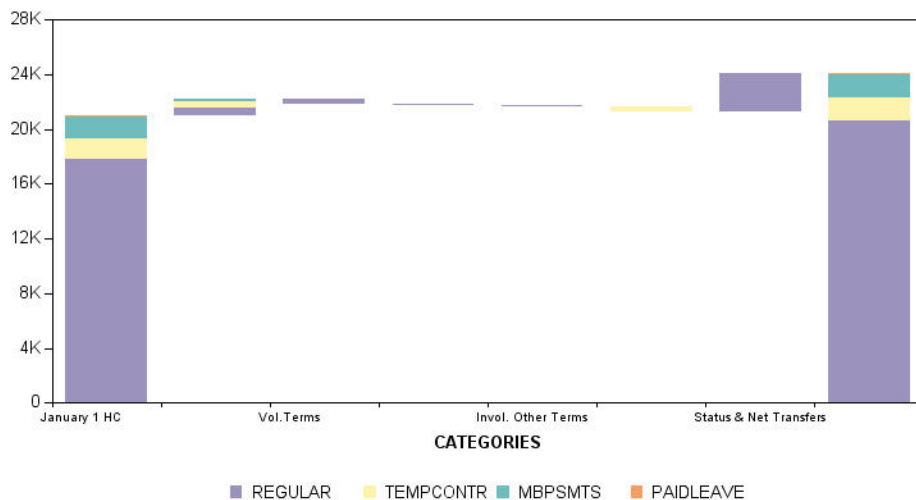
[true](#) displays positive values in green and negative values in red.

[false](#) does not use color to distinguish between positive and negative numbers. This value is the default.

Example:

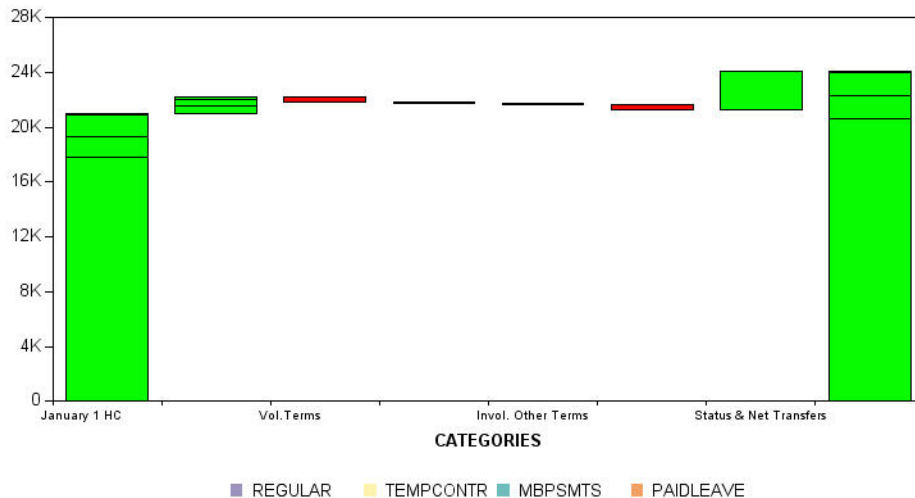
With `setWaterfallStackColorMode(false)`, the colors in a Waterfall chart distinguish between the different series:

```
setWaterfallStackColorMode(false);
```



With `setWaterfallStackColorMode(true)`, the colors in a Waterfall chart distinguish between positive (green) and negative (red) values:

```
setWaterfallStackColorMode(true);
```



X1AxisDescending

This property is used to change the direction of the X-axis.

Note: This property also reverses the order of the axis labels and changes all data-related objects (for example, markers) from their previous state.

Syntax:

```
setX1AxisDescending(boolean);
```

```
boolean = getX1AxisDescending();
```

where:

boolean

Can be one of the following:

true draws axis labels in descending order from left to right on the X-axis.

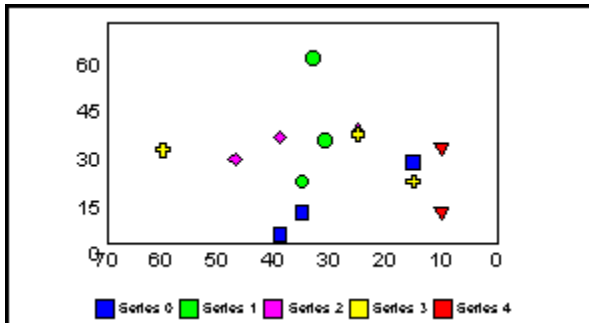
false draws axis labels in ascending order from left to right on the X-axis. This value is the default.

Example:

```

setX1AxisDescending(true);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);

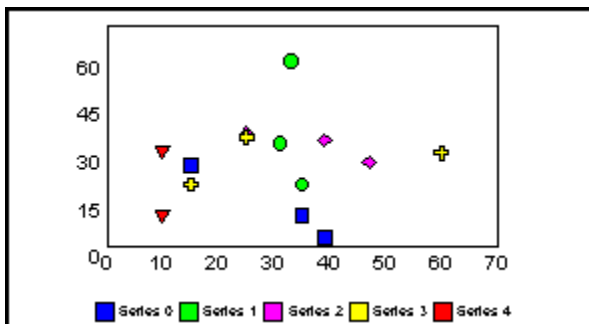
```



```

setX1AxisDescending(false);

```



See [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853.

X1AxisLineDisplay

This property enables or disables the display of the X1-axis line. The X1AxisSide property determines where the X1-axis line is drawn.

Syntax:

```

setX1AxisLineDisplay(boolean);
boolean = getX1AxisLineDisplay();

```

where:

boolean

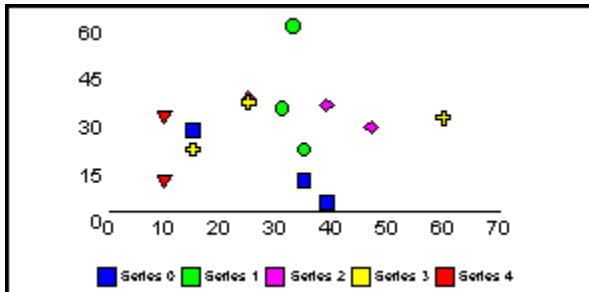
Can be one of the following:

true draws X1 axis base line. This value is the default.

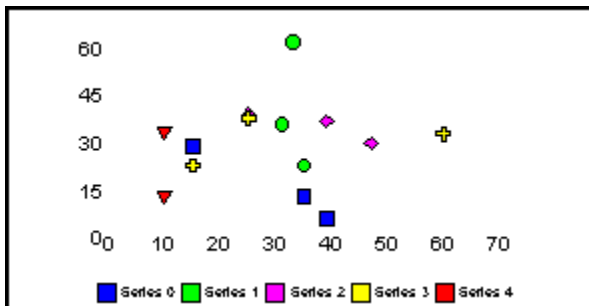
false does not draw X1 axis base line.

Example:

```
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
    setX1AxisLineDisplay(true);
setX1MajorGridDisplay(false);
setFrameDisplay(false);
```



`setX1AxisLineDisplay(true);`



Note: In the default configuration for this graph type, the X1AxisLine is the same width and location as the bottom of the graph frame. When this line size is used, this property does not change the appearance of the graph. To make a thicker, more visible axis line, use `getX1AxisLine()` to get the object ID of the line and `setLineWidth()` method to increase the width of the line.

See [X1AxisSide](#) on page 578.

X1AxisSide

This property controls which side of the graph the X-axis base line, labels, and titles are imaged.

Syntax:

```
setX1AxisSide(value);  
value = getX1AxisSide();
```

where:

value

Can be one of the following:

[0](#) for image to low side of graph (bottom or left). 0 is the default value.

[1](#) for image to high side of graph (top or right).

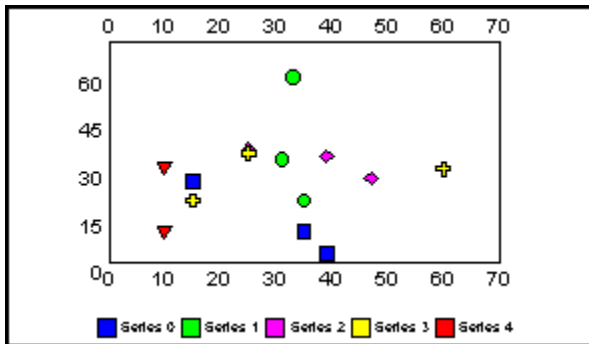
[2](#) for image to both sides.

Example:

```

setMarkerSizeDefault(60);
  setX1AxisSide(2);
setDepthAngle(0);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);
setDepthRadius(0);

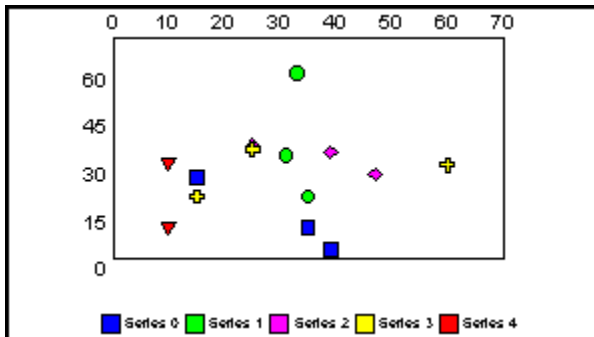
```



```

setX1AxisSide(1);

```



See [X1AxisLineDisplay](#) on page 576.

X1ExcludeMaxLabel

This property excludes or includes the maximum label on the X1-axis.

Syntax:

```

setX1ExcludeMaxLabel(boolean);
boolean = getX1ExcludeMaxLabel();

```

where:

boolean

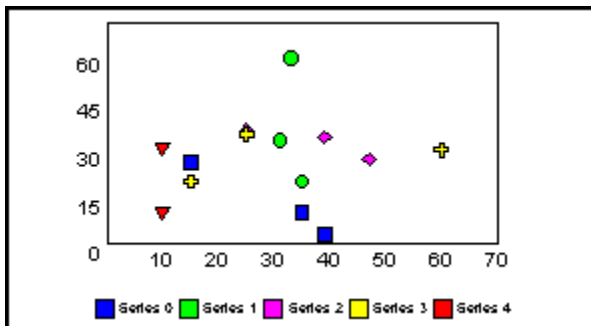
Can be one of the following:

true excludes maximum label.

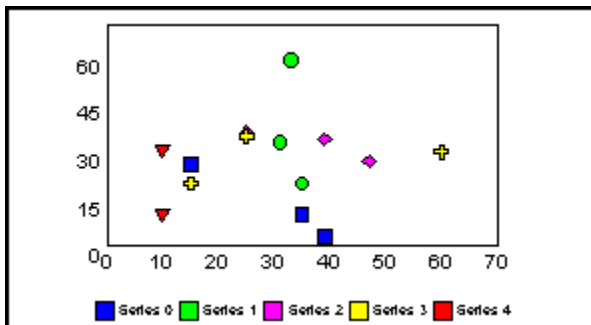
false includes maximum label. This value is the default.

Example:

```
setX1ExcludeMaxLabel(true);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);
```



```
setX1ExcludeMaxLabel(false);
```



Note: This property will not affect the appearance of the graph if drawing of X1-axis labels has been disabled with `setX1LabelDisplay(false)`; When `setX1LabelDisplay(true)` is used, the maximum label will be included or excluded as defined by `X1ExcludeMaxLabel`.

See [X1ExcludeMinLabel](#) on page 581, [X1LabelDisplay](#) on page 582.

X1ExcludeMinLabel

This property excludes or includes the minimum label on the X1-axis.

Syntax:

```
setX1ExcludeMinLabel(boolean);  
boolean = getX1ExcludeMinLabel();
```

where:

boolean

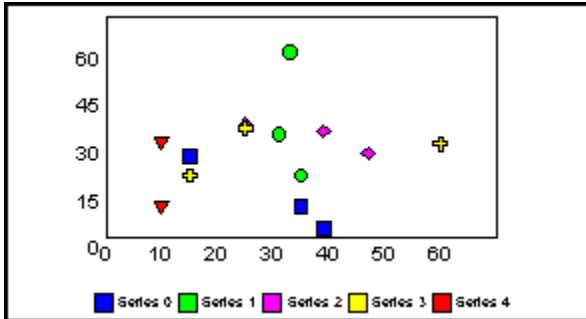
Can be one of the following:

[true](#) excludes minimum label.

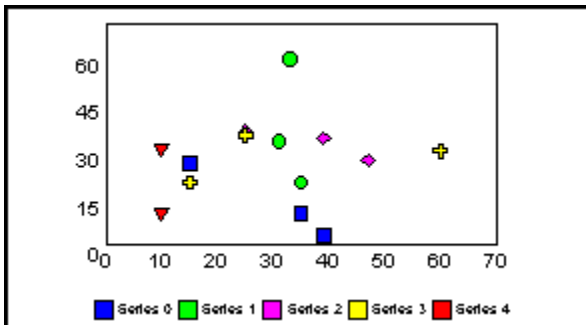
[false](#) includes minimum label. This value is the default.

Example:

```
setMarkerSizeDefault(60);    setX1ExcludeMinLabel(true);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);
```



```
setX1ExcludeMinLabel(false);
```



Note: This property will not affect the appearance of the graph if drawing of X1-axis labels has been disabled with `setX1LabelDisplay(false)`; When `setX1LabelDisplay(true)` is used, the minimum label will be included or excluded as defined by `X1ExcludeMinLabel`.

See [X1ExcludeMaxLabel](#) on page 579, [X1LabelDisplay](#) on page 582.

X1LabelDisplay

This property enables or disables the display of labels on the X-axis.

Syntax:

```
setX1LabelDisplay(boolean);
```

```
boolean = getX1LabelDisplay();
```

where:

boolean

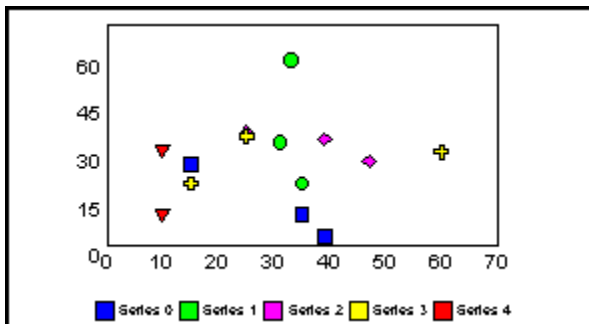
Can be one of the following:

true displays X1-axis labels and title. The X1AxisSide property determines the location where the labels are displayed. This value is the default.

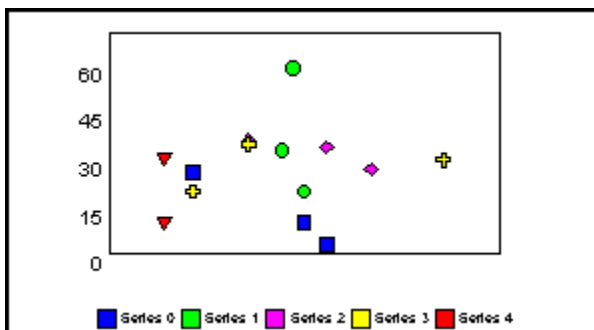
false does not display X1-axis labels and title.

Example:

```
setMarkerSizeDefault(60);
setDepthAngle(0);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);
  setX1LabelDisplay(true);
setDepthRadius(0);
```



```
setX1LabelDisplay(false);
```



See [X1AxisSide](#) on page 578.

X1LabelFormat

This property defines the format of numeric data labels on the X1 axis.

Syntax:

```
setX1LabelFormat(value);
```

```
value = getX1LabelFormat();
```

where:

value

Is any integer between -1 and 21.

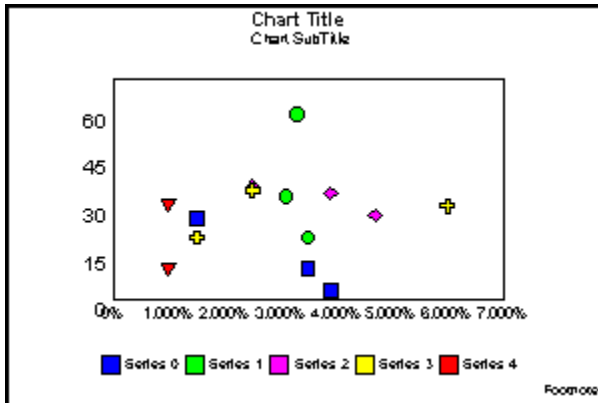
-1 uses the pattern defined by X1LabelFormatPattern.

0 through 21 selects a preset number format. 0 (General) is the default value.

See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setX1MajorGridDisplay(false);
perspective1.setY1MajorGridDisplay(false);
setGraphType(61);
setX1LabelFormat(6);
```



See [X1LabelFormatPattern](#) on page 585.

X1LabelFormatPattern

When X1LabelFormat is set to minus one, use this property to define a standard Java data text format.

Syntax:

```
setX1LabelFormatPattern(pattern);  
pattern = getX1LabelFormatPattern();
```

where:

pattern

Is a pattern string in the following format:

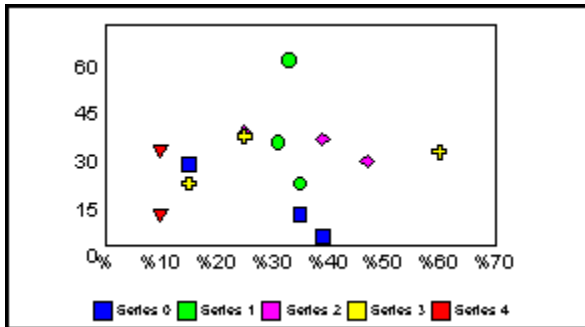
```
pattern:= subpattern{;subpattern}subpattern:=  
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFF' -  
specialCharacterssuffix:= '\\u0000'..'\\uFFFF' -  
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

"#.#" is the default value.

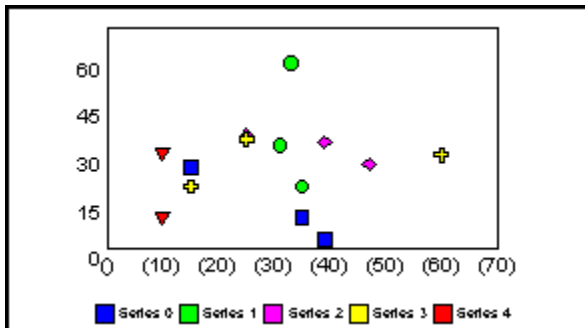
See [Formatting Numbers and Numeric Labels](#) on page 130 for more detailed information about this format pattern.

Example:

```
setGraphType(61);
setX1LabelFormat(-1);  setX1LabelFormatPattern("%#. #");
```



```
setX1LabelFormatPattern("(#. #)");
```



See [X1LabelFormat](#) on page 584.

X1LabelRotate

This property can be used to rotate labels on the X-axis.

Syntax:

```
setX1LabelRotate(value);
value = getX1LabelRotate();
```

where:

value

Can be one of the following:

[0](#) for no text rotation. 0 is the default value.

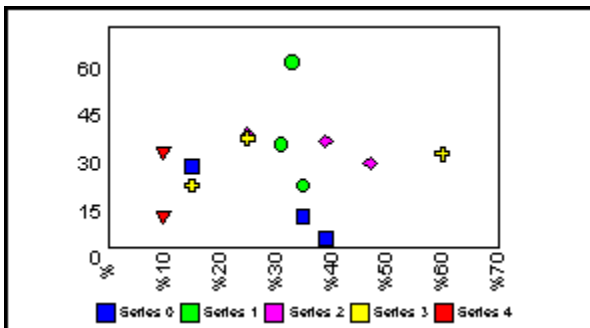
- 1 to rotate text 90 degrees (reads top to bottom).
- 2 to rotate text 270 degrees (reads bottom to top).
- 3 to rotate text 45 degrees.

Example:

```

setX1LabelRotate(1);
setMarkerSizeDefault(60);
setY1MajorGridDisplay(false);
setGraphType(61);
setX1LabelFormat(-1);
setY1AxisLineDisplay(false);
setX1LabelFormatPattern("%#. #");
setX1MajorGridDisplay(false);

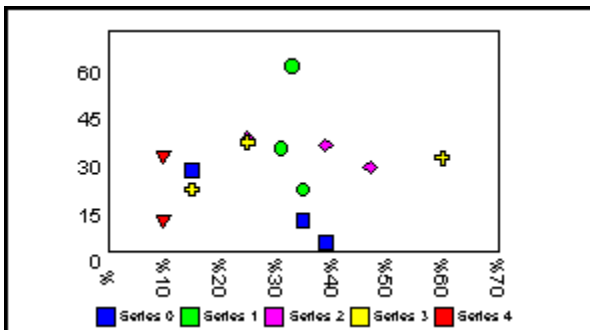
```



```

setX1LabelRotate(1);

```



See [X1LabelDisplay](#) on page 582, [X1LabelFormat](#) on page 584, [X1LabelStagger](#) on page 588.

X1LabelStagger

This property specifies whether or not text for this graph axis should be staggered. Staggered labels are drawn in a zigzag fashion, with labels weaving in and under each other. This property affects all X-axis labels.

Syntax:

```
setX1LabelStagger(boolean);  
boolean = getX1LabelStagger();
```

where:

boolean

Can be one of the following:

`true` staggers labels.

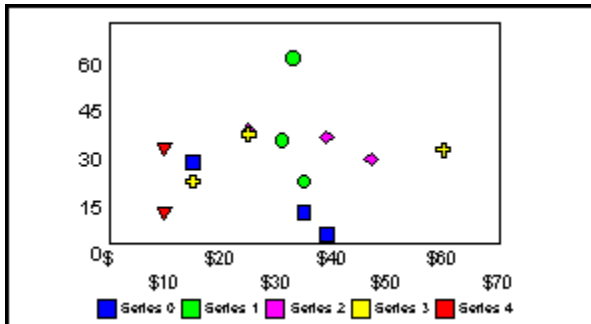
`false` draws labels on one line. This value is the default.

Example:

```

setX1LabelStagger(true);
setY1MajorGridDisplay(false);
setGraphType(61);
setX1LabelFormat(-1);
setY1AxisLineDisplay(false);
setX1LabelFormatPattern("$#. #");
setX1MajorGridDisplay(false);

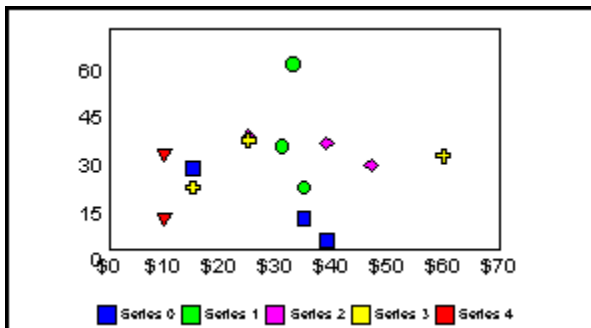
```



```

setX1LabelStagger(false);

```



See [X1LabelDisplay](#) on page 582, [X1LabelFormat](#) on page 584, [X1LabelRotate](#) on page 586, [getLabelStagger\(\)](#) on page 764, [setLabelStagger\(\)](#) on page 918.

X1LogScale

This property specifies the scale (logarithmic or linear) for the numeric X-axis. When the scale range is set to automatic, any data items with X1 values less than or equal to zero are graphed as null for log scale. If a manual scale range is specified to include negative values (X1ScaleMin less than or equal to zero), the request for log scale is ignored. X1LogScale is automatically set to false (linear scale).

Syntax:

```
setX1LogScale(boolean);
boolean = getX1LogScale();
```

where:

boolean

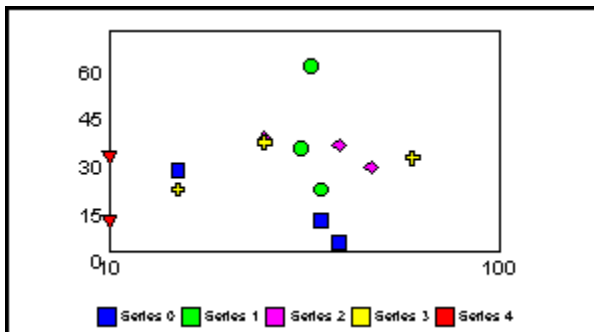
Can be one of the following:

`true` uses logarithmic scale.

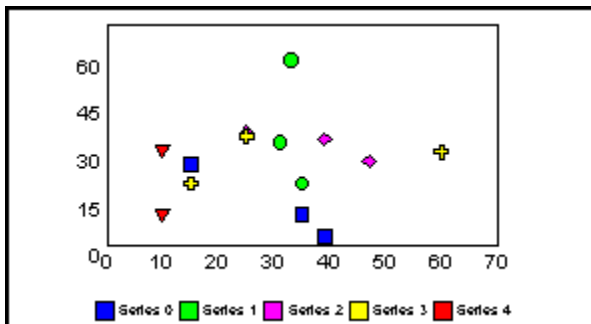
`false` uses linear scale. This value is the default.

Example:

```
setMarkerSizeDefault(60);
setY1MajorGridDisplay(false);
  setX1LogScale(true);
setGraphType(61);
setY1AxisLineDisplay(false);
setX1MajorGridDisplay(false);
```



```
setX1LogScale(false);
```



See [getLogScale\(\)](#) on page 772, [setLogScale\(\)](#) on page 938.

X1MajorGridDisplay

This property controls whether or not major grid lines are displayed on the numeric X-axis. The X-axis can have major and minor grid lines.

Syntax:

```
setX1MajorGridDisplay(boolean);  
boolean = getX1MajorGridDisplay();
```

where:

boolean

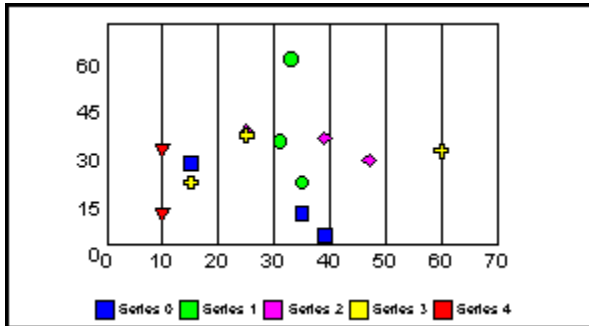
Can be one of the following:

[true](#) draws major grid lines on the X-axis, but does not draw minor grid lines. Use the X1MajorGridStep property to control the number of grid lines that are drawn. This value is the default.

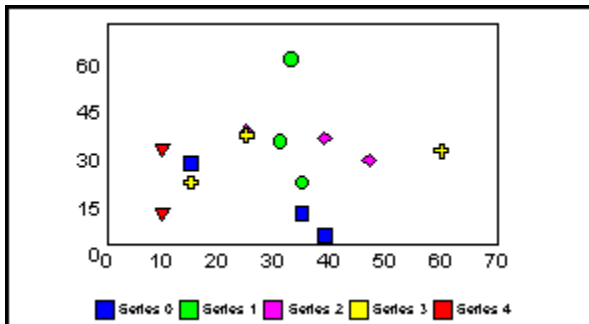
[false](#) does not draw major grid lines on the X-axis.

Example:

```
setMarkerSizeDefault(60);
setY1MajorGridDisplay(false);
setGraphType(61);
setY1AxisLineDisplay(false);  setX1MajorGridDisplay(true);
```



```
setX1MajorGridDisplay(false);
```



See [X1MajorGridStep](#) on page 592, [X1MajorGridStyle](#) on page 596, [X1MinorGridDisplay](#) on page 598, [X1MinorGridStyle](#) on page 603.

X1MajorGridStep

This property can be used to control the number of major grid lines or steps that will be imaged on the X-axis of the graph. For example, grid steps 1, 6, 11, 16, ...51 will be imaged if the range of values in the graph is 1...51 and this property is set to 5. If this property is set to 10, grid steps 1, 11, 21, 31...51 would appear. Assigning a value to this property will automatically set the X1MajorGridStepAuto property to false.

Syntax:

```
setX1MajorGridStep(value);
```

```
value = getX1MajorGridStep();
```

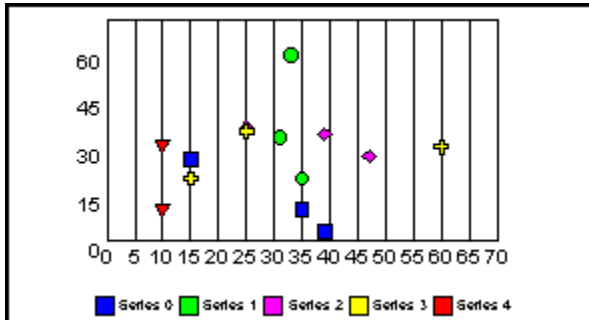
where:

value

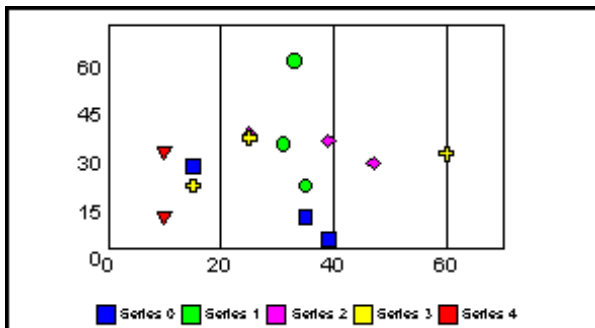
Is any positive value. This value is a Java double-precision number. 15.0 is the default value.

Example:

```
setX1MajorGridStep(5.0);
setGraphType(61);
setX1MajorGridStepAuto(false);
```



```
setX1MajorGridStep(20.0);
```



Note: If the value assigned to X1MajorGridStep is negative or results in greater than 50 grid lines, the X1MajorGridStepAuto property is automatically set to true and the X1MajorGridStep value is ignored.

See [X1MajorGridStepAuto](#) on page 594, [X1MinorGridStep](#) on page 599, [X1MinorGridStepAuto](#) on page 601, [getGridStep\(\)](#) on page 758, [setGridStep\(\)](#) on page 910, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911.

X1MajorGridStepAuto

This property enables or disables automatic calculation of the number of major grid lines on the X-axis.

Syntax:

```
setX1MajorGridStepAuto(boolean);
```

```
boolean = getX1MajorGridStepAuto();
```

where:

boolean

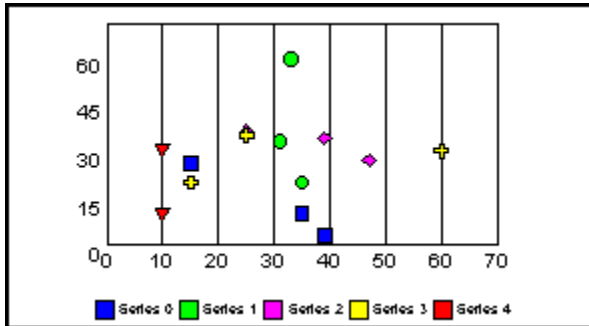
Can be one of the following:

[true](#) automatically calculates the number of grid steps on the X-axis. The value assigned to X1MajorGridStep is ignored. This value is the default.

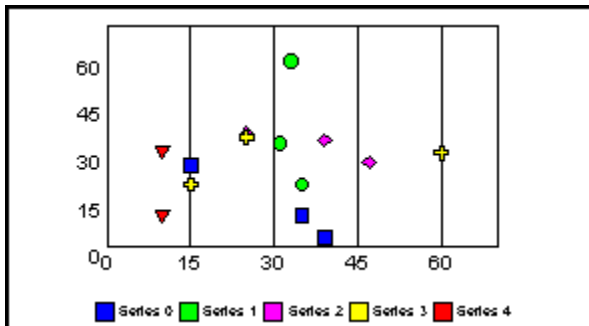
[false](#) uses the value set by X1MajorGridStep.

Example:

```
setMarkerSizeDefault(60);
setGraphType(61);
setX1MajorGridStep(15);
    setX1MajorGridStepAuto(true);
```



```
setX1MajorGridStepAuto(false);
```



Note: If the value assigned to X1MajorGridStep is negative or results in greater than 50 grid lines, the X1MajorGridStepAuto property is automatically set to true and the X1MajorGridStep value is ignored.

See [X1MajorGridStep](#) on page 592, [X1MinorGridStep](#) on page 599, [X1MinorGridStepAuto](#) on page 601, [getGridStep\(\)](#) on page 758, [setGridStep\(\)](#) on page 910, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911, [getX1MajorGridStepAutoValue\(\)](#) on page 826.

X1MajorGridStyle

This property controls the appearance of major grid lines on the numeric X-axis of a graph. Grid lines can be small ticks at the base of the axis line or lines that extend to the height of the graph frame. Major grid lines are normally set to a different style than minor grid lines so that the user can distinguish between the two.

Syntax:

```
setX1MajorGridStyle(value);  
value = getX1MajorGridStyle();
```

where:

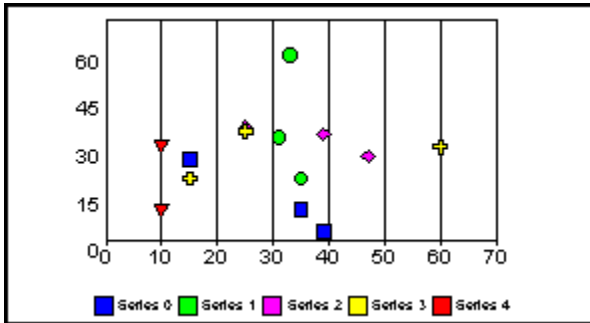
value

Can be one of the following:

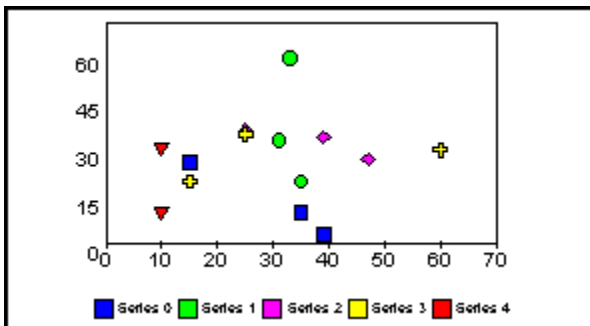
- [0](#) for normal grid lines, height of frame. 0 is the default value.
- [1](#) for normal grid lines extend beyond the height of frame.
- [2](#) for small tick marks from frame edge inward.
- [3](#) for small tick marks from frame edge outward.
- [4](#) for small tick marks span across the frame edge.

Example:

```
setY1MajorGridDisplay(false);
  setX1MajorGridStyle(1);
setGraphType(61);
```



```
setX1MajorGridStyle(4);
```



See [X1MajorGridDisplay](#) on page 591, [X1MinorGridDisplay](#) on page 598, [X1MinorGridStyle](#) on page 603, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

X1MajorTickDisplay

This property enables or disables drawing of major tick marks on the X-axis.

Syntax:

```
setX1MajorTickDisplay (boolean);
boolean = getX1MajorTickDisplay ();
```

where:

boolean

Can be one of the following:

`true` draws major tick marks.

`false` does not draw major tick marks. This value is the default.

X1MajorTickStyle

When major tick marks are drawn on the X-axis with X1MajorTickDisplay or setDisplay(getX1MajorTick(), true), this property defines the style of major tick marks.

Syntax:

```
setX1MajorTickStyle(value);
```

```
value = getX1MajorTickStyle ();
```

where:

value

Can be one of the following:

`0` for small tick marks from frame edge outward. This value is the default.

`1` for small tick marks from frame edge outward.

`2` for small tick marks from frame edge inward.

`3` for small tick marks from frame edge outward.

`4` for small tick marks span across the frame edge.

X1MinorGridDisplay

This property enables or disables the display of minor grid lines on the X-axis. The X-axis can have major and minor grid lines.

Syntax:

```
setX1MinorGridDisplay(boolean);
```

```
boolean = getX1MinorGridDisplay();
```

where:

boolean

Can be one of the following:

`true` draws minor grid lines on the X-axis.

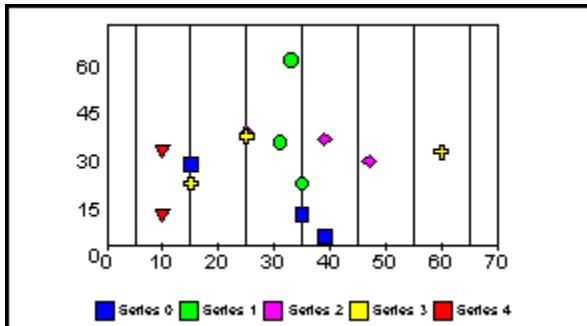
`false` does not draw minor grid lines on the X-axis. This value is the default.

Example:

```

setMarkerSizeDefault(60);
setY1MajorGridDisplay(false);
setX1MajorGridStyle(4);
setGraphType(61);
setX1MinorGridStepAuto(false);
setY1AxisLineDisplay(false);  setX1MinorGridDisplay(true);

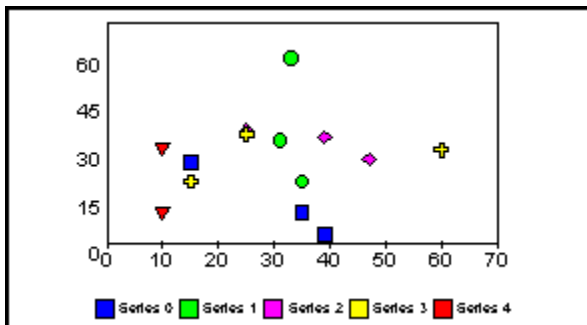
```



```

setX1MinorGridDisplay(false);

```



See [X1MajorGridDisplay](#) on page 591, [X1MinorGridStep](#) on page 599, [X1MinorGridStepAuto](#) on page 601, [X1MinorGridStyle](#) on page 603, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

X1MinorGridStep

This property sets a number of minor grid steps on the X-axis of the graph. When they are displayed, minor grid lines are drawn between the major grid lines and the location of the minor gridlines depends very much on the setting for the major grid lines. When a value is assigned to this property, the `X1MinorGridStepAuto` property is automatically set to false. If the value assigned to this property results in greater than 50 minor grid lines between each major grid line, the `X1MinorGridStepAuto` property is automatically set to true.

Syntax:

```
setX1MinorGridStep(value);  
value = getX1MinorGridStep();
```

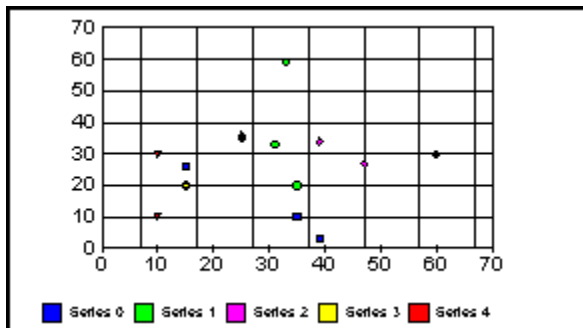
where:

value

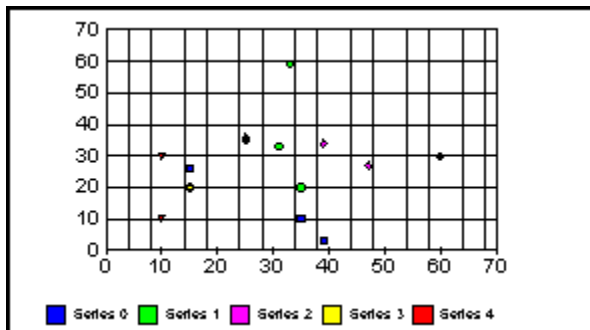
Is any positive value. This value is a Java double-precision number. 15.0 is the default value.

Example:

```
setGraphType(61);  
setX1MinorGridStep(7.0);
```



```
setX1MinorGridStep(4.0);
```



Note: If X1MinorGridStep is set to a negative value or a value that results in greater than 50 minor grid lines between each major grid line, the X1MinorGridStepAuto property is automatically set to true.

See [X1MinorGridStepAuto](#) on page 601, [X1MajorGridStyle](#) on page 596, [X1MajorGridStepAuto](#) on page 594, [getGridStep\(\)](#) on page 758, [setGridStep\(\)](#) on page 910, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911.

X1MinorGridStepAuto

This property enables or disables automatic grid steps for the X-axis minor grid lines.

Syntax:

```
setX1MinorGridStepAuto(boolean);
```

```
boolean = getX1MinorGridStepAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates the number of grid lines on the X-axis. The value assigned to the X1MinorGridStep property is ignored. This value is the default.

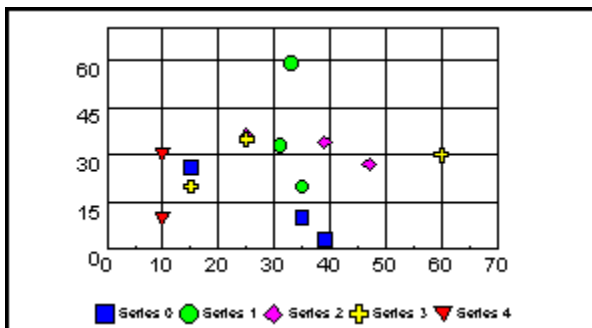
[false](#) uses the value set by X1MinorGridStep.

Example:

```

setX1MinorGridStyle(2);
setGraphType(61);
setX1MinorGridDisplay(true);
setX1MinorGridStep(4.0);  setX1MinorGridStepAuto(true);

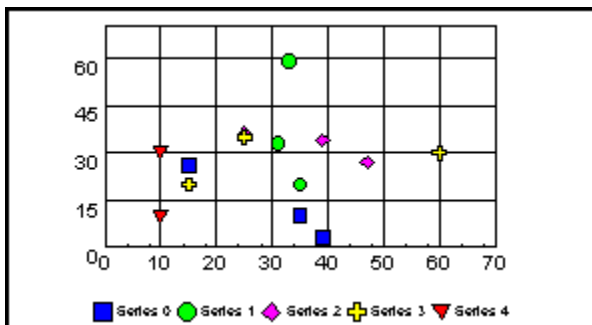
```



```

setX1MinorGridStepAuto(false);

```



Note: If X1MinorGridStep is set to a negative value or a value that results in greater than 50 minor grid lines between each major grid line, the X1MinorGridStepAuto property is automatically set to true.

See [X1MinorGridStep](#) on page 599, [X1MajorGridStep](#) on page 592, [X1MajorGridStepAuto](#) on page 594, [getGridStep\(\)](#) on page 758, [setGridStep\(\)](#) on page 910, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911.

X1MinorGridStyle

This property controls the appearance of minor grid lines on the numeric X-axis of a graph. Grid lines can be small ticks at the base of the axis line or lines that extend to the height of the graph frame. Minor grid lines are normally set to a different style than major grid lines so that the user can distinguish between the two.

Syntax:

```
setX1MinorGridStyle(value);  
value = getX1MinorGridStyle();
```

where:

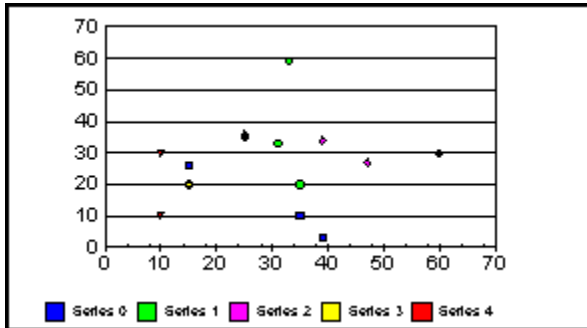
value

Can be one of the following:

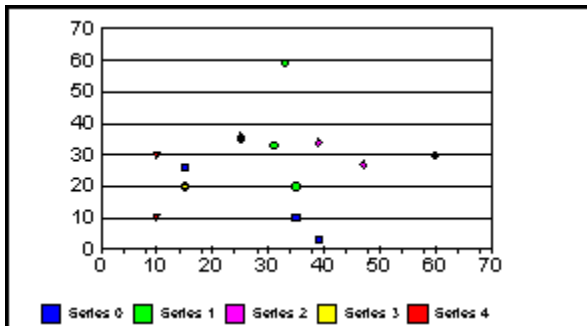
- 0 for normal grid lines, height of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```
setX1MinorGridStyle(2);
setGraphType(61);
setX1MinorGridDisplay(true);
```



```
setX1MinorGridStyle(3);
```



See [X1MinorGridDisplay](#) on page 598, [X1MinorGridStep](#) on page 599, [X1MinorGridStepAuto](#) on page 601, [X1MajorGridStyle](#) on page 596, [X1MajorGridDisplay](#) on page 591, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

X1MinorTickDisplay

This property enables or disables drawing of minor tick marks on the X-axis.

Syntax:

```
setX1MinorTickDisplay (boolean);
boolean = getX1MinorTickDisplay ();
```


where:

boolean

Can be one of the following:

true draws minor tick marks.

false does not draw minor tick marks. This value is the default.

X1MinorTickStyle

When minor tick marks are drawn on the X-axis with X1MinorTickDisplay or setDisplay(getX1MinorTick(), true), this property defines the style of minor tick marks.

Syntax:

```
setX1MinorTickStyle(value);
```

```
value = getX1MinorTickStyle ();
```

where:

value

Can be one of the following:

0 for small tick marks from frame edge outward. This value is the default.

1 for small tick marks from frame edge outward.

2 for small tick marks from frame edge inward

3 for small tick marks from frame edge outward

4 for small tick marks span across the frame edge

X1MustIncludeZero

This property determines whether or not a graph includes zero. If the logarithmic scale is enabled with X1LogScale(true) or a manual scale range excludes zero (X1ScaleMax and X1ScaleMin are greater than zero), this property is ignored.

Syntax:

```
setX1MustIncludeZero(boolean);
```

```
boolean = getX1MustIncludeZero();
```

where:

boolean

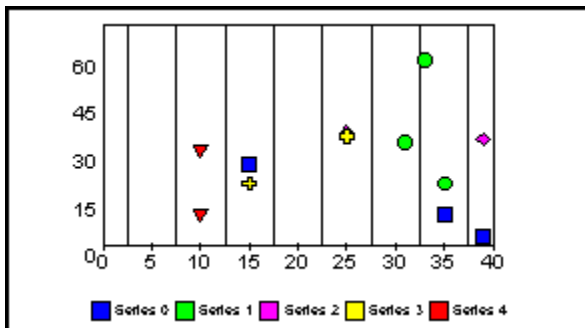
Can be one of the following:

`true` includes zero.

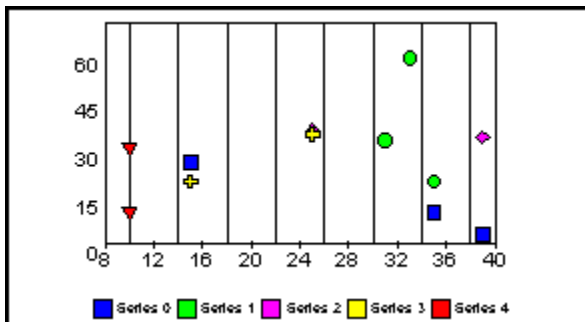
`false` excludes zero if it is not within the given automatic scale range. This value is the default.

Example:

```
setMarkerSizeDefault(60);
setX1OffScaleDisplay(false);
setX1ScaleMax(40.0);
    setX1MustIncludeZero(true);
setY1MajorGridDisplay(false);
setX1MajorGridStyle(4);
setGraphType(61);
```



`setX1MustIncludeZero(false);`



See [X1ScaleMax](#) on page 608, [X1ScaleMin](#) on page 611, [X1ZeroLineDisplay](#) on page 616.

X1OffScaleDisplay

This property enables or disables the display of values that are out of range by clipping them to fit within the range.

Syntax:

```
setX10ffScaleDisplay(boolean);  
boolean = getX10ffScaleDisplay();
```

where:

boolean

Can be one of the following:

[true](#) displays out-of-range values at scale limit (minimum or maximum). For example, a value of 700 goes beyond the specified range of 250-500, so its marker in a scatter graph would appear at the right edge of the graph. This value is the default.

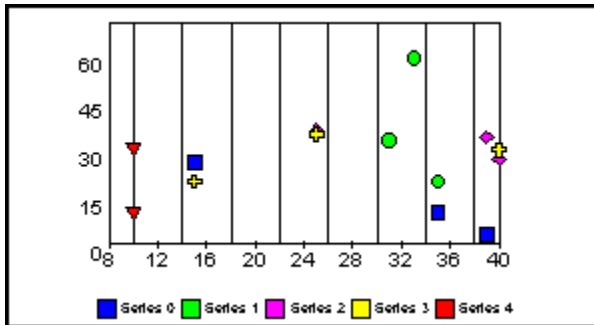
[false](#) only displays values that fall within a given range. In the example above, the 700 marker would be omitted.

Example:

```

setMarkerSizeDefault(60);
  setX1offScaleDisplay(true);
setX1ScaleMaxAuto(false);
setX1ScaleMax(40.0);
setX1MajorGridStyle(4);
setGraphType(61);

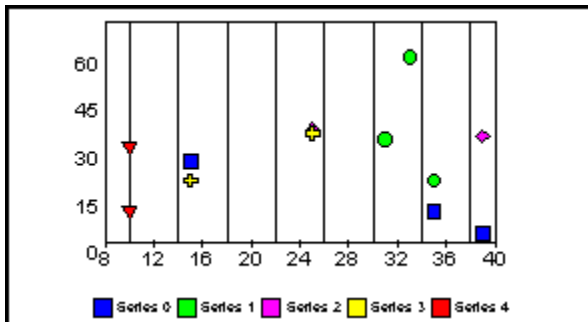
```



```

setX1offScaleDisplay(false);

```



See [getDisplayOffScale\(\)](#) on page 737, [setDisplayOffScale\(\)](#) on page 884.

X1ScaleMax

This property specifies the maximum data value that can be used to image the graph and its axis. When a value is assigned to this property, the X1ScaleMaxAuto property is automatically set to false.

Syntax:

```

setX1ScaleMax(value);
value = getX1ScaleMax();

```

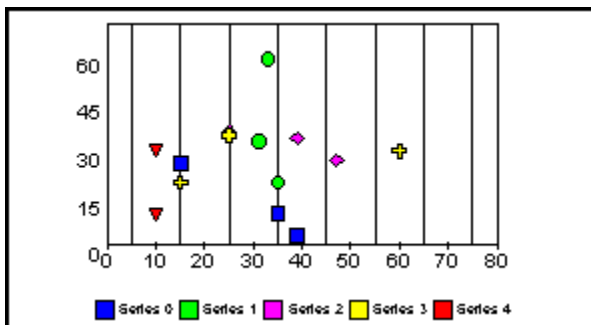
where:

value

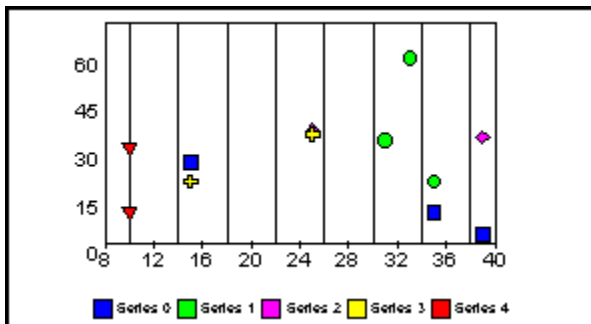
Is any value. This value is a Java double-precision number. 100 is the default value.

Example:

```
setMarkerSizeDefault(60);
setX1ScaleMaxAuto(false);
    setX1ScaleMax(80.0);
setY1MajorGridDisplay(false);
setX1MajorGridStyle(4);
setGraphType(61);
```



`setX1ScaleMax(40.0);`



Note: If `X1ScaleMax` is set to a value less than `X1ScaleMin`, the values will be automatically switched and the `X1AxisDescending` property will be set to true.

See [X1ScaleMaxAuto](#) on page 610, [X1ScaleMin](#) on page 611, [X1ScaleMinAuto](#) on page 613, [getScaleMax\(\)](#) on page 803, [setScaleMax\(\)](#) on page 968, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMaxAuto\(\)](#) on page 969.

X1ScaleMaxAuto

This property enables or disables use of the value assigned to the setX1ScaleMax property.

Syntax:

```
setX1ScaleMaxAuto(boolean);
```

```
boolean = getX1ScaleMaxAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates maximum scale. This value is the default.

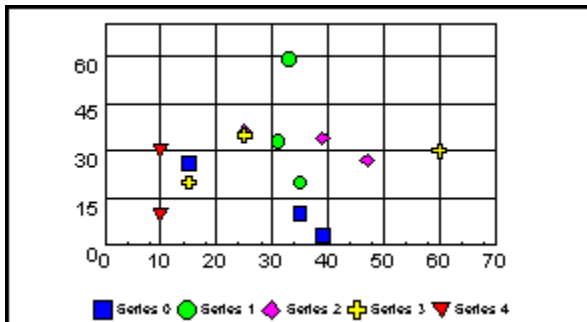
[false](#) uses value set by X1ScaleMax.

Example:

```

setX1MinorGridStyle(2);
setMarkerSizeDefault(60);
setX1ScaleMax(50.0);
setGraphType(61);
setX1MinorGridDisplay(true);
setX1MinorGridStep(4.0);
setX1ScaleMaxAuto(false);
setUseSeriesShapes(true);

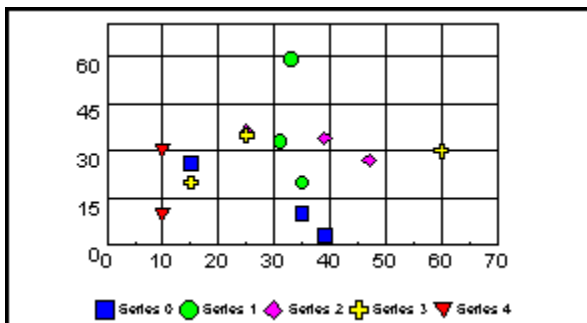
```



```

setX1ScaleMaxAuto(true);

```



See [X1ScaleMax](#) on page 608, [X1ScaleMin](#) on page 611, [X1ScaleMinAuto](#) on page 613, [getScaleMax\(\)](#) on page 803, [setScaleMax\(\)](#) on page 968, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMaxAuto\(\)](#) on page 969, [getX1ScaleMaxAutoValue\(\)](#) on page 827.

X1ScaleMin

This property specifies the minimum data value that can be used to manually image the graph and its axis. When a value is assigned to this property, the `X1ScaleMinAuto` property is automatically set to false.

Syntax:

```
setX1ScaleMin(value);
value = getX1ScaleMin();
```

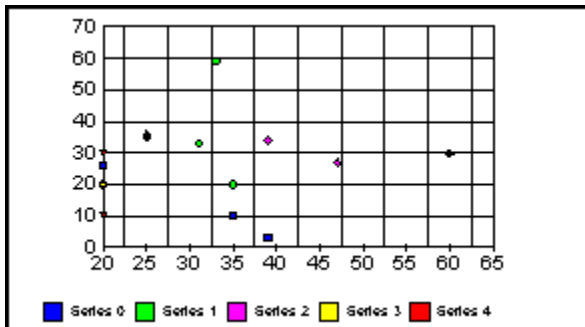
where:

value

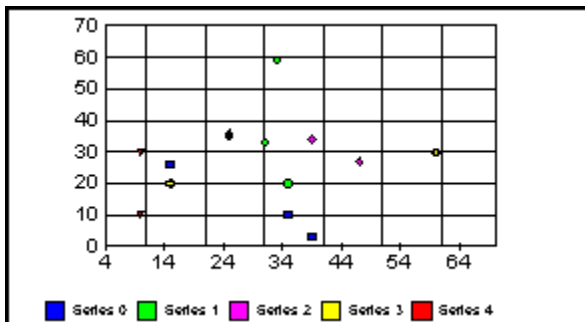
Is any value. This value is a Java double-precision number. 0.0 is the default value.

Example:

```
setGraphType(61);
setX1ScaleMaxAuto(false);
setX1ScaleMin(20.0);
```



```
setX1ScaleMin(4.0);
```



Note: If X1ScaleMax is set to a value less than X1ScaleMin, the values will be automatically switched and the X1AxisDescending property will be set to true.

See [X1ScaleMinAuto](#) on page 613, [X1ScaleMax](#) on page 608, [X1ScaleMaxAuto](#) on page 610, [getScaleMin\(\)](#) on page 805, [setScaleMin\(\)](#) on page 970, [getScaleMinAuto\(\)](#) on page 806, [setScaleMinAuto\(\)](#) on page 971.

X1ScaleMinAuto

This property enables or disables automatic minimum scaling.

Syntax:

```
setX1ScaleMinAuto(boolean);
```

```
boolean = getX1ScaleMinAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates minimum scale. This value is the default.

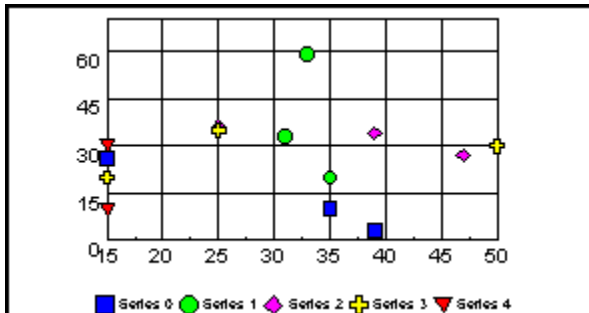
[false](#) uses the value set by X1ScaleMin.

Example:

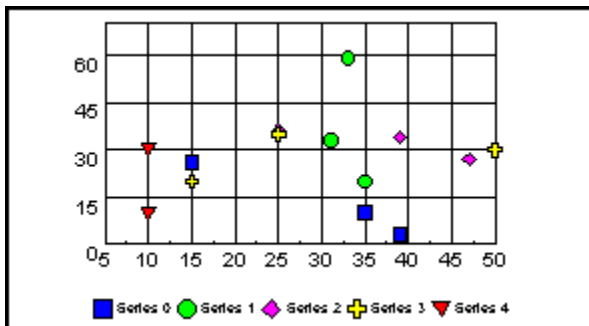
```

setX1MinorGridStyle(2);
setMarkerSizeDefault(60);
setX1ScaleMax(50.0);
    setX1ScaleMinAuto(false);
setGraphType(61);
setX1ScaleMin(15.0);
setX1MinorGridDisplay(true);
setX1MinorGridStep(4.0);
setX1ScaleMaxAuto(false);
setUseSeriesShapes(true);

```



```
setX1ScaleMinAuto(true);
```



See [X1ScaleMin](#) on page 611, [X1ScaleMax](#) on page 608, [X1ScaleMaxAuto](#) on page 610, [getScaleMin\(\)](#) on page 805, [setScaleMin\(\)](#) on page 970, [getScaleMinAuto\(\)](#) on page 806, [setScaleMinAuto\(\)](#) on page 971, [getX1ScaleMinAutoValue\(\)](#) on page 827.

X1TitleDisplay

This property enables or disables drawing of X-axis title string that is defined by the X1Titlestring property.

Syntax:

```
setX1TitleDisplay(boolean);
boolean = getX1TitleDisplay();
```

where:

boolean

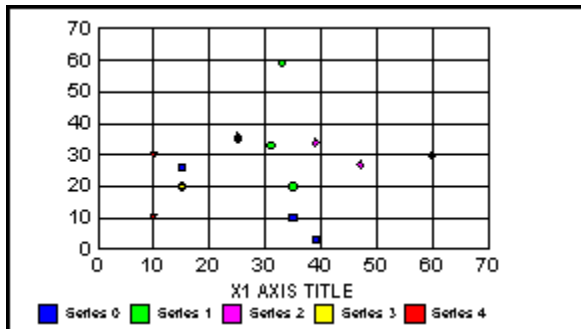
Can be one of the following:

true draws X-axis title string defined by X1TitleString.

false does not draw X-axis title. This value is the default.

Example:

```
setGraphType(61);
setX1TitleString("X1 AXIS TITLE"); setX1TitleDisplay(true);
```



Note: If the X1LabelDisplay property is set to false, the title string will not be drawn regardless of the X1TitleDisplay or X1TitleString property.

See [X1TitleString](#) on page 615.

X1TitleString

This property can be used to define the title string for the X-axis in a graph. When a title string is assigned to this property, the X1TitleDisplay property is automatically set to true.

Syntax:

```
setX1TitleString(string);
string = getX1TitleString();
```

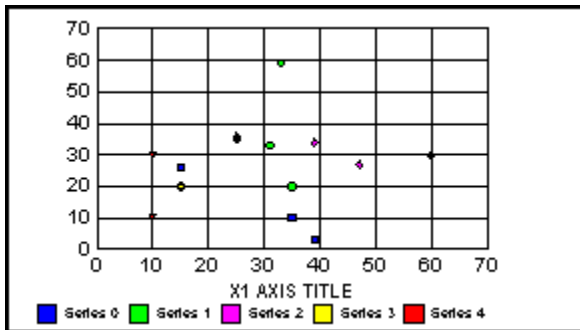
where:

string

Represents any printable characters. Null String is the default value.

Example:

```
setGraphType(61);
  setX1TitleString("X1 AXIS TITLE");
setX1TitleDisplay(true);
```



Note: If the X1LabelDisplay property is set to false, the title string will not be drawn regardless of the X1TitleDisplay or X1Titlestring property.

See [X1TitleDisplay](#) on page 614.

X1ZeroLineDisplay

This property enables or disables drawing of the zero line on the X-axis when zero is within the X1 axis scale range.

Syntax:

```
setX1ZeroLineDisplay(boolean);
boolean = getX1ZeroLineDisplay();
```

where:

boolean

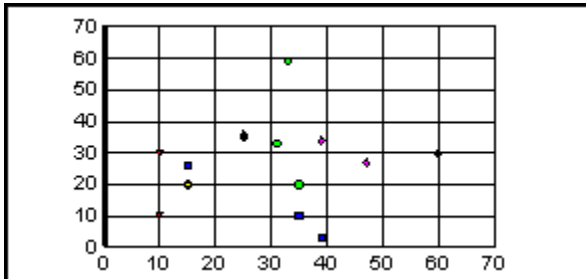
Can be one of the following:

true draws zero line.

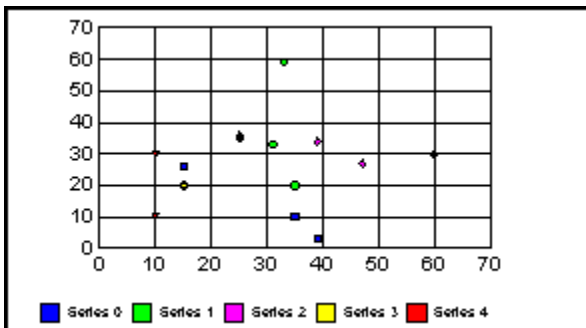
false does not draw zero line. This value is the default.

Example:

```
setGraphType(61);  setX1ZeroLineDisplay(true);
```



```
setX1ZeroLineDisplay(false);
```



See [X1MustIncludeZero](#) on page 605, [X1ScaleMax](#) on page 608, [X1ScaleMin](#) on page 611.

Y1AxisDescending

This property can be used to change the direction of the Y1-axis.

Syntax:

```
setY1AxisDescending(boolean);  
boolean = getY1AxisDescending();
```

where:

boolean

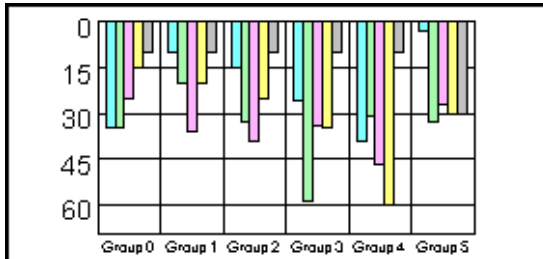
Can be one of the following:

true draws labels on Y1-axis in descending order, from top-to-bottom in a vertical graph or from right-to-left in a horizontal graph. This property also reverses the order of the axis labels and changes all data-related objects (markers) from their previous state.

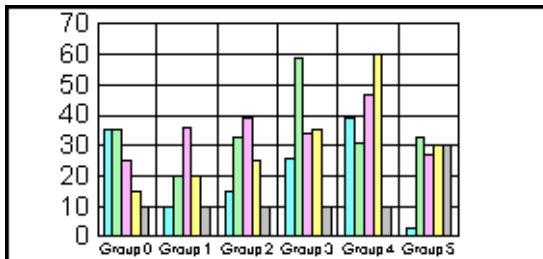
[false](#) draws labels on Y1-axis in ascending order, from bottom-to-top in a vertical graph or from left-to-right in a horizontal graph Y1-axis. This value is the default.

Example:

```
setY1AxisDescending(true);  
setDepthAngle(0);  
setDepthRadius(0);  
setLegendDisplay(false);
```



```
setY1AxisDescending(false);
```



See [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853.

Y1AxisLineDisplay

This property enables or disables display of the Y1 axis base line. Y1AxisSide determines the location where the Y1 axis base line is drawn.

Syntax:

```
setY1AxisLineDisplay(boolean);  
boolean = getY1AxisLineDisplay();
```

where:

boolean

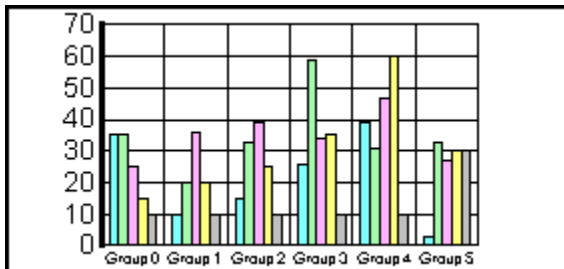
Can be one of the following:

`true` draws the Y1-axis base line. This value is the default.

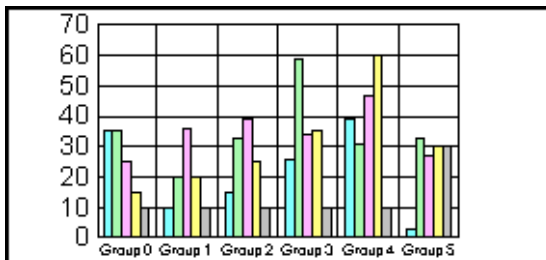
`false` does not draw the Y1-axis base line.

Example:

```
setFillColor(getY1AxisLine(), newColor (0,0,0));
setDepthAngle(0);
    setY1AxisLineDisplay(true);
setLineWidth(getY1AxisLine(),5);
setDepthRadius(0);
setLegendDisplay(false);
```



```
setY1AxisLineDisplay(false);
```



Note: In the default configuration for this graph type, the Y1AxisLine is the same width and location as the bottom of the graph frame. When this line size is used, the setting of this property does not change the appearance of the graph. To make a thicker and more visible axis line, use the `getY1AxisLine()` method to get the object ID of the line and `setLineWidth()` method to increase the width of the line.

See [Y1AxisSide](#) on page 619, [getAxisSide\(\)](#) on page 720, [setLineWidth\(\)](#) on page 937, [setAxisSide\(\)](#) on page 856.

Y1AxisSide

This property controls which side of the graph the Y1-axis will be imaged.

Syntax:

```
setY1AxisSide(value);
value = getY1AxisSide();
```

where:

value

Can be one of the following:

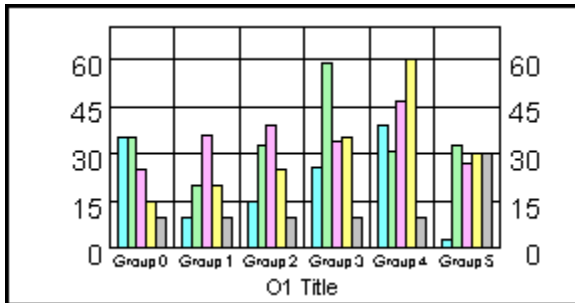
0 sets image to low side of graph (bottom or left). 0 is the default value.

1 sets image to high side of graph (top or right).

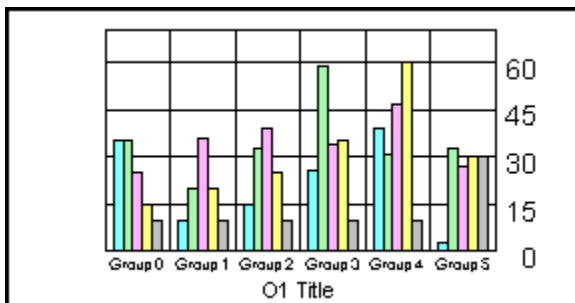
2 sets image to both sides.

Example:

```
setDepthAngle(0);
    setY1AxisSide(2);
setDepthRadius(0);
setLegendDisplay(false);
```



```
setY1AxisSide(1);
```



See [X1AxisLineDisplay](#) on page 576, [setAxisAssignment\(\)](#) on page 850, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856.

Y1ExcludeMaxLabel

This property enables or disables exclusion of the maximum label on the Y1 axis. For example, if the Y1-axis labels include values in the range 0, 10, 20, 30, ... 70 and this property is set to true, the new set of Y1-axis labels will include 0, 10, 20, 30, ... 60.

Syntax:

```
setY1ExcludeMaxLabel(boolean);
```

```
boolean = getY1ExcludeMaxLabel();
```

where:

boolean

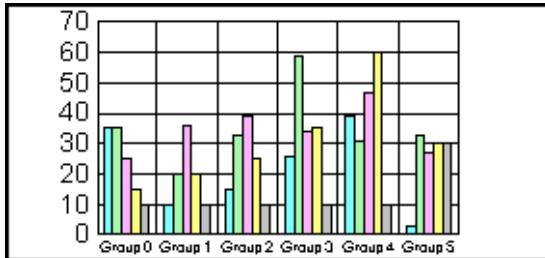
Can be one of the following:

[true](#) excludes the maximum label.

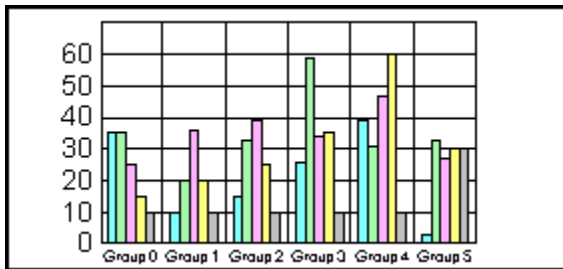
[false](#) includes the maximum label. This value is the default.

Example:

```
setY1ExcludeMaxLabel(false);
setLegendDisplay(false);
setDepthRadius(0);
setDepthAngle(0);
```



```
setY1ExcludeMaxLabel(true);
```



Note: This property will not affect the appearance of the graph if drawing of Y1-axis labels has been disabled with `setY1LabelDisplay(false)`; When `setY1LabelDisplay(true)` is used, the maximum label will be included or excluded as defined by `Y1ExcludeMaxLabel`.

See [Y1ExcludeMinLabel](#) on page 622, [Y1LabelDisplay](#) on page 624, [getExcludeMaxLabel\(\)](#) on page 744, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMaxLabel\(\)](#) on page 888, [setExcludeMinLabel\(\)](#) on page 889.

Y1ExcludeMinLabel

This property enables or disables exclusion of the minimum label on the Y1 axis. For example, if the Y1-axis labels include values in the range 0, 10, 20, 30, ... 70 and this property is set to true, the new set of Y1-axis labels will include 10, 20, 30, ... 70.

Syntax:

```
setY1ExcludeMinLabel(boolean);
```

```
boolean = getY1ExcludeMinLabel();
```

where:

boolean

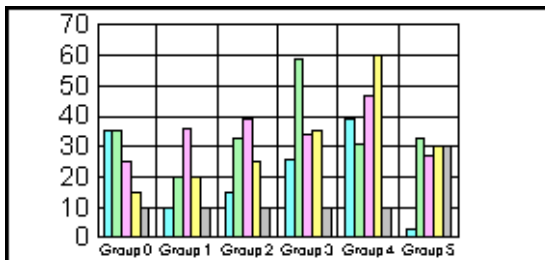
Can be one of the following:

`true` excludes the maximum label.

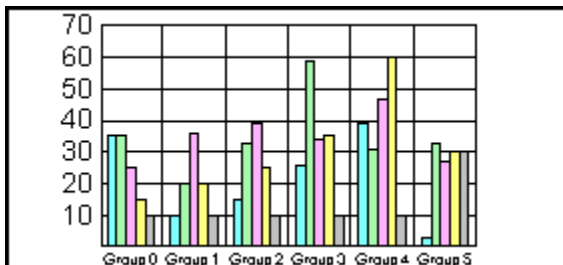
`false` includes the maximum label. This value is the default.

Example:

```
setY1ExcludeMinLabel(false);
setLegendDisplay(false);
setDepthAngle(0);
setDepthRadius(0);
```



```
setY1ExcludeMinLabel(true);
```



Note: This property will not affect the appearance of the graph if drawing of Y1-axis labels has been disabled with `setY1LabelDisplay(false)`; When `setY1LabelDisplay(true)` is used, the minimum label will be included or excluded as defined by `Y1ExcludeMinLabel`.

See [Y1ExcludeMaxLabel](#) on page 621, [Y1LabelDisplay](#) on page 624, [getY1ExcludeMaxLabel\(\)](#) on page 744, [getY1ExcludeMinLabel\(\)](#) on page 745, [setY1ExcludeMaxLabel\(\)](#) on page 888, [setY1ExcludeMinLabel\(\)](#) on page 889.

Y1LabelDisplay

This property enables or disables drawing of Y1-axis labels and the Y1-axis title.

Syntax:

```
setY1LabelDisplay(boolean);
```

```
boolean = getY1LabelDisplay();
```

where:

boolean

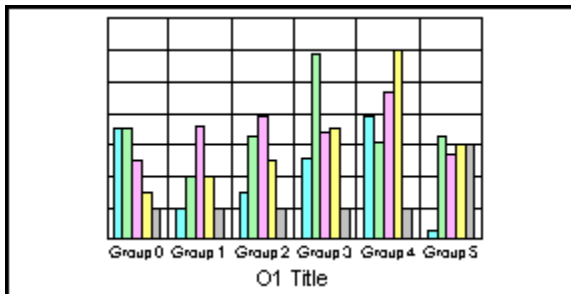
Can be one of the following:

[true](#) draws Y1-axis labels. This value is the default.

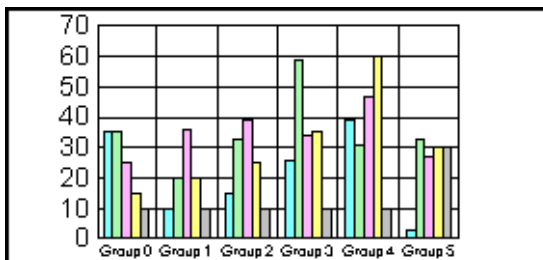
[false](#) does not draw Y1-axis labels.

Example:

```
setY1LabelDisplay(false);  
setDepthAngle(0);  
setLegendDisplay(false);  
setDepthRadius(0);
```



```
setY1LabelDisplay(true);
```



Note: If the Y1 axis title is defined with `Y1TitleString` and displayed with `Y1TitleDisplay`, setting `Y1LabelDisplay` to false will also remove the title string.

See [Y1LabelFormat](#) on page 625, [Y1LabelFormatPattern](#) on page 626, [Y1LabelRotate](#) on page 627, [Y1LabelStagger](#) on page 628.

Y1LabelFormat

When `Y1LabelDisplay` is enabled, this property selects a preset standard format for Y1-axis labels. Set this property to minus one if you want to define a standard Java format pattern string with `Y1LabelFormatPattern`.

Syntax:

```
setY1LabelFormat(value);
```

```
value = getY1LabelFormat();
```

where:

value

Is any integer between -1 and 21.

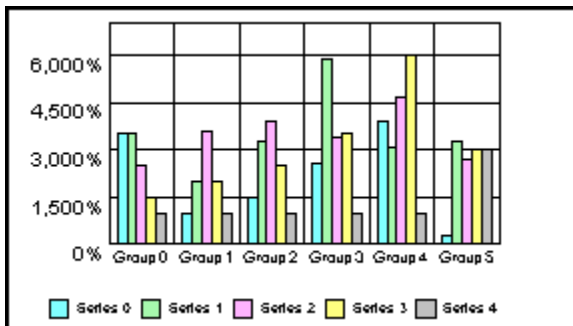
-1 uses the pattern defined by `Y1LabelFormatPattern`.

0 through 21 selects a preset number format. 0 (General) is the default value.

See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setY1LabelFormat(2);
```



See [Y1LabelFormatPattern](#) on page 626.

Y1LabelFormatPattern

When Y1LabelFormat is set to minus one, this property can be used to define a standard Java format pattern for Y1-axis labels.

Syntax:

```
setY1LabelFormatPattern(string);  
string = getY1LabelFormatPattern();
```

where:

string

Is a pattern string in the following format:

```
pattern:= subpattern{;subpattern}subpattern:=  
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFF' -  
specialCharacterssuffix:= '\\u0000'..'\\uFFFF' -  
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

"#.#" is the default value.

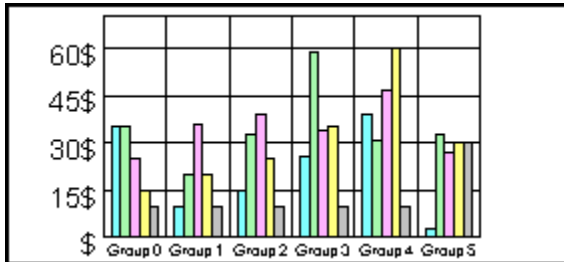
See [Formatting Numbers and Numeric Labels](#) on page 130 for more detailed information about this format pattern.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setY1LabelFormat(-1);  setY1LabelFormatPattern("#.##$");

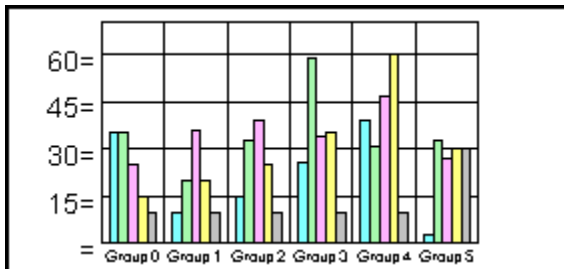
```



```

setY1LabelFormatPattern("#.##=");

```



See [Y1LabelFormat](#) on page 625.

Y1LabelRotate

This property can be used to rotate labels on the Y1-axis.

Syntax:

```

setY1LabelRotate(value);
value = getY1LabelRotate();

```

where:

value

Can be one of the following:

[0](#) for no text rotation. 0 is the default value.

[1](#) to rotate text 90 degrees (reads top to bottom).

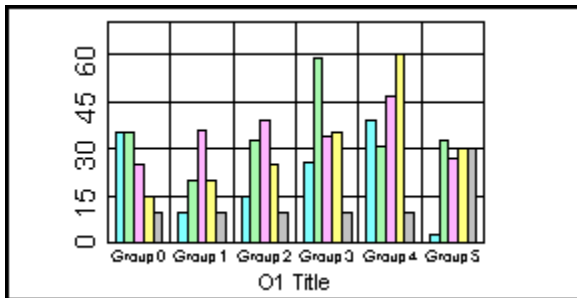
[2](#) to rotate text 270 degrees (reads bottom to top).

Example:

```

setY1LabelRotate(1);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);

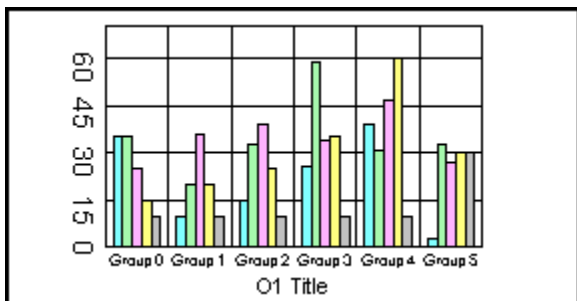
```



```

setY1LabelRotate(2);

```



See [Y1LabelDisplay](#) on page 624, [Y1LabelFormat](#) on page 625, [Y1LabelStagger](#) on page 628.

Y1LabelStagger

This property enables or disables staggered labels on the Y1-axis.

Syntax:

```

setY1LabelStagger(boolean);
boolean = getY1LabelStagger();

```

where:

boolean

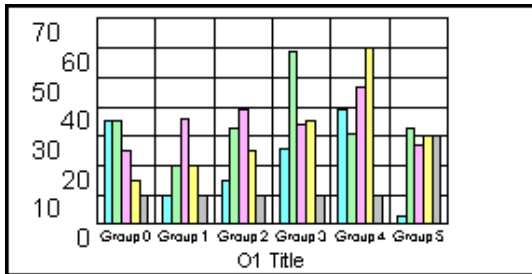
Can be one of the following:

true staggers labels.

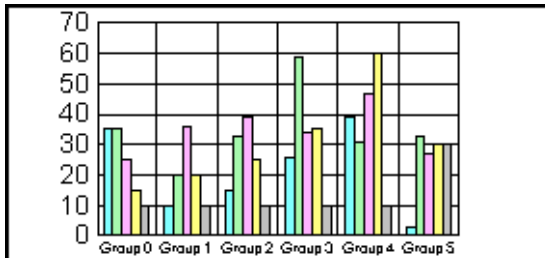
[false](#) draws labels on a single, even line. This value is the default.

Example:

```
setY1LabelStagger(true);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY1LabelStagger(false);
```



Note: This property is not supported in 3D graphs.

See [Y1LabelDisplay](#) on page 624, [Y1LabelFormat](#) on page 625, [Y1LabelRotate](#) on page 627, [getLabelStagger\(\)](#) on page 764, [setLabelStagger\(\)](#) on page 918.

Y1LogScale

This property selects logarithmic or linear scaling for the Y1-axis. When the scale range is set to automatic, any data items with Y1 values less than or equal to zero are graphed as null for log scale. If a manual scale range is specified to include negative values (Y1ScaleMin less than or equal to zero), the request for log scale is ignored and Y1LogScale is automatically set to false (linear scale).

Syntax:

```
setY1LogScale(boolean)
```

```
boolean = getY1LogScale();
```

where:

`boolean`

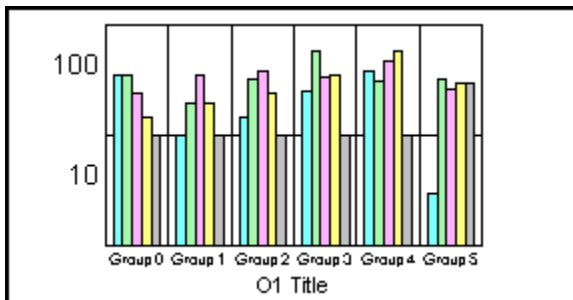
Can be one of the following:

`true` uses logarithmic scale.

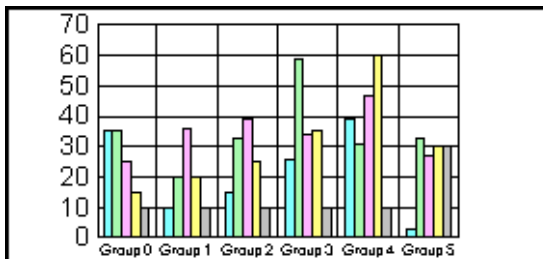
`false` uses linear scale. This value is the default.

Example:

```
setY1LogScale(true);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY1LogScale(false);
```



See [getLogScale\(\)](#) on page 772, [setLogScale\(\)](#) on page 938.

Y1MajorGridDisplay

This property enables or disables the display of major grid lines on the Y1-axis. The Y1-axis can include major and minor grid lines. In the default configuration, major grid lines are displayed and minor grid lines are not displayed.

Syntax:

```
setY1MajorGridDisplay(boolean);
boolean = getY1MajorGridDisplay();
```

where:

boolean

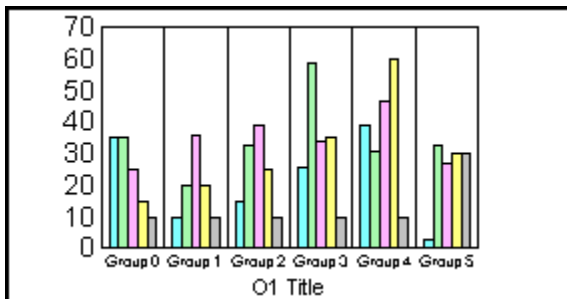
Can be one of the following:

[true](#) draws major grid lines on the Y1-axis. Use the [Y1MajorGridStep](#) property to control the number of grid lines that are drawn. Use the [Y1MajorGridStyle](#) property to control the appearance of major grid lines. This value is the default.

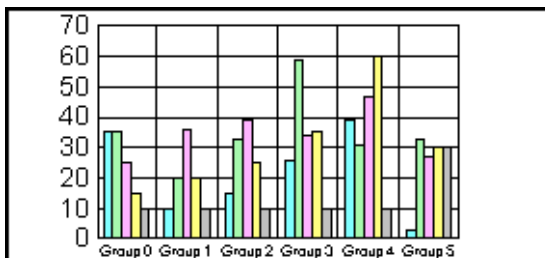
[false](#) does not draw major grid lines on the Y1-axis.

Example:

```
setY1MajorGridDisplay(false);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY1MajorGridDisplay(true);
```



See [Y1MajorGridStyle](#) on page 634, [Y1MinorGridDisplay](#) on page 636, [Y1MinorGridStyle](#) on page 639, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y1MajorGridStep

When Y1MajorGridStepAuto is set to false, this property controls the number of major grid steps on the Y1-axis. Assigning a value to this property will automatically set the Y1MajorGridStepAuto property to false.

Syntax:

```
setY1MajorGridStep(value);
```

```
value = getY1MajorGridStep();
```

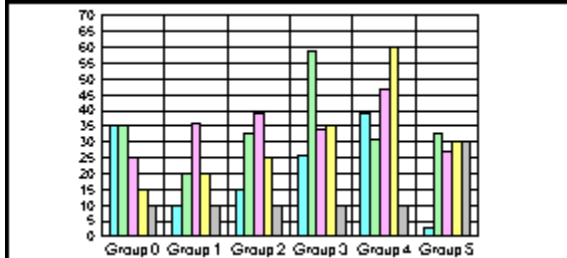
where:

value

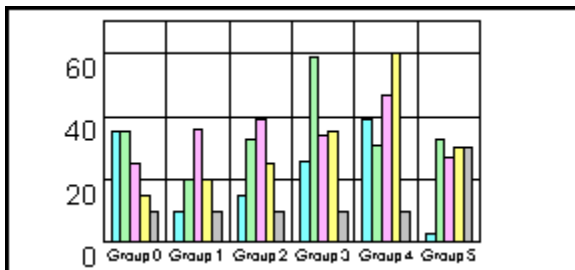
Is any positive value. This value is a Java double-precision number. 15.0 is the default value.

Example:

```
setY1MajorGridStepAuto(false);
setDepthAngle(0);
setY1LabelFormat(1);
setLegendDisplay(false);
    setY1MajorGridStep(5.0);
setDepthRadius(0);
```



```
setY1MajorGridStep(20.0);
```



Note: If the value assigned to Y1MajorGridStep is negative or results in greater than 50 grid lines, the Y1MajorGridStepAuto property is automatically set to true and the Y1MajorGridStep value is ignored.

See [Y1MajorGridStepAuto](#) on page 633, [Y1MinorGridStepAuto](#) on page 638, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y1MajorGridStepAuto

This property enables or disables automatic calculation of the number of major grid lines on the numeric Y1-axis of a graph.

Syntax:

```
setY1MajorGridStepAuto(boolean);
```

```
boolean = getY1MajorGridStepAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates the number of major grid steps on the Y1-axis. The value assigned to Y1MajorGridStep is ignored. This value is the default.

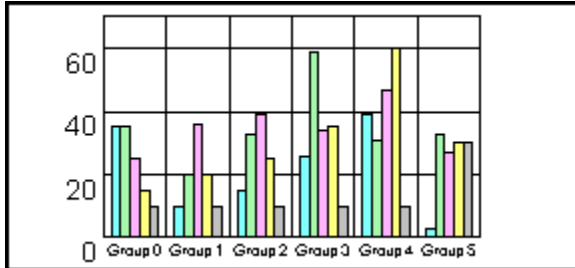
[false](#) uses the value set by Y1MajorGridStep.

Example:

```

setDepthRadius(0);
setDepthAngle(0);
setLegendDisplay(false);
setY1MajorGridStep(20.0);  setY1MajorGridStepAuto(false);

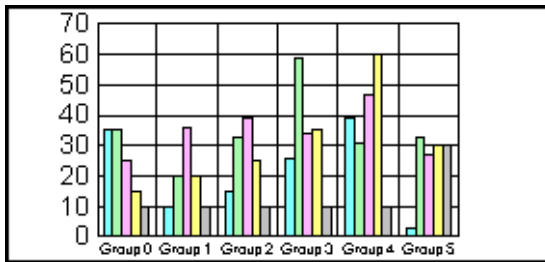
```



```

setY1MajorGridStepAuto(true);

```



Note: If the value assigned to Y1MajorGridStep is negative or results in greater than 50 grid lines, the Y1MajorGridStepAuto property is automatically set to true and the Y1MajorGridStep value is ignored.

See [Y1MajorGridStep](#) on page 632, [Y1MinorGridStepAuto](#) on page 638, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911, [getY1MajorGridStepAutoValue\(\)](#) on page 830.

Y1MajorGridStyle

This property controls the appearance of major grid lines on the numeric Y1-axis. Major grid lines can be small ticks at the base of the axis or lines that extend to the height or width of the graph frame. The Y1-axis can include major and minor grid lines. Major grid lines are normally set to a different style than minor grid lines so that the user can distinguish between the two.

Syntax:

```
value = getY1MajorGridStyle();
```

where:

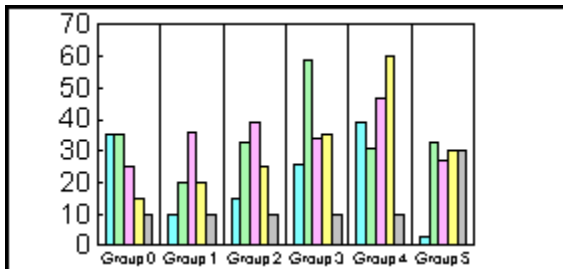
value

Can be one of the following:

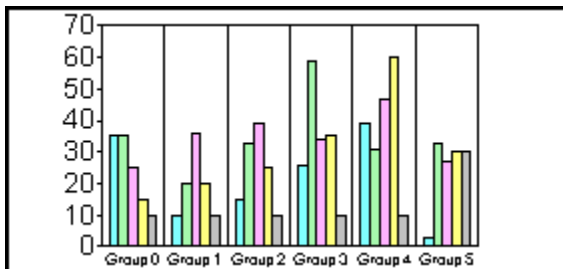
- 0 for normal grid lines, height or width of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height or width of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setY1MajorGridStyle(2);
```



```
setY1MajorGridStyle(3);
```



See [Y1MajorGridDisplay](#) on page 630, [Y1MinorGridDisplay](#) on page 636, [Y1MinorGridStyle](#) on page 639, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y1MajorTickDisplay

This property enables or disables drawing of major tick marks on the Y1-axis.

Syntax:

```
setY1MajorTickDisplay (boolean);  
boolean = getY1MajorTickDisplay ();
```

where:

boolean

Can be one of the following:

[true](#) draws major tick marks.

[false](#) does not draw major tick marks. This value is the default.

Y1MajorTickStyle

When major tick marks are enable on the Y1 axis with the Y1MajorTickDisplay property or setDisplay(getY1MajorTick(), true), this property sets the style of major tick marks on the Y1 axis.

Syntax:

```
setY1MajorTickStyle(value);  
value = getY1MajorTickStyle ();
```

where:

value

Can be one of the following:

[0](#) for small tick marks from frame edge outward. 0 is the default value.

[1](#) for small tick marks from frame edge outward

[2](#) for small tick marks from frame edge inward

[3](#) for small tick marks from frame edge outward

[4](#) for small tick marks span across the frame edge

Y1MinorGridDisplay

This property enables or disables the display of minor grid lines on the Y1-axis. The Y1-axis can include major and minor grid lines. In the default configuration, major grid lines are displayed and minor grid lines are not displayed.

Syntax:

```
setY1MinorGridDisplay(boolean);
boolean = getY1MinorGridDisplay();
```

where:

boolean

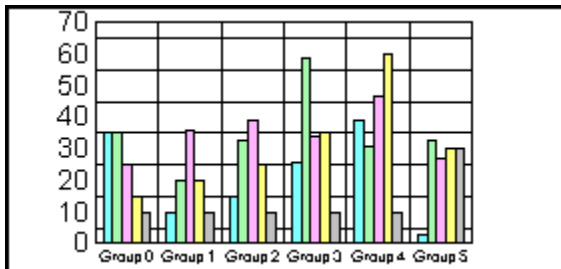
Can be one of the following:

true draws minor grid lines on the Y1-axis. Use the `Y1MinorGridStep` property to control the number of grid lines that are drawn. Use `Y1MinorGridStyle` to control the appearance of minor grid lines.

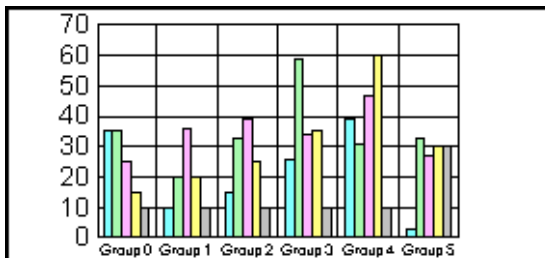
false does not draw minor grid lines on the Y1-axis. This value is the default.

Example:

```
setDepthAngle(0);
setLegendDisplay(false);
setY1MajorGridDisplay(false);
    setY1MinorGridDisplay(true);
setDepthRadius(0);
```



```
setY1MajorGridDisplay(true);  setY1MinorGridDisplay(false);
```



See [Y1MajorGridDisplay](#) on page 630, [Y1MajorGridStyle](#) on page 634, [Y1MinorGridStepAuto](#) on page 638, [Y1MinorGridStyle](#) on page 639, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y1MinorGridStepAuto

This property enables or disables automatic calculation of the number grid steps for Y1-axis minor grid lines.

Syntax:

```
setY1MinorGridStepAuto(boolean);  
boolean = getY1MinorGridStepAuto();
```

where:

boolean

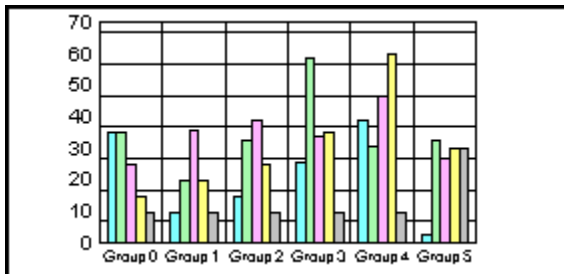
Can be one of the following:

[true](#) automatically calculates the number of minor grid steps on the Y1-axis. This value is the default.

[false](#) uses the value set by Y1MinorGridStep.

Example:

```
setDepthAngle(0);  
setY1MinorGridDisplay(true);  
    setY1MinorGridStepAuto(false);  
setY1MinorGridStep(7);  
setLegendDisplay(false);  
setDepthRadius(0);
```



Note: If Y1MinorGridStep is set to a negative value or a value that results in greater than 50 minor grid lines between each major grid line, the Y1MinorGridStepAuto property is automatically set to true.

See [Y1MajorGridStep](#) on page 632, [Y1MajorGridStepAuto](#) on page 633, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y1MinorGridStyle

This property controls the appearance of minor grid lines on the Y1-axis. Minor grid lines can be small ticks at the base of the axis or lines that extend to the height or width of the graph frame. The Y1-axis can include major and minor grid lines. Minor grid lines are normally set to a different style than major grid lines so that the user can distinguish between the two.

Syntax:

```
setY1MinorGridStyle(value);  
value = getY1MinorGridStyle();
```

where:

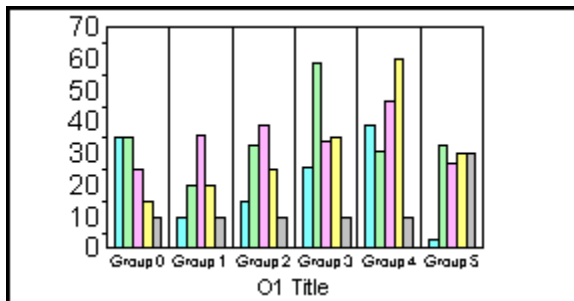
value

Can be one of the following:

- 0 for normal grid lines, height or width of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height or width of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```
setDepthAngle(0);  
setY1MajorGridDisplay(false);  
setY1MinorGridDisplay(true);  
setLegendDisplay(false);  
    setY1MinorGridStyle(3);  
setDepthRadius(0);
```



See [Y1MajorGridDisplay](#) on page 630, [Y1MajorGridStyle](#) on page 634, [Y1MinorGridDisplay](#) on page 636, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y1MinorTickDisplay

This property enables or disables drawing of minor tick marks on the Y-axis.

Syntax:

```
setY1MinorTickDisplay(boolean);  
boolean = getY1MinorTickDisplay ();
```

where:

boolean

Can be one of the following:

[true](#) draws minor tick marks.

[false](#) does not draw minor tick marks. This value is the default.

Y1MinorTickStepAuto

This property enables or disables automatic calculation of minor tick marks on the Y1 axis. When this property is enabled, the Y1MinorTickStep property is ignored.

Syntax:

```
setY1MinorTickStepAuto (boolean);  
boolean = getY1MinorTickStepAuto ();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates the number of tick steps on the X-axis. This value is the default.

[false](#) uses the value set by Y1MinorTickStep.

Y1MinorTickStyle

When minor tick marks are enable on the Y1 axis with the Y1MinorTickDisplay property or setDisplay(getY1MinorTick(), true), this property sets the style of minor tick marks on the Y1 axis.

Syntax:

```
setY1MinorTickStyle(value);  
value = getY1MinorTickStyle ();
```

where:

value

Can be one of the following:

- 0 for small tick marks from frame edge outward
- 1 for small tick marks from frame edge outward
- 2 for small tick marks from frame edge inward
- 3 for small tick marks from frame edge outward
- 4 for small tick marks spanning across the frame edge. This value is the default.

Y1MustIncludeZero

When automatic scaling is used (Y1ScaleMaxAuto and Y1ScaleMinAuto are true) and the range of values provided to the graph includes zero, this property determines whether or not the graph must include zero on the Y1 axis. If the logarithmic scale is enabled with Y1LogScale(true) or a manual scale range excludes zero (Y1ScaleMax and Y1ScaleMin are greater than zero), this property is ignored.

Syntax:

```
setY1MustIncludeZero(boolean);
boolean = getY1MustIncludeZero();
```

where:

boolean

Can be one of the following:

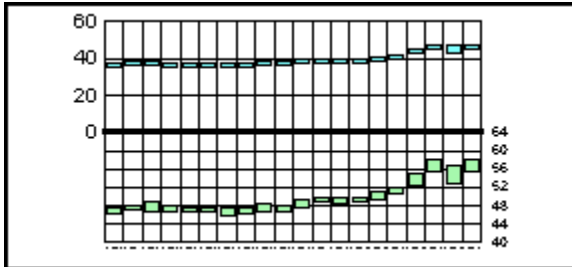
- true includes zero. This value is the default.
- false excludes zero if it is not within the given automatic scale range.

Example:

```

setDepthAngle(0);
    setY1MustIncludeZero(true);
setGraphType(21);
setDepthRadius(0);
getContentPane().add(perspective1);

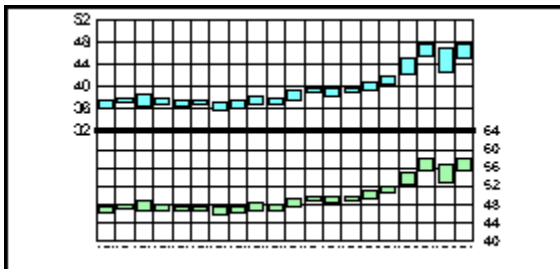
```



```

setDepthAngle(0);
    setY1MustIncludeZero(false);
setGraphType(21);
setDepthRadius(0);
getContentPane().add(perspective1);

```



See [Y1ScaleMax](#) on page 643, [Y1ScaleMin](#) on page 646, [Y1ZeroLineDisplay](#) on page 651.

Y1OffScaleDisplay

This property enables or disables the display of values that are out of range by clipping them to fit within the range.

Syntax:

```

setY1OffScaleDisplay(boolean);
boolean = getY1OffScaleDisplay();

```

where:

boolean

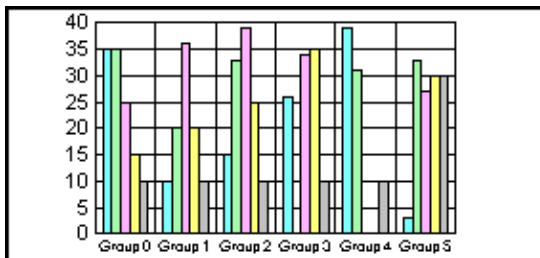
Can be one of the following:

[true](#) draws off scale values. Values less than or greater than a given range are represented at the scale limit (minimum or maximum). For example, a value of 700 goes beyond the specified range of 250-500, so its bar would go all the way to the top of the graph. This value is the default.

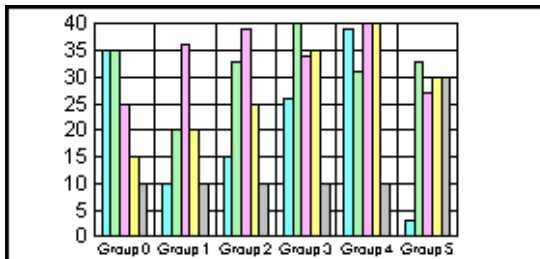
[false](#) does not draw off scale values. Only values that fall directly between a given range are displayed. In the example above, the 700 bar would be omitted.

Example:

```
setScaleMaxAuto(getY1Axis(), false);
setY1ScaleMax(40.0);
setY1OffScaleDisplay(false);
```



```
setY1OffScaleDisplay(true);
```



See [getDisplayOffScale\(\)](#) on page 737, [setDisplayOffScale\(\)](#) on page 884.

Y1ScaleMax

This property specifies the maximum data value that can be used to image the graph and its axis. When a value is assigned to this property, the Y1ScaleMaxAuto property must be set to false.

Syntax:

```
setY1ScaleMax(value);
value = getY1ScaleMax();
```

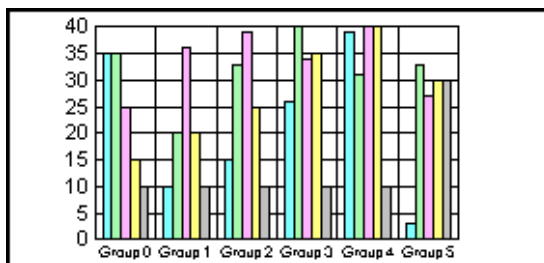
where:

value

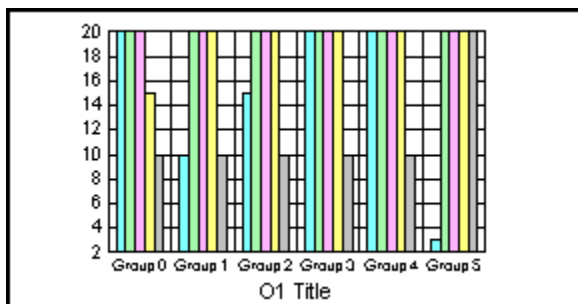
Is any value. This value is a Java double-precision number. 70.0 is the default value.

Example:

```
setY1ScaleMaxAuto(false);
setDepthAngle(0);
    setY1ScaleMax(40.0);
setLegendDisplay(false);
setY1OffScaleDisplay(true);
setDepthRadius(0);
```



```
setY1ScaleMax(20.0);
```



Note: If Y1ScaleMax is set to a value less than Y1ScaleMin, the values will be automatically switched and the Y1AxisDescending property will be set to true.

See [Y1ScaleMaxAuto](#) on page 645, [Y1ScaleMin](#) on page 646, [Y1ScaleMinAuto](#) on page 648, [getScaleMax\(\)](#) on page 803, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969.

Y1ScaleMaxAuto

This property enables or disables use of the value assigned to the `setY1ScaleMax` property.

Syntax:

```
setY1ScaleMaxAuto(boolean);
```

```
boolean = getY1ScaleMaxAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates the maximum value on the Y1-axis. This value is the default.

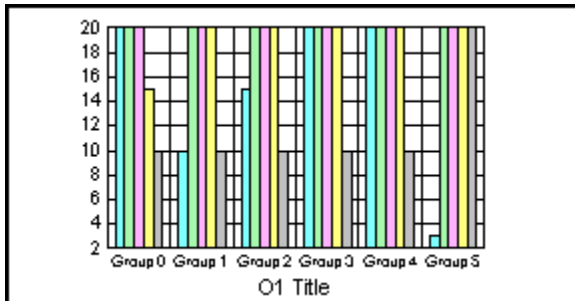
[false](#) uses value set by `Y1ScaleMax`.

Example:

```

setY1ScaleMaxAuto(false);
setDepthAngle(0);
setY1ScaleMax(20.0);
setLegendDisplay(false);
setY1OffScaleDisplay(true);
setDepthRadius(0);

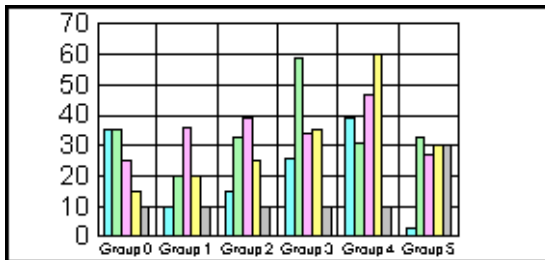
```



```

setY1ScaleMaxAuto(true);

```



See [Y1ScaleMax](#) on page 643, [Y1ScaleMin](#) on page 646, [Y1ScaleMinAuto](#) on page 648, [getScaleMax\(\)](#) on page 803, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969, [getY1ScaleMaxAutoValue\(\)](#) on page 831.

Y1ScaleMin

This property specifies the minimum data value that can be used to manually image the graph and its axis. When a value is assigned to this property, the Y1ScaleMinAuto property must be set to false.

Syntax:

```

setY1ScaleMin(value);
value = getY1ScaleMin();

```

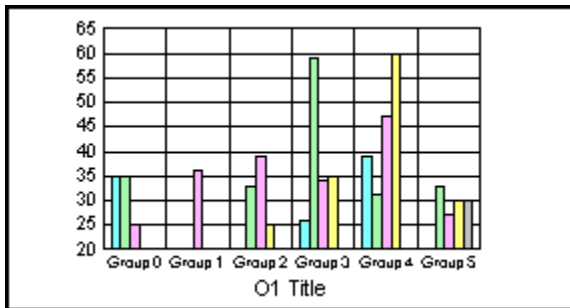
where:

value

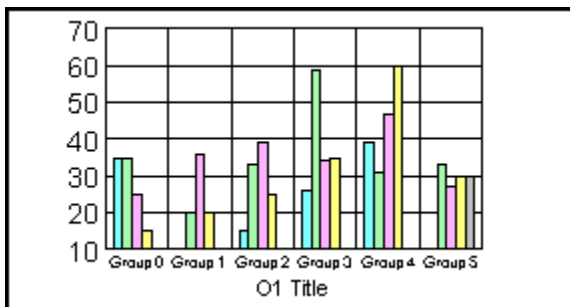
Is any value. This value is a Java double-precision number. 0.0 is the default value.

Example:

```
setY1ScaleMinAuto(false);
setDepthAngle(0);
    setY1ScaleMin(20.0);
setLegendDisplay(false);
setY1OffScaleDisplay(true);
setDepthRadius(0);
```



`setY1ScaleMin(10.0);`



Note: If Y1ScaleMax is set to a value less than Y1ScaleMin, the values will be automatically switched and the Y1AxisDescending property will be set to true.

See [Y1ScaleMinAuto](#) on page 648, [Y1ScaleMax](#) on page 643, [Y1ScaleMaxAuto](#) on page 645, [getScaleMin\(\)](#) on page 805, [getScaleMinAuto\(\)](#) on page 806, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971.

Y1ScaleMinAuto

This property enables or disables automatic minimum scaling.

Syntax:

```
setY1ScaleMinAuto(boolean);  
boolean = getY1ScaleMinAuto();
```

where:

boolean

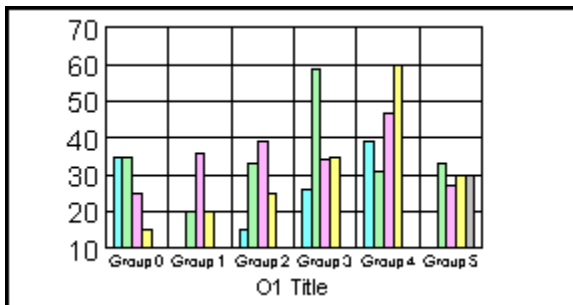
Can be one of the following:

[true](#) automatically calculates the minimum value on the Y1-axis. The value assigned to the Y1ScaleMin property is ignored. This value is the default.

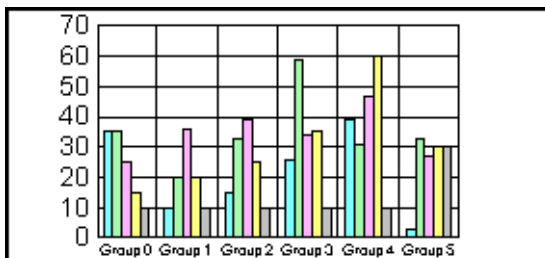
[false](#) uses value set by Y1ScaleMin.

Example:

```
setDepthAngle(0);  
    setY1ScaleMinAuto(false);  
setLegendDisplay(false);  
setY1ScaleMin(10.0);  
setDepthRadius(0);
```



```
setY1ScaleMinAuto(true);
```



See [Y1ScaleMin](#) on page 646, [Y1ScaleMax](#) on page 643, [Y1ScaleMaxAuto](#) on page 645, [getScaleMin\(\)](#) on page 805, [getScaleMinAuto\(\)](#) on page 806, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971, [getY1ScaleMinAutoValue\(\)](#) on page 832.

Y1TitleDisplay

This property enables or disables drawing of the Y1-axis title string. The Y1-axis title is defined by the Y1Titlestring property.

Syntax:

```
setY1TitleDisplay(boolean);  
boolean = getY1TitleDisplay();
```

where:

boolean

Can be one of the following:

[true](#) draws Y1-axis title.

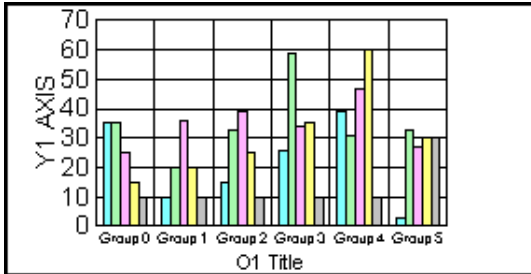
[false](#) does not draw Y1-axis title. This value is the default.

Example:

```

setDepthAngle(0);
    setY1TitleDisplay(true);
setLegendDisplay(false);
setY1TitleString("Y1 AXIS");
setDepthRadius(0);

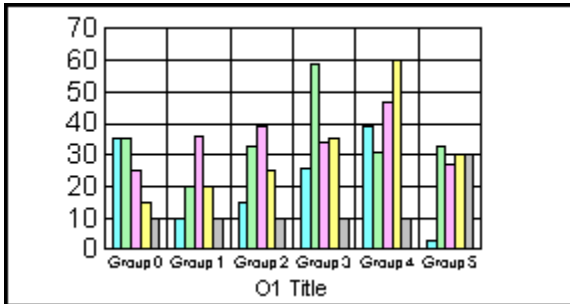
```



```

setY1TitleDisplay(false);

```



Note: If Y1LabelDisplay is set to false, the title string will not be drawn regardless of the Y1TitleDisplay or Y1TitleString property.

See [Y1LabelDisplay](#) on page 624, [Y1TitleString](#) on page 650.

Y1TitleString

This property defines the Y1-axis title string. When a title string is assigned to this property, the Y1TitleDisplay property is automatically set to true.

Syntax:

```

setY1TitleString("string");
string = getY1TitleString();

```

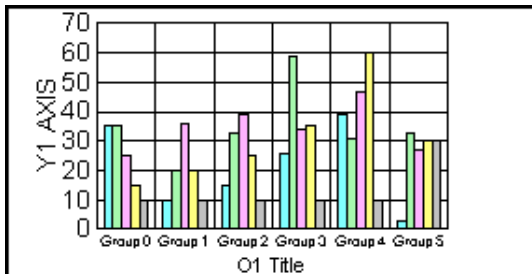
where:

string

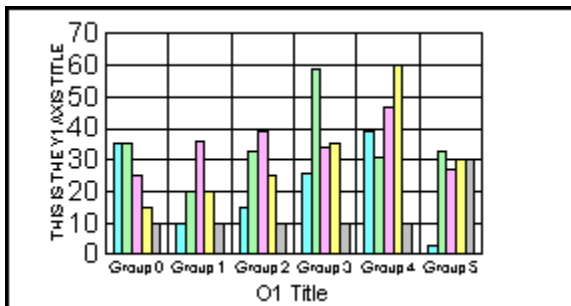
Represents any printable characters. Null String is the default value.

Example:

```
setDepthAngle(0);
setY1TitleDisplay(true);
setLegendDisplay(false);
    setY1TitleString("Y1 AXIS");
setDepthRadius(0);
```



```
setY1Titlestring("THIS IS THE Y1 AXIS TITLE");
```



Note: If Y1LabelDisplay is set to false, the title string will not be drawn regardless of the Y1TitleDisplay or Y1Titlestring property.

See [Y1LabelDisplay](#) on page 624, [Y1TitleDisplay](#) on page 649.

Y1ZeroLineDisplay

This property enables or disables drawing of the zero line on the Y1-axis when zero is within the Y1 axis scale range.

Syntax:

```
setY1ZeroLineDisplay(boolean);
boolean = getY1ZeroLineDisplay();
```

where:

boolean

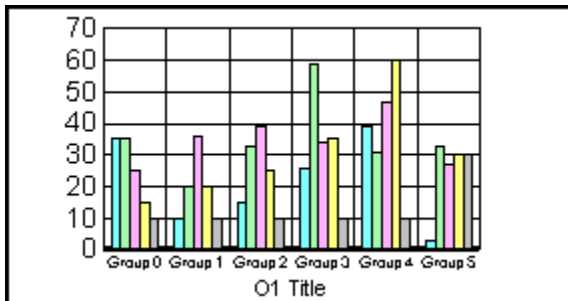
Can be one of the following:

true draw zero line on Y1-axis.

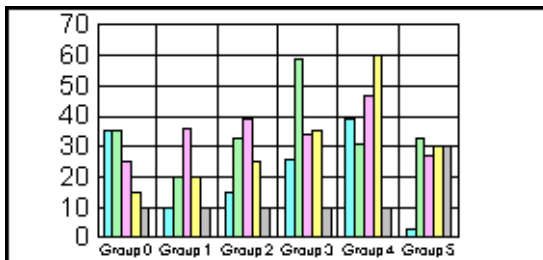
false does not draw zero line. This value is the default.

Example:

```
setDepthAngle(0);
  setY1ZeroLineDisplay(true);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY1ZeroLineDisplay(false);
```



Note: To increase the width of the zero line, use the `getY1ZeroLine()` method to get the object ID of the line and the `setLineWidth()` method to specify the width (in pixels) of the line.

See [Y1MustIncludeZero](#) on page 641, [Y1ScaleMax](#) on page 643, [Y1ScaleMin](#) on page 646, [setLineWidth\(\)](#) on page 937.

Y2AxisDescending

This property changes the direction of the Y2-axis.

Syntax:

```
setY2AxisDescending(boolean);
```

```
boolean = getY2AxisDescending();
```

where:

boolean

Can be one of the following:

[true](#) for descending order, drawn from top-to-bottom in a vertical graph or from right-to-left in a horizontal graph. This property also reverses the order of the axis labels and changes all data-related objects (markers) from their previous state.

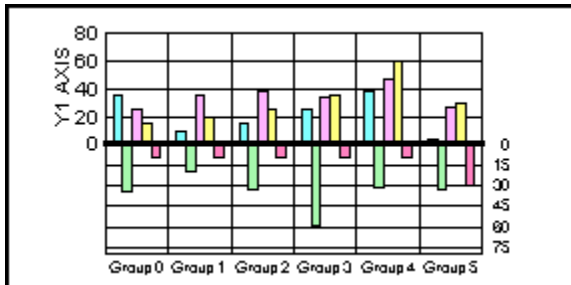
[false](#) for ascending order, drawn from bottom-to-top in a vertical graph or from left-to-right in a horizontal graph. This value is the default.

Example:

```

setY2AxisDescending(true);
setDepthAngle(0);
setGraphType(21);
setY2AxisLineDisplay(false);
setDepthRadius(0);
setLegendDisplay(false);

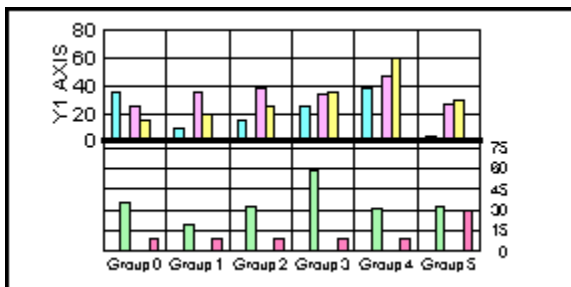
```



```

setY2AxisDescending(false);

```



See [setAxisDescending\(\)](#) on page 853, [getAxisDescending\(\)](#) on page 717.

Y2AxisLineDisplay

This property enables or disables display of the Y2 axis base line. The Y2AxisSide property determines the location where the Y2 axis base line is drawn.

Syntax:

```

setY2AxisLineDisplay(boolean);
boolean = getY2AxisLineDisplay();

```

where:

boolean

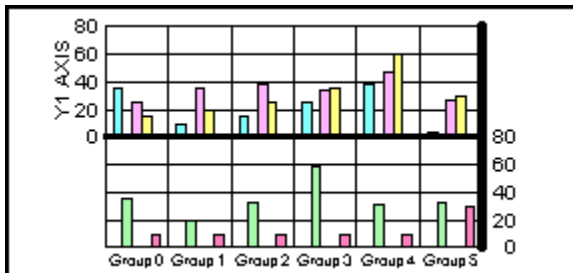
Can be one of the following:

`true` draws the Y2-axis base line. This value is the default.

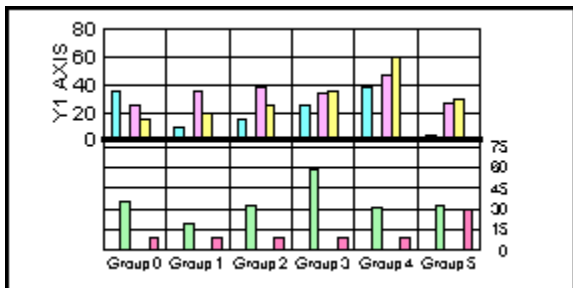
`false` does not draw the Y2-axis base line.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(21);
  setY2AxisLineDisplay(true);
setLineWidth(getY2AxisLine(),8);
```



```
setY2AxisLineDisplay(false);
```



Note: In the default configuration for this graph type, the Y2AxisLine is the same width and location as the bottom of the graph frame. When this line size is used, the setting of this property does not change the appearance of the graph. To make a thicker and more visible axis line, use the `getY2AxisLine()` method to get the object ID of the line and `setLineWidth()` method to increase the width of the line.

See [Y2AxisSide](#) on page 655, [setLineWidth\(\)](#) on page 937, [getAxisAssignment\(\)](#) on page 716, [setAxisAssignment\(\)](#) on page 850, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856.

Y2AxisSide

This property controls which side of the graph the Y2-axis will be imaged.

Syntax:

```
setY2AxisSide(value);
value = getY2AxisSide();
```

where:

value

Can be one of the following:

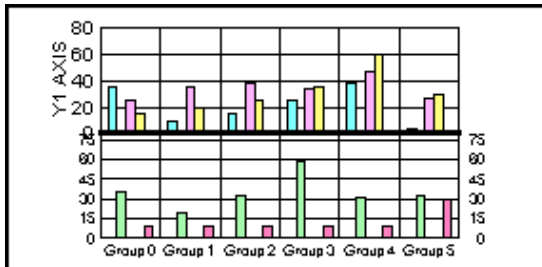
0 images to low side of graph (bottom or left).

1 images to high side of graph (top or right). 1 is the default value.

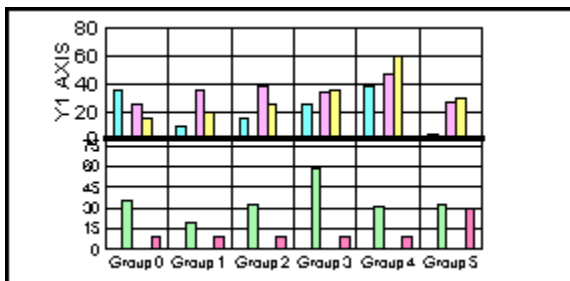
2 images to both sides.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
    setY2AxisSide(2);
```



```
setY2AxisSide(0);
```



See [Y2AxisLineDisplay](#) on page 654, [setAxisAssignment\(\)](#) on page 850, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856.

Y2ExcludeMaxLabel

This property enables or disables exclusion of the maximum label on the Y2 axis. For example, if the Y2-axis labels include values in the range 0, 15, 30, 45, 60, 75 and this property is set to true, the new set of Y2-axis labels will include 0, 15, 30, 45, 60.

Syntax:

```
setY2ExcludeMaxLabel(boolean);  
boolean = getY2ExcludeMaxLabel();
```

where:

boolean

Can be one of the following:

[true](#) excludes maximum label.

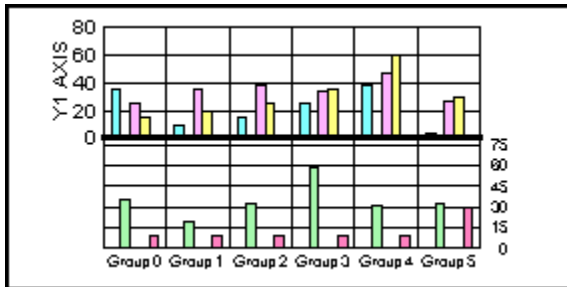
[false](#) include maximum label. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
    setY2ExcludeMaxLabel(false);

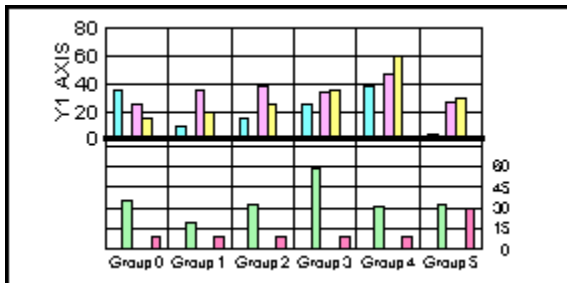
```



```

setY2ExcludeMaxLabel(true);

```



Note: This property will not affect the appearance of the graph if drawing of Y2-axis labels has been disabled with `setY2LabelDisplay(false)`; When `setY2LabelDisplay(true)` is used, the maximum label will be included or excluded as defined by `Y2ExcludeMaxLabel`.

See [Y2ExcludeMinLabel](#) on page 658, [Y2LabelDisplay](#) on page 660, [getExcludeMaxLabel\(\)](#) on page 744, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMaxLabel\(\)](#) on page 888, [setExcludeMinLabel\(\)](#) on page 889.

Y2ExcludeMinLabel

This property enables or disables exclusion of the minimum label on the Y2 axis. For example, if the Y2-axis labels include values in the range 0, 15, 30, 45, 60, 75 and this property is set to true, the new set of Y2-axis labels will include 15, 30, 45, 60, 75.

Syntax:

```
setY2ExcludeMinLabel(boolean);
boolean = getY2ExcludeMinLabel();
```

where:

boolean

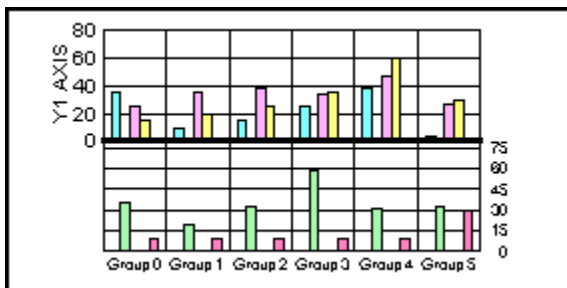
Can be one of the following:

`true` excludes minimum label.

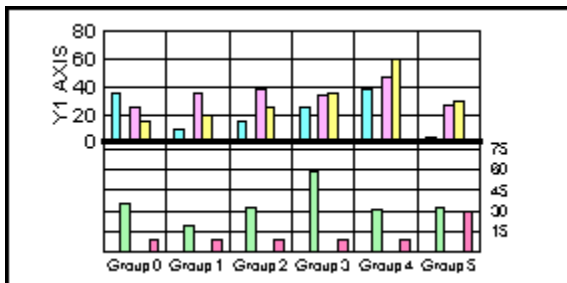
`false` includes minimum label. This value is the default.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
setY2ExcludeMinLabel(false);
```



```
setY2ExcludeMinLabel(true);
```



Note: This property will not affect the appearance of the graph if drawing of Y2-axis labels has been disabled with `setY2LabelDisplay(false)`; When `setY2LabelDisplay(true)` is used, the minimum label will be included or excluded as defined by `Y2ExcludeMinLabel`.

See [Y2ExcludeMaxLabel](#) on page 657, [Y2LabelDisplay](#) on page 660, [getExcludeMaxLabel\(\)](#) on page 744, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMaxLabel\(\)](#) on page 888, [setExcludeMinLabel\(\)](#) on page 889.

Y2LabelDisplay

This property enables or disables drawing of Y2-axis labels and the Y2-axis title.

Syntax:

```
setY2LabelDisplay(boolean);  
boolean = getY2LabelDisplay();
```

where:

boolean

Can be one of the following:

[true](#) draws Y2-axis labels. This value is the default.

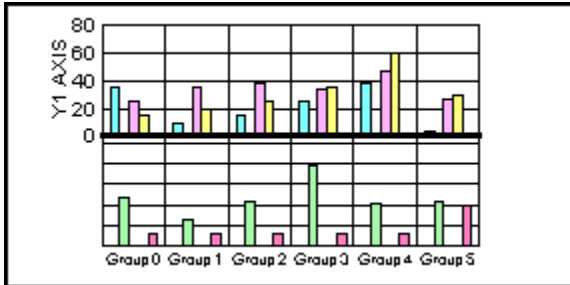
[false](#) does not draw Y2-axis labels.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
    setY2LabelDisplay(false);

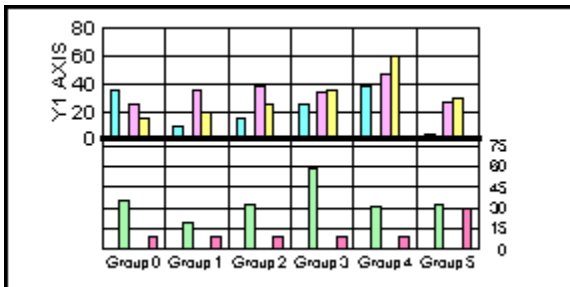
```



```

setY2LabelDisplay(true);

```



Note: If the Y2 axis title is defined with `Y2TitleString` and displayed with `Y2TitleDisplay`, setting `Y2LabelDisplay` to false will also remove the title string.

See [Y2LabelFormat](#) on page 661, [Y2LabelFormatPattern](#) on page 662, [Y2LabelRotate](#) on page 663, [Y2LabelStagger](#) on page 664.

Y2LabelFormat

When `Y2LabelDisplay` is enabled, this property selects a preset standard format for Y2-axis labels. Set this property to minus one if you want to define a standard Java format pattern string with `Y2LabelFormatPattern`.

Syntax:

```

setY2LabelFormat(value);

```

```
value = getY2LabelFormat();
```

where:

value

Is any integer between -1 and 21.

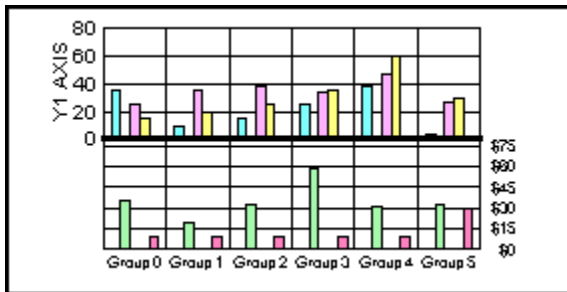
-1 uses the pattern defined by Y2LabelFormatPattern.

0 through 21 selects a preset number format. 0 (General) is the default value.

See [Formatting Numbers and Numeric Labels](#) on page 130.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
setY2LabelFormat(6);
```



See [Y2LabelFormatPattern](#) on page 662.

Y2LabelFormatPattern

When Y2LabelFormat is set to minus one, this property can be used to define a standard Java format pattern for Y2-axis labels.

Syntax:

```
setY2LabelFormatPattern(string);
string = getY2LabelFormatPattern();
```

where:

string

Is a pattern string in the following format:

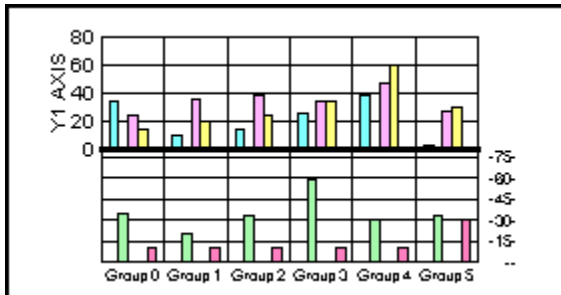
```
pattern:= subpattern{;subpattern}subpattern:=
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFF' -
specialCharacterssuffix:= '\\u0000'..'\\uFFFF' -
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

"#.#" is the default value.

See [Formatting Numbers and Numeric Labels](#) on page 130 for more information about this format pattern.

Example:

```
setGraphType(21);
setY2LabelFormat(-1);
  setY2LabelFormatPattern("-#.#-");
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



See [Y2LabelFormat](#) on page 661.

Y2LabelRotate

This property can be used to rotate labels on the Y2-axis.

Syntax:

```
setY2LabelRotate(value);
```

where:

value

Can be one of the following:

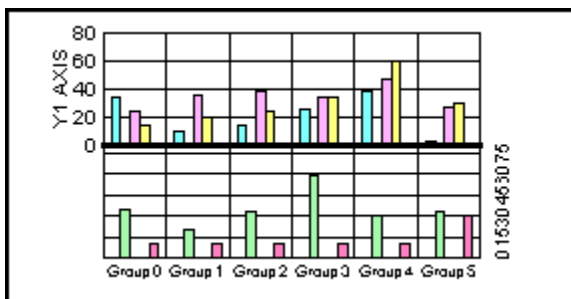
0 for no text rotation. 0 is the default value.

1 rotates text 90 degrees (reads top to bottom)

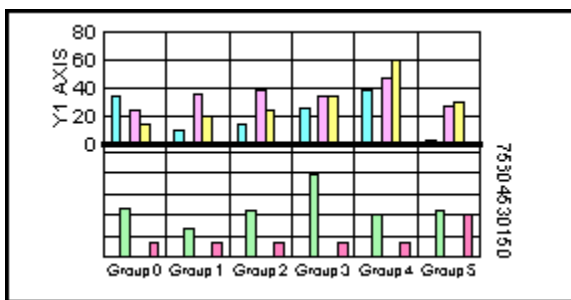
2 rotates text 270 degrees (reads bottom to top)

Example:

```
setGraphType(21);
    setY2LabelRotate(1);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY2LabelRotate(2);
```



See [Y2LabelDisplay](#) on page 660, [Y2LabelFormat](#) on page 661.

Y2LabelStagger

This property enables or disables staggered labels on the Y2-axis.

Syntax:

```
setY2LabelStagger(boolean);
boolean = getY2LabelStagger();
```

where:

boolean

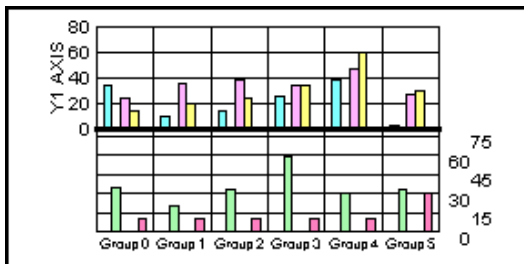
Can be one of the following:

true staggers labels.

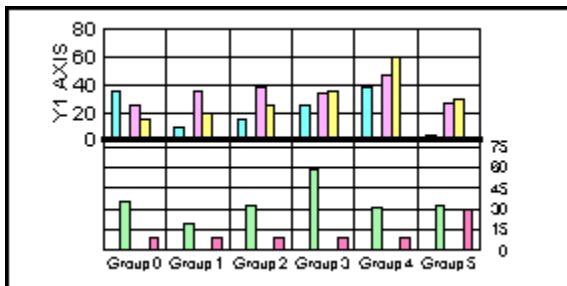
false draws labels on a single, even line. This value is the default.

Example:

```
setGraphType(21);
  setY2LabelStagger(true);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



`setY2LabelStagger(false);`



Note: This property is not supported in 3D graphs.

See [Y2LabelDisplay](#) on page 660, [Y2LabelFormat](#) on page 661, [Y2LabelRotate](#) on page 663, [getLabelStagger\(\)](#) on page 764, [setLabelStagger\(\)](#) on page 918.

Y2LogScale

This property selects logarithmic or linear scaling for the Y2-axis. When the scale range is set to automatic, any data items with Y2 values less than or equal to zero are graphed as null for log scale. If a manual scale range is specified to include negative values (Y2ScaleMin less than or equal to zero), the request for log scale is ignored and Y2LogScale is automatically set to false (linear scale).

Syntax:

```
setY2LogScale(boolean)
```

```
boolean = getY2LogScale();
```

where:

boolean

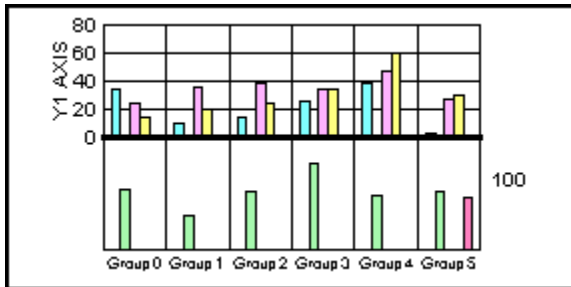
Can be one of the following:

[true](#) uses logarithmic scale.

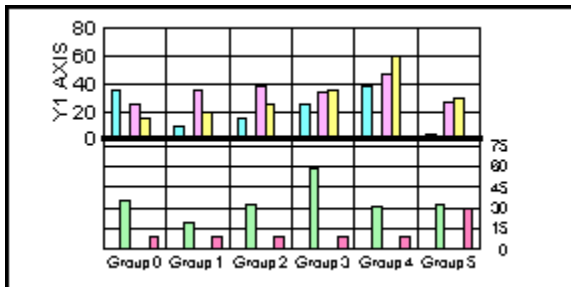
[false](#) uses linear scale. This value is the default.

Example:

```
setGraphType(21);
    setY2LogScale(true);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY2LogScale(false);
```



See [getLogScale\(\)](#) on page 772, [setLogScale\(\)](#) on page 938.

Y2MajorGridDisplay

This property enables or disables the display of major grid lines on the Y2-axis. The Y2-axis can include major and minor grid lines.

Syntax:

```
setY2MajorGridDisplay(boolean);
boolean = getY2MajorGridDisplay();
```

where:

boolean

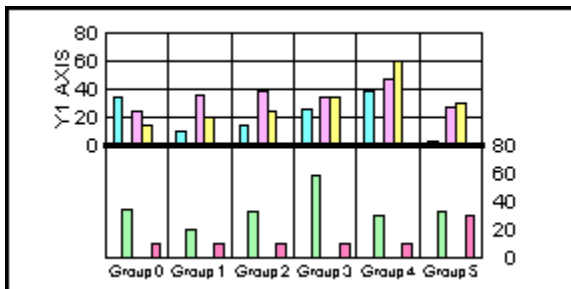
Can be one of the following:

`false` does not draw major grid lines on the Y2-axis.

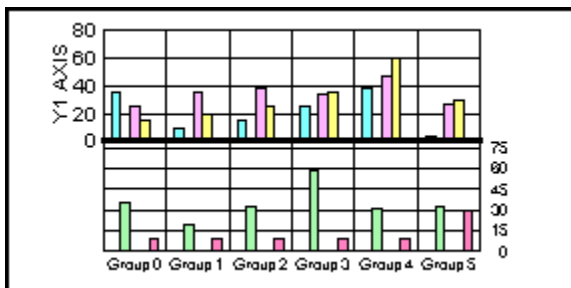
`true` draws major grid lines on the Y2-axis. Use the `Y2MajorGridStep` property to control the number of grid lines that are drawn. Use the `Y2MajorGridStyle` property to control the appearance of major grid lines. This value is the default.

Example:

```
setGraphType(21);
  setY2MajorGridDisplay(false);
setDepthAngle(0);
setLegendDisplay(false);
setDepthRadius(0);
```



```
setY2MajorGridDisplay(true);
```



See [Y2MajorGridStyle](#) on page 671, [Y2MinorGridDisplay](#) on page 673, [Y2MinorGridStyle](#) on page 677, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y2MajorGridStep

When `Y2MajorGridStepAuto` is set to `false`, this property controls the number of major grid steps on the Y2-axis. Assigning a value to this property will automatically set the `Y2MajorGridStepAuto` property to `false`.

Syntax:

```
setY2MajorGridStep(value);
value = getY2MajorGridStep();
```

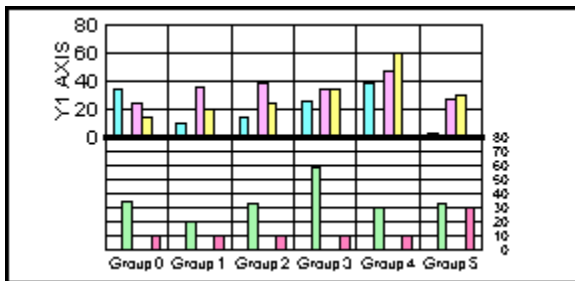
where:

value

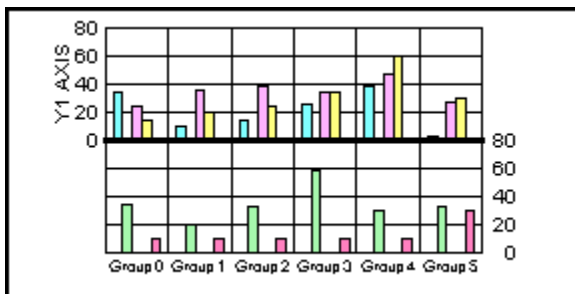
Is any positive value and is a Java double-precision number. 15.0 is the default value.

Example:

```
setGraphType(21);
setY2MajorGridStepAuto(false);
setDepthAngle(0);
setLegendDisplay(false);
    setY2MajorGridStep(10.0);
setDepthRadius(0);
```



```
setY2MajorGridStep(20.0);
```



Note: If the value assigned to Y2MajorGridStep is negative or results in greater than 50 grid lines, the Y2MajorGridStepAuto property is automatically set to true and the Y2MajorGridStep value is ignored.

See [Y2MajorGridStepAuto](#) on page 670, [Y2MinorGridStep](#) on page 675, [Y2MinorGridStepAuto](#) on page 676, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y2MajorGridStepAuto

This property enables or disables automatic calculation of the number of major grid lines on the numeric Y2-axis of a graph.

Syntax:

```
setY2MajorGridStepAuto(boolean);
```

```
boolean = getY2MajorGridStepAuto();
```

where:

boolean

Can be one of the following:

[true](#) automatically calculates the number of major grid steps on the Y2-axis. The value assigned to Y2MajorGridStep is ignored.

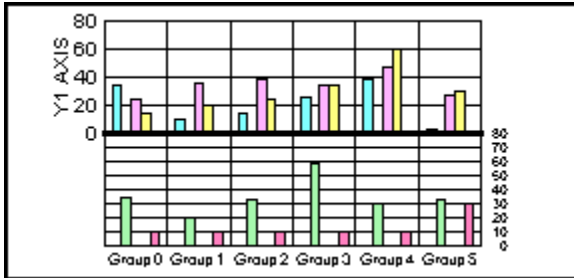
[false](#) uses the value set by Y2MajorGridStep. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
setY2MajorGridStep(10.0);  setY2MajorGridStepAuto(false);

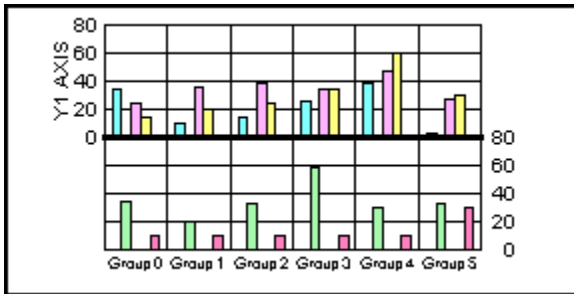
```



```

setY2MajorGridStepAuto(true);

```



Note: If the value assigned to Y2MajorGridStep is negative or results in greater than 50 grid lines, the Y2MajorGridStepAuto property is automatically set to true and the Y2MajorGridStep value is ignored.

See [Y2MajorGridStep](#) on page 668, [Y2MinorGridStep](#) on page 675, [Y2MinorGridStepAuto](#) on page 676, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y2MajorGridStyle

This property controls the appearance of major grid lines on the numeric Y2-axis. Major grid lines can be small ticks at the base of the axis or lines that extend to the height or width of the graph frame. The Y2-axis can include major and minor grid lines. Major grid lines are normally set to a different style than minor grid lines so that the user can distinguish between the two.

Syntax:

```
setY2MajorGridStyle(value);  
value = getY2MajorGridStyle();
```

where:

value

Can be one of the following:

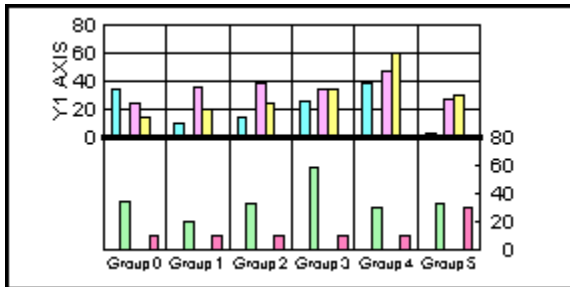
- 0 for normal grid lines, height or width of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height or width of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
    setY2MajorGridStyle(2);

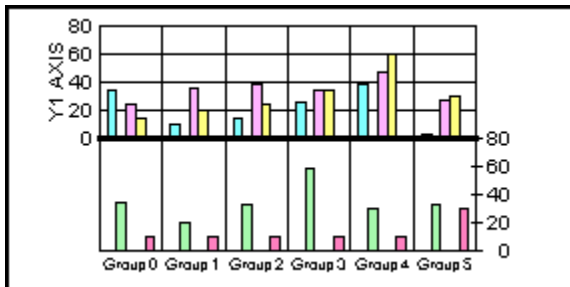
```



```

setY2MajorGridStyle(4);

```



See [Y2MajorGridDisplay](#) on page 667, [Y2MinorGridDisplay](#) on page 673, [Y2MinorGridStyle](#) on page 677, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y2MinorGridDisplay

This property enables or disables the display of minor grid lines on the Y2-axis. The Y2-axis can include major and minor grid lines.

Syntax:

```

setY2MinorGridDisplay(boolean);
boolean = getY2MinorGridDisplay();

```

where:

boolean

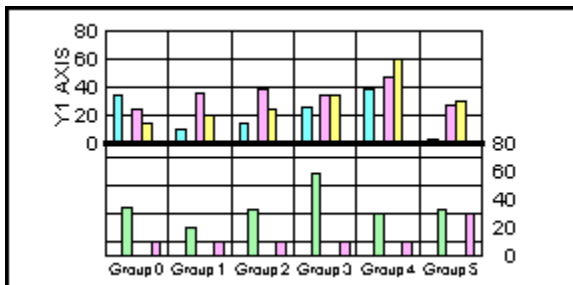
Can be one of the following:

`true` draws minor grid lines on the Y2-axis. Use the `Y2MinorGridStep` property to control the number of grid lines that are drawn. Use `Y2MinorGridStyle` to control the appearance of minor grid lines.

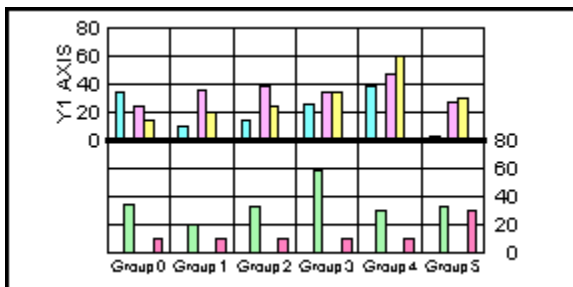
`false` does not draw minor grid lines on the Y2-axis. This value is the default.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
setY2MajorGridDisplay(false); setY2MinorGridDisplay(true);
```



```
setY2MajorGridDisplay(true); setY2MinorGridDisplay(false);
```



See [Y2MajorGridDisplay](#) on page 667, [Y2MajorGridStyle](#) on page 671, [Y2MinorGridStep](#) on page 675, [Y2MinorGridStepAuto](#) on page 676, [Y2MinorGridStyle](#) on page 677, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y2MinorGridStep

This property sets a number of minor grid steps on the Y2-axis of the graph. When they are displayed, minor grid lines are drawn between the major grid lines and the location of the minor gridlines depend very much on the setting for the major grid lines. When a value is assigned to this property, the Y2MinorGridStepAuto property is automatically set to false. If the value assigned to this property results in greater than 50 minor grid lines between each major grid line, the Y2MinorGridStepAuto property is automatically set to true.

Syntax:

```
setY2MinorGridStep(value);
```

```
value = getY2MinorGridStep();
```

where:

value

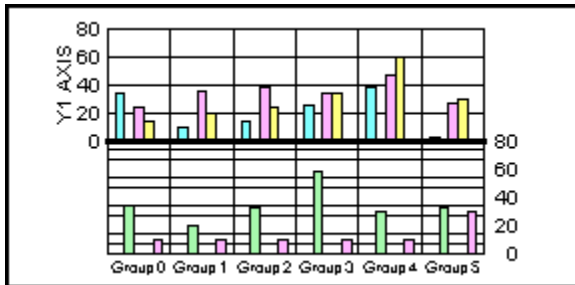
Is any positive value. This value is a Java double-precision number. 15.0 is the default value.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setGraphType(21);
setLegendDisplay(false);
setY2MinorGridStepAuto(false);  setY2MinorGridStep(7.0);

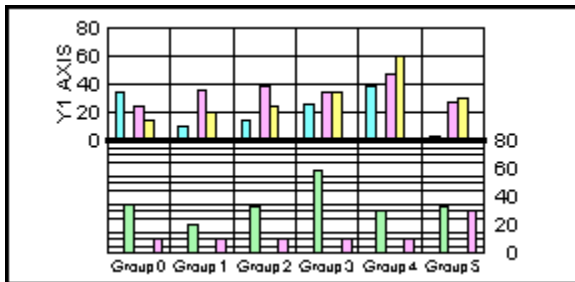
```



```

setY2MinorGridStep(5.0);

```



Note: If Y2MinorGridStep is set to a negative value or a value resulting in greater than 50 minor grid lines between each major grid line, Y2MinorGridStepAuto is set to true.

See [Y2MinorGridStepAuto](#) on page 676, [Y2MajorGridStep](#) on page 668, [Y2MajorGridStepAuto](#) on page 670, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y2MinorGridStepAuto

This property enables or disables automatic calculation of the number grid steps for Y2-axis minor grid lines.

Syntax:

```

setY2MinorGridStepAuto(boolean);

```



```
boolean = getY2MinorGridStepAuto();
```

where:

boolean

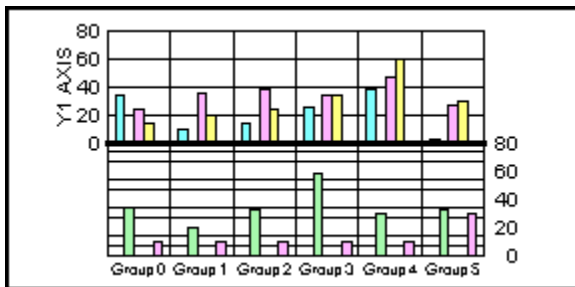
Can be one of the following:

[true](#) automatically calculates the number of minor grid steps on the Y2-axis. This value is the default.

[false](#) uses the value set by Y2MinorGridStep.

Example:

```
setDepthAngle(0);
setGraphType(21);
setY2MinorGridDisplay(true);
    setY2MinorGridStepAuto(false);
setY2MinorGridStep(7);
setLegendDisplay(false);
setDepthRadius(0);
```



Note: If Y2MinorGridStep is set to a negative value or a value that results in greater than 50 minor grid lines between each major grid line, the Y2MinorGridStepAuto property is automatically set to true.

See [Y2MinorGridStep](#) on page 675, [Y2MajorGridStep](#) on page 668, [Y2MajorGridStepAuto](#) on page 670, [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStep\(\)](#) on page 910, [setGridStepAuto\(\)](#) on page 911.

Y2MinorGridStyle

This property controls the appearance of minor grid lines on the Y2-axis. Minor grid lines can be small ticks at the base of the axis or lines that extend to the height or width of the graph frame. The Y2-axis can include major and minor grid lines. Minor grid lines are normally set to a different style than major grid lines so that the user can distinguish between the two.

Syntax:

```
setY2MinorGridStyle(value);
value = getY2MinorGridStyle();
```

where:

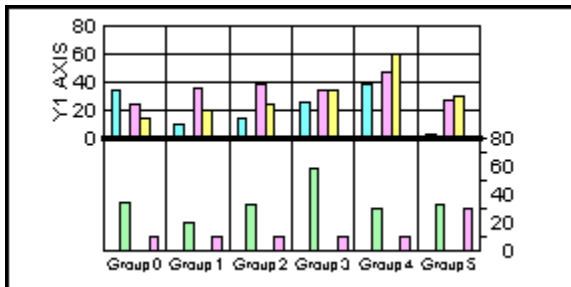
value

Can be one of the following:

- 0 for normal grid lines, height or width of frame. 0 is the default value.
- 1 for normal grid lines extending beyond the height or width of frame.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks spanning across the frame edge.

Example:

```
setGraphType(21);
setDepthAngle(0);
setY2MajorGridDisplay(false);
setY2MinorGridDisplay(true);
setLegendDisplay(false);
    setY2MinorGridStyle(3);
setDepthRadius(0);
```



See [Y2MinorGridDisplay](#) on page 673, [Y2MajorGridStyle](#) on page 671, [Y2MinorGridDisplay](#) on page 673, [getGridStyle\(\)](#) on page 760, [setGridStyle\(\)](#) on page 913.

Y2MustIncludeZero

When automatic scaling is used (Y2ScaleMaxAuto and Y2ScaleMinAuto are true) and the range of values provided to the graph includes zero, this property determines whether or not the graph must include zero on the Y2 axis. If the logarithmic scale is enabled with Y2LogScale(true) or a manual scale range excludes zero (Y2ScaleMax and Y2ScaleMin are greater than zero), this property is ignored.

Syntax:

```
setY2MustIncludeZero(boolean);
```

```
boolean = getY2MustIncludeZero();
```

where:

boolean

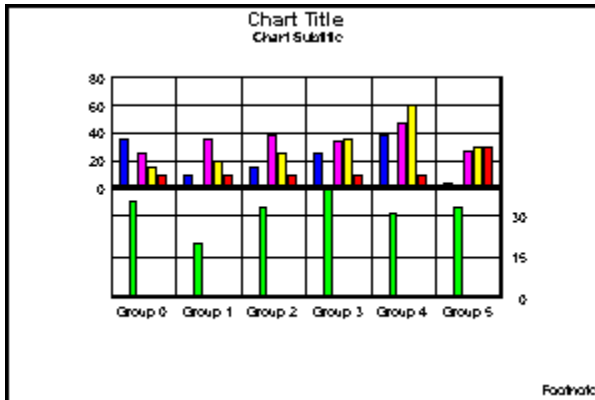
Can be one of the following:

[true](#) includes zero.

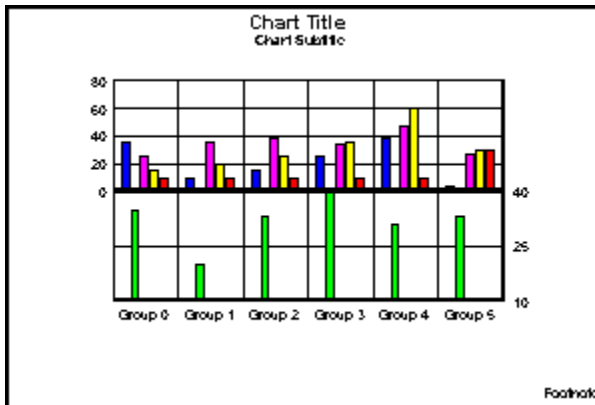
[false](#) excludes zero if it is not within the given automatic scale range. This value is the default.

Example:

```
setY2MustIncludeZero(true);
setGraphType(21);
```



```
setY2MustIncludeZero(false);
setGraphType(21);
```



See [Y2ScaleMax](#) on page 682, [Y2ScaleMin](#) on page 685, [Y2ZeroLineDisplay](#) on page 690.

Y2OffScaleDisplay

This property enables or disables the display of values that are out of range by clipping them to fit within the range. When this property is disabled (false), only values that fall directly between a given range are displayed.

Syntax:

```
setY2OffScaleDisplay(boolean);
```

```
boolean = getY2offScaleDisplay();
```

where:

`boolean`

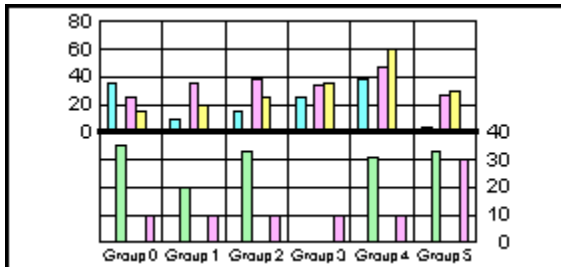
Can be one of the following:

`true` represents values less than or greater than a given range at the scale limit (minimum or maximum). For example, a value of 700 goes beyond the specified range of 250-500, so its bar would go all the way to the top of the graph. This value is the default.

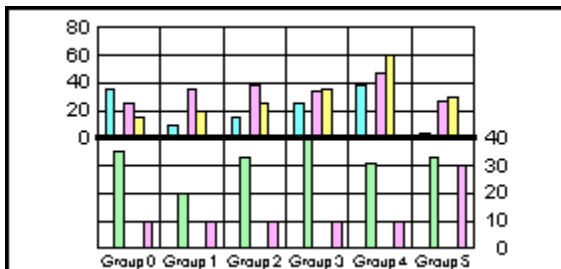
`false` does not draw off scale values.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setLegendDisplay(false);
setGraphType(21);
setY2ScaleMaxAuto(false);
setY2ScaleMax(40.0);
setY2offScaleDisplay(false);
```



```
setY2offScaleDisplay(true);
```



See [getDisplayOffScale\(\)](#) on page 737, [setDisplayOffScale\(\)](#) on page 884.

Y2ScaleMax

This property specifies the maximum data value that can be used to image the graph and its axis. When a value is assigned to this property, the Y2ScaleMaxAuto property is automatically set to false.

Syntax:

```
setY2ScaleMax(value);  
value = getY2ScaleMax();
```

where:

value

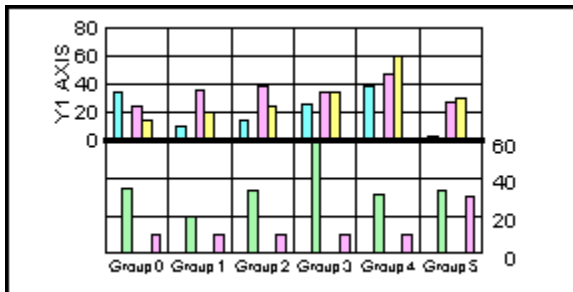
Is any value. This value is a Java double-precision number. 80.0 is the default value.

Example:

```

setGraphType(21);
setY2ScaleMaxAuto(false);
setDepthAngle(0);
    setY2ScaleMaxAuto(false);
    setY2ScaleMax(60.0);
setLegendDisplay(false);
setY2OffScaleDisplay(true);
setDepthRadius(0);

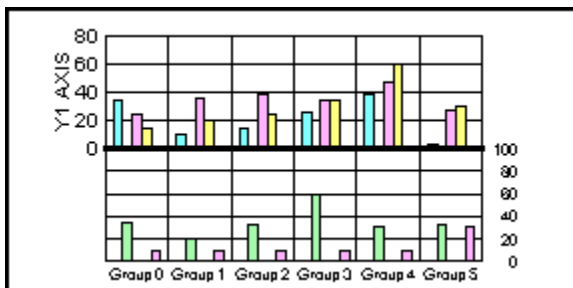
```



```

setY2ScaleMax(100.0);

```



Note: If Y2ScaleMax is set to a value less than Y2ScaleMin, the values will be automatically switched and Y2AxisDescending will be set to true.

See [Y2ScaleMaxAuto](#) on page 683, [Y2ScaleMin](#) on page 685, [Y2ScaleMinAuto](#) on page 686, [getScaleMax\(\)](#) on page 803, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969.

Y2ScaleMaxAuto

This property enables or disables use of the value assigned to the setY2ScaleMax property.

Syntax:

```
setY2ScaleMaxAuto(boolean);
boolean = getY2ScaleMaxAuto();
```

where:

boolean

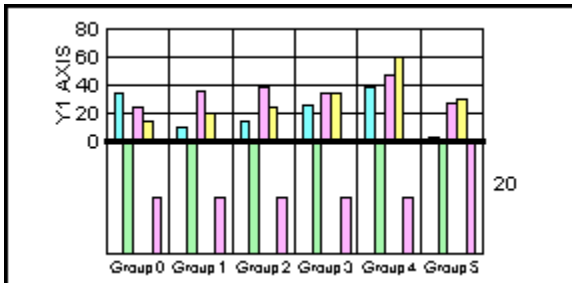
Can be one of the following:

[true](#) automatically calculates the maximum value on the Y2-axis. This value is the default.

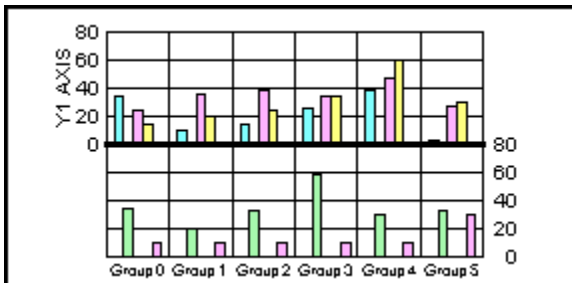
[false](#) uses value set by Y2ScaleMax.

Example:

```
setY2ScaleMaxAuto(false);
setDepthAngle(0);
setY2ScaleMax(20.0);
setLegendDisplay(false);
setY2OffScaleDisplay(true);
setDepthRadius(0);
setGraphType(21);
```



```
setY2ScaleMaxAuto(true);
```



See [Y2ScaleMax](#) on page 682, [Y2ScaleMin](#) on page 685, [Y2ScaleMinAuto](#) on page 686, [getScaleMax\(\)](#) on page 803, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969.

Y2ScaleMin

This property specifies the minimum data value that can be used to manually image the graph and its axis. When a value is assigned to this property, the Y2ScaleMinAuto property is automatically set to false.

Syntax:

```
setY2ScaleMin(value);  
value = getY2ScaleMin();
```

where:

value

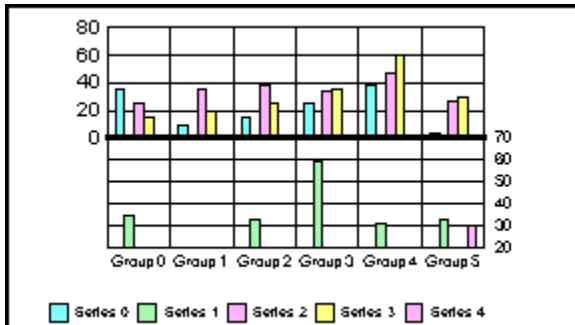
Is any value. This value is a Java double-precision number. 10.0 is the default.

Example:

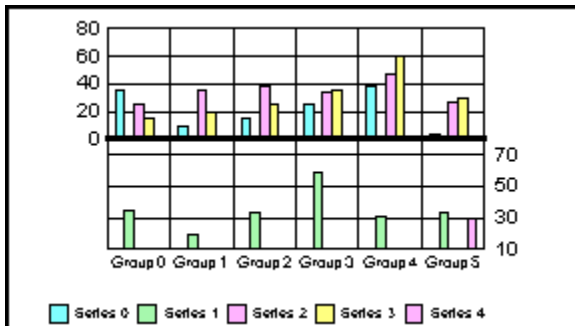
```

setGraphType(21);
setDepthAngle(0);
setY2ScaleMinAuto(false);
    setY2ScaleMin(20.0);
setDepthRadius(0);

```



```
setY2ScaleMin(10.0);
```



Note: If Y2ScaleMax is set to a value less than Y2ScaleMin, the values will be automatically switched and Y2AxisDescending will be set to true.

See [Y2ScaleMinAuto](#) on page 686, [Y2ScaleMax](#) on page 682, [Y2ScaleMaxAuto](#) on page 683, [getScaleMin\(\)](#) on page 805, [getScaleMinAuto\(\)](#) on page 806, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971.

Y2ScaleMinAuto

This property enables or disables automatic minimum scaling. When this property is set to true, the value assigned to the Y2ScaleMin property is ignored.

Syntax:

```
setY2ScaleMinAuto(boolean);
boolean = getY2ScaleMinAuto();
```

where:

boolean

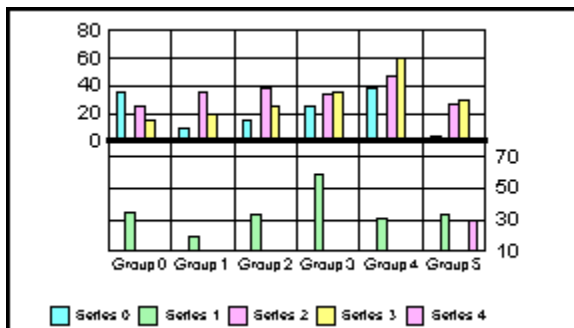
Can be one of the following:

[true](#) automatically calculates the minimum value on the Y2-axis. This value is the default.

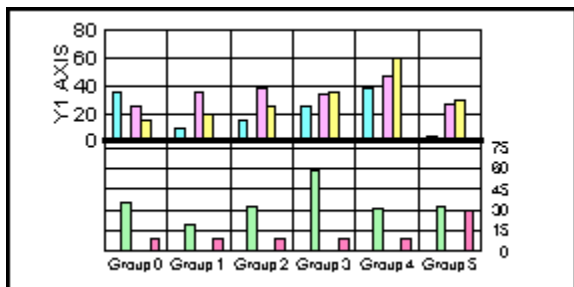
[false](#) uses value set by Y2ScaleMin.

Example:

```
setGraphType(21);
setDepthAngle(0);
  setY2ScaleMinAuto(false);
setY2ScaleMin(10.0);
setDepthRadius(0);
```



```
setY2ScaleMinAuto(true);
```



See [Y2ScaleMin](#) on page 685, [Y2ScaleMax](#) on page 682, [Y2ScaleMaxAuto](#) on page 683, [getScaleMin\(\)](#) on page 805, [getScaleMinAuto\(\)](#) on page 806, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971.

Y2TitleDisplay

This property enables or disables drawing of the Y2-axis title string.

Syntax:

```
setY2TitleDisplay(boolean);  
boolean = getY2TitleDisplay();
```

where:

boolean

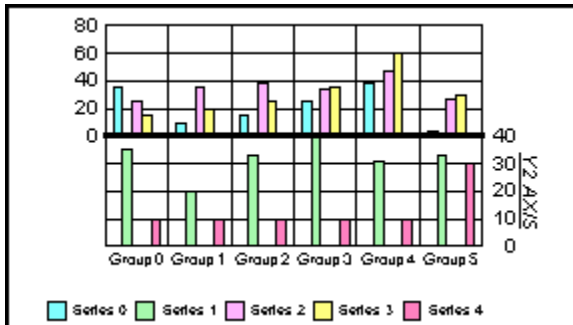
Can be one of the following:

[true](#) draws Y2-axis title. This value is the default.

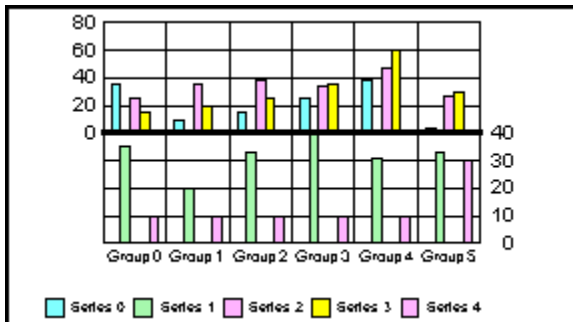
[false](#) does not draw Y2-axis title.

Example:

```
setGraphType(21);
setDepthAngle(0);
    setY2TitleString("Y2 AXIS");
setY2TitleDisplay(true);
setDepthRadius(0);
```



```
setY2TitleDisplay(false);
```



Note: If `Y2LabelDisplay` is set to false, the title string will not be drawn regardless of the `Y2TitleDisplay` or `Y2TitleString` property.

See [Y2LabelDisplay](#) on page 660, [Y2TitleString](#) on page 689.

Y2TitleString

This property defines the Y2-axis title string. When a title string is assigned to this property, the `Y2TitleDisplay` property is automatically set to true.

Syntax:

```
setY2TitleString("string");
```

```
string = getY2TitleString();
```

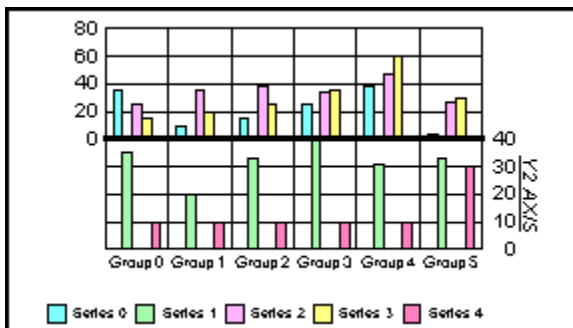
where:

string

Represents any printable characters. Null string is the default value.

Example:

```
setGraphType(21);
setDepthAngle(0);
setY2TitleDisplay(true);
    setY2TitleString("Y2 AXIS");
setDepthRadius(0);
```



Note: If Y2LabelDisplay is set to false, the title string will not be drawn regardless of the Y2TitleDisplay or Y2Titlestring property.

See [Y2LabelDisplay](#) on page 660, [Y2TitleDisplay](#) on page 688.

Y2ZeroLineDisplay

This property enables or disables drawing of the zero line on the Y2-axis when zero is within the Y2 axis scale range.

Syntax:

```
setY2ZeroLineDisplay(boolean);
boolean = getY2ZeroLineDisplay();
```

where:

boolean

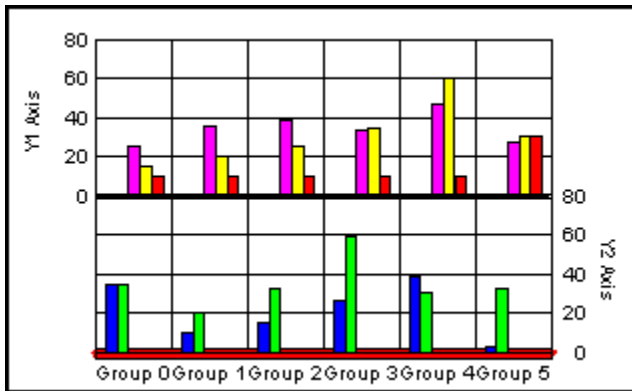
Can be one of the following:

true draws zero line on Y2-axis.

[false](#) does not draw zero line. This value is the default.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(21);
setAxisAssignment(0,1);
setY2TitleString("Y2 Axis");
setY1TitleString("Y1 Axis");
    setY2ZeroLineDisplay(true);
setLineWidth(getY2ZeroLine(), 5);
setFillColor(getY2ZeroLine(), new Color(255,0,0));
```



Note: To increase the width of the zero line, use the `getY2ZeroLine()` method to get the object ID of the line and the `setLineWidth()` method to specify the width of the line (in pixels).

Y3AxisDescending

This property changes the direction of the Y3-axis.

Syntax:

```
setY3AxisDescending(boolean);
boolean = getY3AxisDescending();
```

where:

boolean

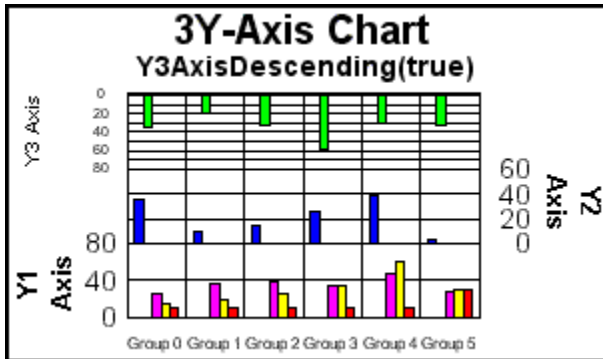
Can be one of the following:

[false](#) for ascending order from bottom-to-top in a vertical graph or from left-to-right in a horizontal graph. This value is the default.

`true` for descending order from top-to-bottom in a vertical graph or from right-to-left in a horizontal graph. This property also reverses the order of the axis labels and changes all data-related objects (markers) from their previous state.

Example:

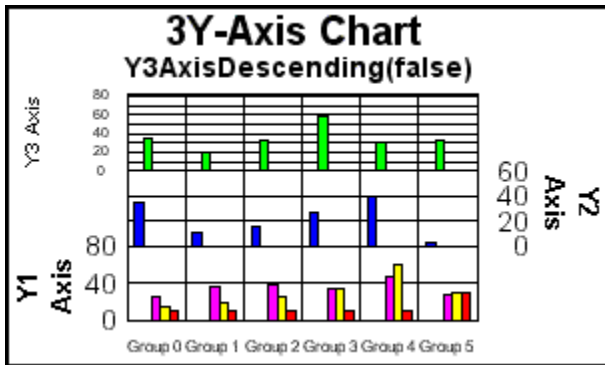
```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setGraphType(103);
setTitleString("3Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis"); setY3AxisDescending(true);
setSubtitleString("Y3AxisDescending(true)");
```



```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setGraphType(103);
setTitleString("3Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
```



```
setY3AxisDescending(false);
setSubtitleString("Y3AxisDescending(false)");
```



Y3LogScale

This property selects logarithmic or linear scaling for the Y3-axis. Any data items with Y3 values less than or equal to zero are graphed as null for log scale.

Syntax:

```
setY3LogScale(boolean)
```

```
boolean = getY3LogScale();
```

where:

boolean

Can be one of the following:

`true` uses logarithmic scale.

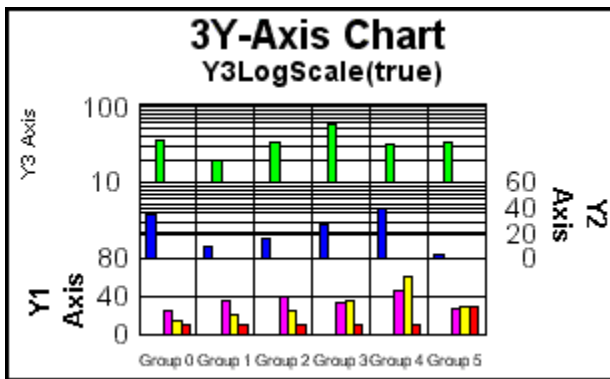
`false` uses linear scale. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setGraphType(103);
setTitleString("3Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setY3LogScale(true);
setSubtitleString("Y3LogScale(true)");

```

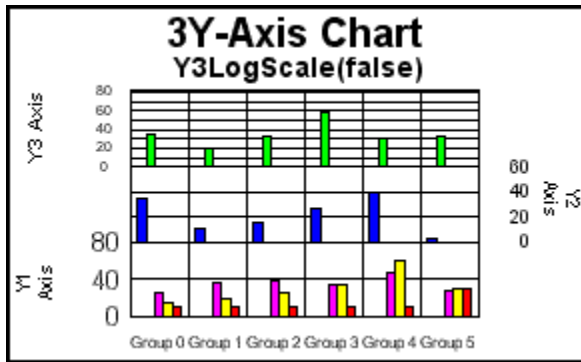


```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setGraphType(103);
setTitleString("3Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");

```

```
setY3LogScale(false);
setSubtitleString("Y3LogScale(false)");
```



Y3ZeroLineDisplay

This property enables or disables drawing of the zero line on the Y3-axis when zero is within the Y3 axis scale range.

Syntax:

```
setY3ZeroLineDisplay(boolean);
boolean = getY3ZeroLineDisplay();
```

where:

boolean

Can be one of the following:

true draws zero line on Y3-axis.

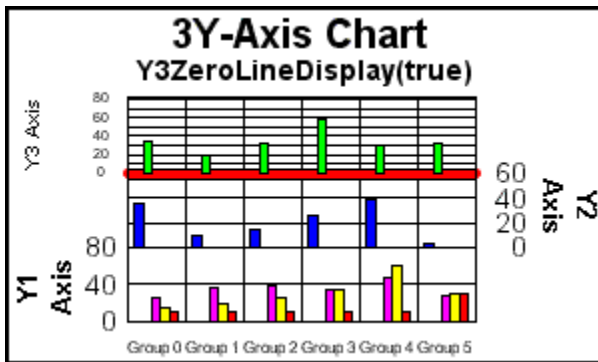
false does not draw zero line. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setGraphType(103);
setTitleString("3Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis"); setY3ZeroLineDisplay(true);
setLineWidth(getY3ZeroLine(), 5);
setFillColor(getY3ZeroLine(), new Color(255,0,0));
setSubtitleString("Y3ZeroLineDisplay(true)");

```



Note: To increase the width of the zero line, use the `getY3ZeroLine()` method to get the object ID of the line and the `setLineWidth()` method to specify the width of the line (in pixels).

Y4AxisDescending

This property changes the direction of the Y4-axis.

Syntax:

```

setY4AxisDescending(boolean);
boolean = getY4AxisDescending();

```

where:

boolean

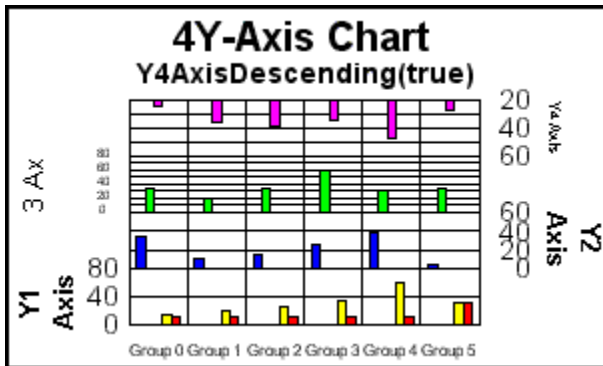
Can be one of the following:

`false` for ascending order, from bottom-to-top in a vertical graph or from left-to-right in a horizontal graph. This value is the default.

`true` for descending order, from top-to-bottom in a vertical graph or from right-to-left in a horizontal graph. This property also reverses the order of the axis labels and changes all data-related objects (markers) from their previous state.

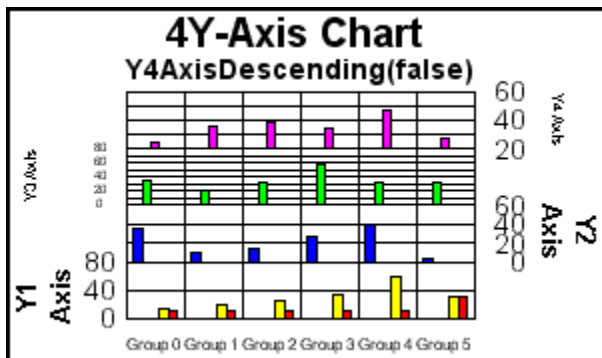
Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setGraphType(104);
setTitleString("4Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis"); setY4AxisDescending(true);
setSubtitleString("Y4AxisDescending(true)");
```



```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setGraphType(104);
setTitleString("4Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
```

```
setY4AxisDescending(false);
setSubtitleString("Y4AxisDescending(false)");
```



Y4LogScale

This property selects logarithmic or linear scaling for the Y4-axis. Any data items with Y3 values less than or equal to zero are graphed as null for log scale.

Syntax:

```
setY4LogScale(boolean)
boolean = getY4LogScale();
```

where:

boolean

Can be one of the following:

true uses logarithmic scale.

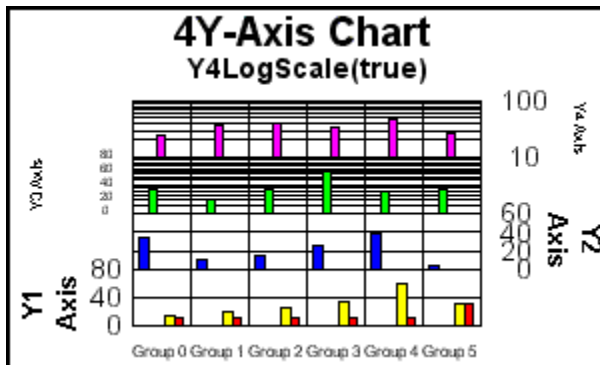
false uses linear scale. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setGraphType(104);
setTitleString("4Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
    setY4LogScale(true);
setSubtitleString("Y4LogScale(true)");

```

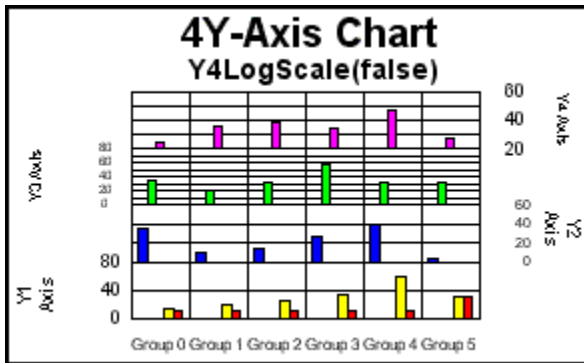


```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setGraphType(104);
setTitleString("4Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");

```

```
setY4LogScale(false);
setSubtitleString("Y4LogScale(false)");
```



Y4ZeroLineDisplay

This property enables or disables drawing of the zero line on the Y4-axis when zero is within the Y4 axis scale range.

Syntax:

```
setY4ZeroLineDisplay(boolean);
boolean = getY4ZeroLineDisplay();
```

where:

boolean

Can be one of the following:

true draws zero line on Y4-axis.

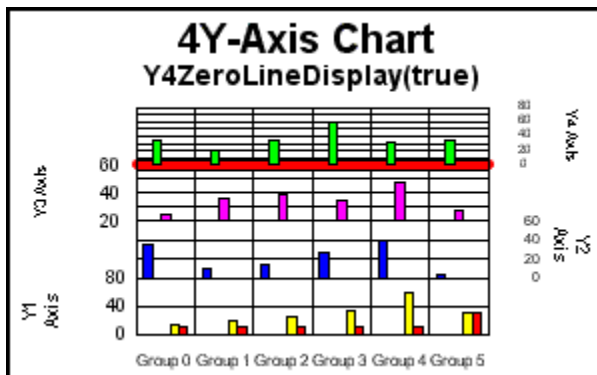
false does not draw zero line. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 3);
setAxisAssignment(getSeries(2), 2);
setGraphType(104);
setTitleString("4Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis"); setY4ZeroLineDisplay(true);
setLineWidth(getY4ZeroLine(), 5);
setFillColor(getY4ZeroLine(), new Color(255,0,0));
setSubtitleString("Y4ZeroLineDisplay(true)");

```



Note: To increase the width of the zero line, use the `getY4ZeroLine()` method to get the object ID of the line and the `setLineWidth()` method to specify the width (in pixels) of the line.

Y5AxisDescending

This property changes the direction of the Y5-axis.

Syntax:

```
setY5AxisDescending(boolean);
```

```
boolean = getY5AxisDescending();
```

where:

boolean

Can be one of the following:

[true](#) for descending order, from top-to-bottom in a vertical graph or from right-to-left in a horizontal graph. This property also reverses the order of the axis labels and changes all data-related objects (markers) from their previous state.

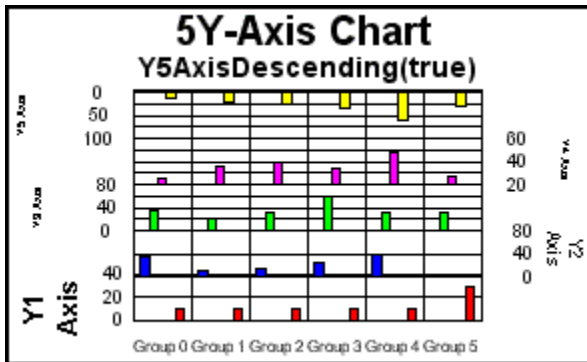
[false](#) for ascending order, from bottom-to-top in a vertical graph or from left-to-right in a horizontal graph. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis"); setY5AxisDescending(true);
setSubtitleString("Y5AxisDescending(true)");

```

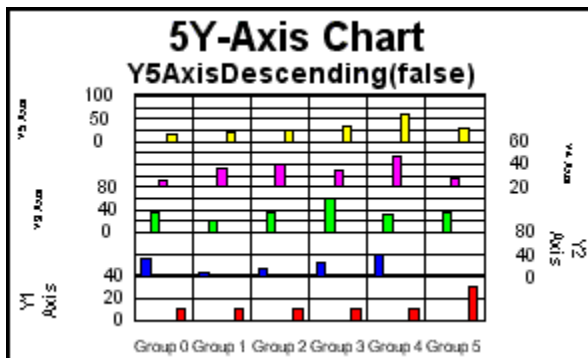


```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");

```

```
setY5AxisDescending(false);
setSubtitleString("Y5AxisDescending(false)");
```



Y5LogScale

This property selects logarithmic or linear scaling for the Y5-axis. Any data items with Y3 values less than or equal to zero are graphed as null for log scale.

Syntax:

```
setY5LogScale(boolean)
boolean = getY5LogScale();
```

where:

boolean

Can be one of the following:

true uses logarithmic scale.

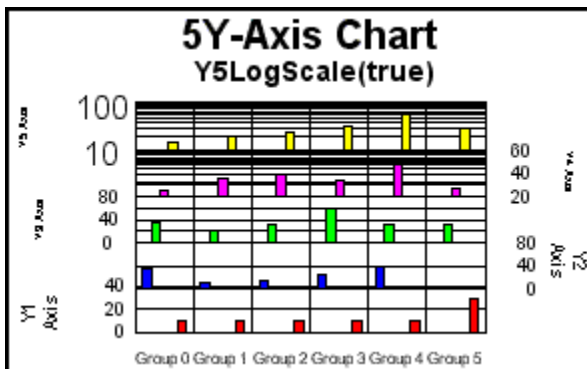
false uses linear scale. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
    setY5LogScale(true);
setSubtitleString("Y5LogScale(true)");

```

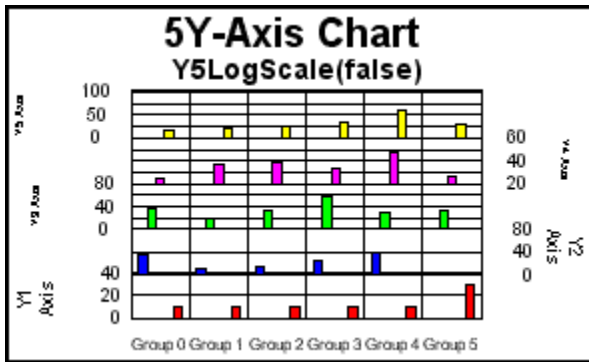


```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");

```

```
setY5LogScale(false);
setSubtitleString("Y5LogScale(false)");
```



Y5ZeroLineDisplay

This property enables or disables drawing of the zero line on the Y5-axis when zero is within the Y5 axis scale range.

Syntax:

```
setY5ZeroLineDisplay(boolean);
boolean = getY5ZeroLineDisplay();
```

where:

boolean

Can be one of the following:

true draws zero line on Y5-axis.

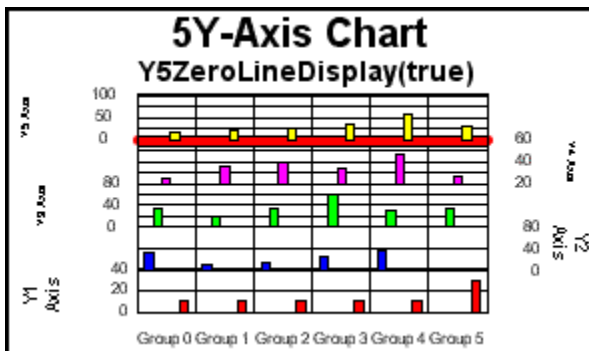
false does not draw zero line. This value is the default.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");  setY5ZeroLineDisplay(true);
setLineWidth(getY5ZeroLine(), 5);
setFillColor(getY5ZeroLine(), new Color(255,0,0));
setSubtitleString("Y5ZeroLineDisplay(true)");

```



Note: To increase the width of the zero line, use the `getY5ZeroLine()` method to get the object ID of the line and the `setLineWidth()` method to specify the width of the line (in pixels).

ZeroValueDataTextDisplay

This property controls the display of zero data text on the risers. When enabled, zeros are displayed on the chart. When disabled, zeros are hidden.

Syntax:

```

setZeroValueDataTextDisplay(boolean);
boolean = getZeroValueDataTextDisplay();

```

where:

boolean

Can be one of the following:

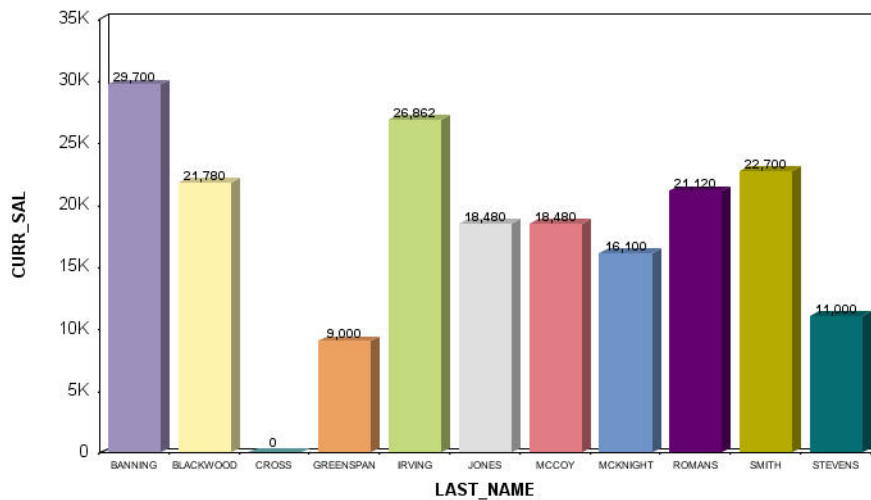
true displays the zero text value. This is the default value.

false hides the zero text value.

Example:

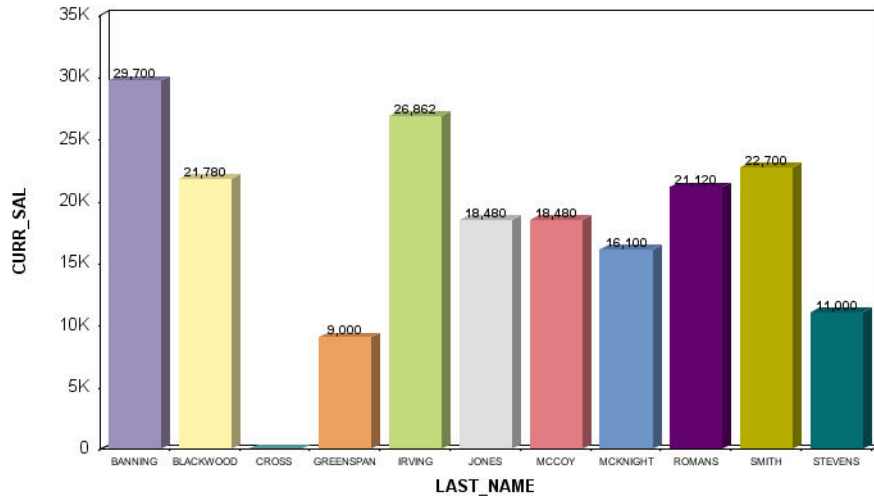
With `setZeroValueDataTextDisplay(true)`, the zero text value is displayed even though data value is zero:

`setZeroValueDataTextDisplay(true);`



With `setZeroValueDataTextDisplay(false)`, the zero text value is hidden when the data value is zero:

```
setZeroValueDataTextDisplay(false);
```



ZeroValueDataTextStackedTotalOnTopDisplay

This property controls whether the zero data value is hidden when the cumulative stack total is equal to zero.

Syntax:

```
setZeroValueDataTextStackedTotalOnTopDisplay(boolean);
```

```
boolean = getZeroValueDataTextStackedTotalOnTopDisplay();
```

where:

boolean

Can be one of the following:

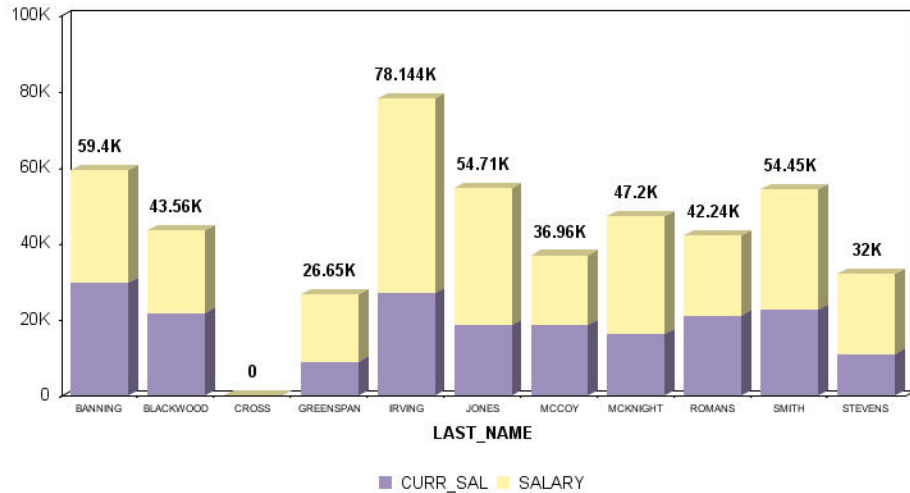
`true` displays the zero text value. This is the default value.

`false` hides the zero text value.

Example:

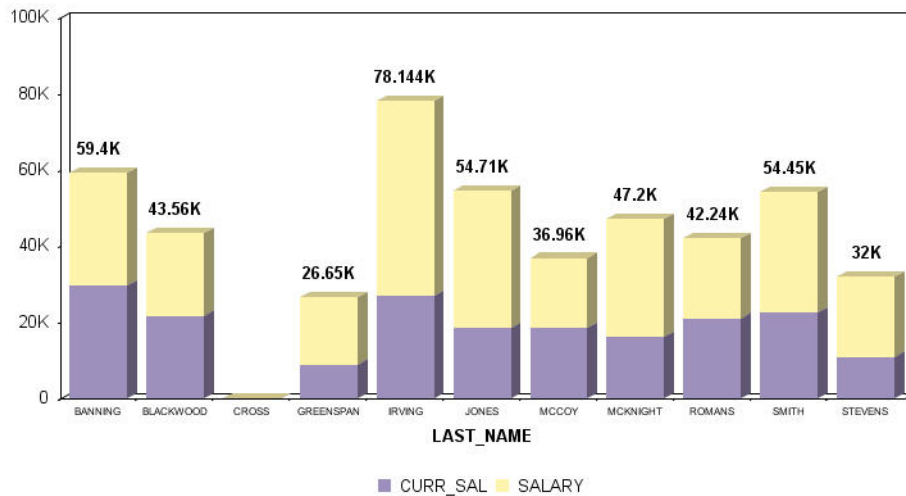
With `setZeroValueDataTextStackedTotalOnTopDisplay(true)`, the zero value is displayed even though the stack has a cumulative total of zero:

`setZeroValueDataTextStackedTotalOnTopDisplay(true);`



With `setZeroValueDataTextStackedTotalOnTopDisplay(false)`, the zero value is hidden for the stack that has a cumulative total of zero:

```
setZeroValueDataTextStackedTotalOnTopDisplay(false);
```



ZoomDirection

This property sets the direction (X-axis, Y-axis, or both) in which data zooming operations are performed by `doZoomIn()` and `doZoomOut()`. If the graph is a dual Y-axes graph and `ZoomDirection` is set to zero or two, zooming is done on both the Y1 and Y2 axes.

Syntax:

```
setZoomDirection(value);
```

```
value = getZoomDirection();
```

where:

value

Can be one of the following:

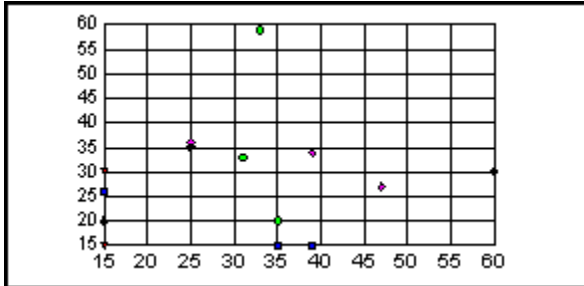
[0](#) to zoom in on both directions. 0 is the default value.

[1](#) to zoom in on just the X-axis (or ordinal axis if applicable).

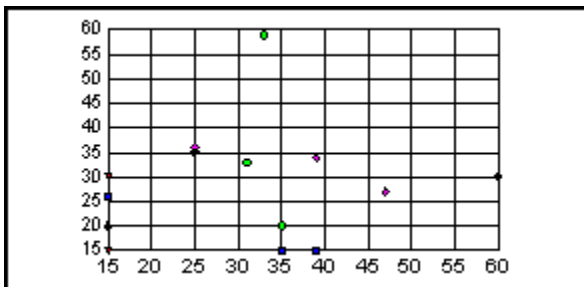
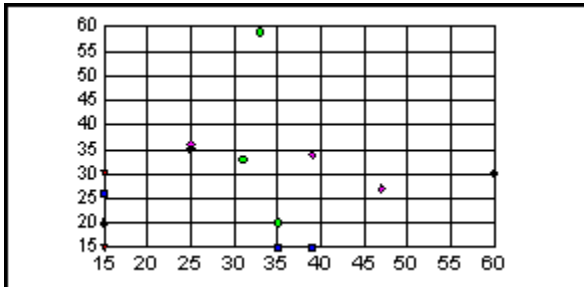
[2](#) to zoom in on just the Y-axis.

Example:

```
setGraphType(61);  
    setZoomDirection(1);  
setZoomPercentage(10);  
doZoomIn();
```



```
setZoomDirection(2);  
doZoomIn();
```



Note: This property is not currently supported in 3D graph types.

See [ZoomPercentage](#) on page 713.

ZoomPercentage

This property specifies the percentage or amount of data zooming that will be done when the `doZoomIn()` and/or `doZoomOut()` methods are used. The `ZoomDirection` property specifies the direction (X-axis, Y-axis, or both) in which data zooming is performed.

Syntax:

```
setZoomPercentage(value);
```

```
value = getZoomPercentage();
```

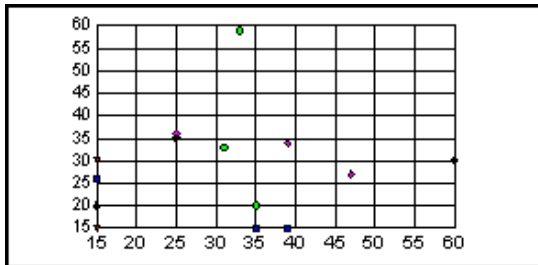
where:

value

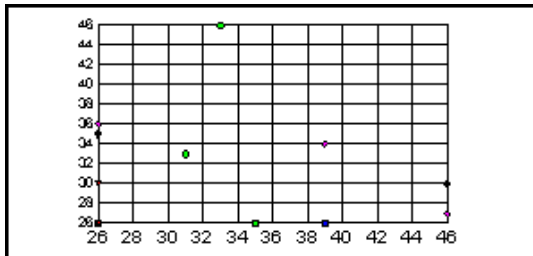
Is any real value. This value is a Java double-precision number. 50.0 is the default value.

Example:

```
setGraphType(61);
  setZoomPercentage(10.0);
doZoomIn();
```



```
clearZoomList();
  setZoomPercentage(20.0);
doZoomIn();
```



Note: This property is not currently supported in 3D graph types.

See [ZoomDirection](#) on page 711.

This topic provides information on get Methods, restore Methods, send Methods, and set Methods as well as the save() method.

In this chapter:

- ❑ [get Methods](#)
 - ❑ [restore Methods](#)
 - ❑ [save\(\)](#)
 - ❑ [send Methods](#)
 - ❑ [set Methods](#)
-

get Methods

This section contains information on the available get methods. The methods are listed in alphabetical order.

getAutoSkip()

This method returns the label skipping mode that may be assigned to primary ordinal (O1) axis labels.

Syntax:

```
int getAutoSkip();  
int getAutoSkip(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getO1Label()).

This method returns the following values for label skipping:

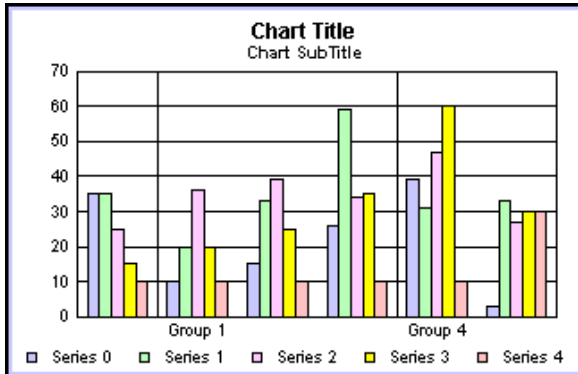
- ❑ 0 (no label skipping).
- ❑ 1 (automatic label skip).
- ❑ 2 (manual label skip).

Example:

```

setDepthRadius(0);
setO1LabelAutoSkip(2);
setO1LabelSkipBegin(1);
setO1LabelSkipCount(getAutoSkip(getO1Label()));

```

**Note:**

- ❑ If a specific object is not identified with the input parameter *id*, this method returns the associated value (if any) for the first item in the selection list.
- ❑ You may also use the object-specific property to determine label skip mode (for example, *O1LabelAutoSkip*).

See [setAutoSkip\(\)](#) on page 849, [O1LabelAutoSkip](#) on page 432.

getAxisAssignment()

In multi-Y axis graphs, this method returns the axis assignment (Y1, Y2, Y3, Y4, or Y5) for a series or an object.

Syntax:

```

int getAxisAssignment(IdentObj id);
int getAxisAssignment(int seriesID);

```

where:

id

Is an object ID returned by a *getObject()* method (for example, *getY1Axis()*).

seriesID

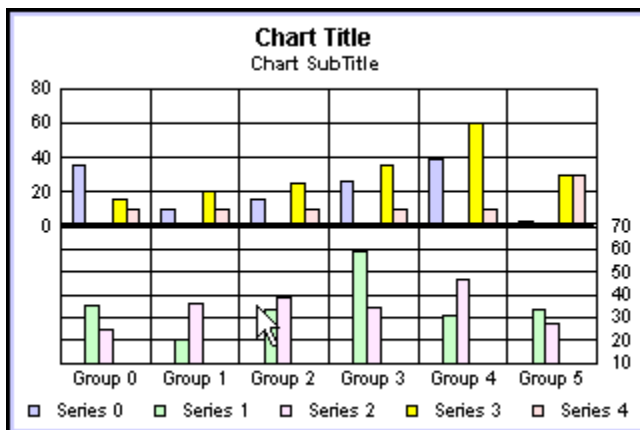
Is a series number.

This method returns the following values for axis assignments:

- ☐ 0 (Y1 axis)
- ☐ 1 (Y2 axis)
- ☐ 2 (Y3 axis)
- ☐ 3 (Y4 axis)
- ☐ 4 (Y5 axis)

Example:

```
setGraphType(21);
setDepthRadius(0);
setAxisAssignment(2, getAxisAssignment(1));
```



See [setAxisAssignment\(\)](#) on page 850, [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856, [Y1AxisSide](#) on page 619, [Y2AxisSide](#) on page 655.

getAxisDescending()

This method returns a boolean that identifies whether a numeric axis is drawn in ascending (false) or descending order (true).

Syntax:

```
boolean getAxisDescending(IdentObj id);
```

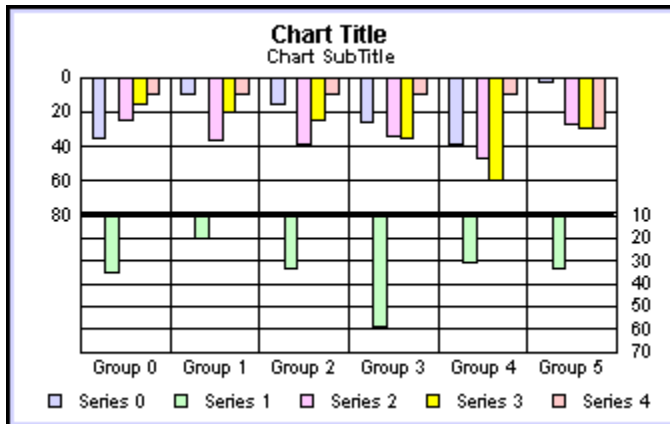
where:

id

Is an object ID returned by a getObject() method (for example, getY1Axis()).

Example:

```
setGraphType(21);
setDepthRadius(0);
setAxisDescending(getY1Axis(), true);
setAxisDescending(getY2Axis(), getAxisDescending(getY1Axis()));
```



Note: You may also use one of the axis-specific properties (for example, X1AxisDescending, Y1AxisDescending, and so on) to get the descending attribute of an axis.

See [setAxisDescending\(\)](#) on page 853, [getAxisAssignment\(\)](#) on page 716, [setAxisAssignment\(\)](#) on page 850, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856, [X1AxisDescending](#) on page 575, [Y1AxisDescending](#) on page 617, [Y2AxisDescending](#) on page 653.

getAxisOffset()

In multi-Y graphs (3 or more Y-axes), axes can overlap. The setAxisOffset() and setAxisSize() methods can be used to create a more desirable graph layout for multi-Y axes graphs. setAxisSize() sets the size of the frame, setAxisOffset() sets the axis offset from the base of the frame.

Syntax:

```
int getAxisOffset()
int getAxisOffset(IdentObj id)
```

where:

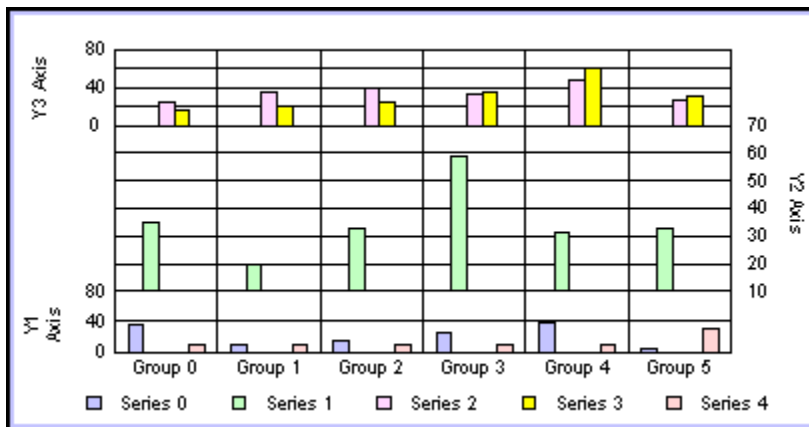
id

Is an object ID returned by a getObject() method (for example, getY2Axis()).

This method returns the value (0...100) set by setAxisOffset() for a specified Y-axis. The default value is 0 (automatic).

Example:

```
setAxisAssignment(getSeries(0), 0);
setAxisAssignment(getSeries(1), 1);
setAxisAssignment(getSeries(2), 2);
setAxisAssignment(getSeries(3), 2);
setAxisOffset(getY1Axis(), 0);
setAxisOffset(getY2Axis(), 20));
setAxisOffset(getY3Axis(), 75);
setAxisSize(getY1Axis(), getAxisOffset(getY2Axis()));
setAxisSize(getY2Axis(), 55);
setAxisSize(getY3Axis(), 25);
setGraphType(103);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setDepthRadius(0);
```



Note: If a specific object is not identified with the input parameter *id*, this method returns the associated value (if any) for the first item in the selection list.

See [setAxisOffset\(\)](#) on page 854.

getAxisSide()

This method returns an integer value that identifies where an axis is imaged in the graph.

Syntax:

```
int getAxisSide(IdentObj id);
```

where:

id

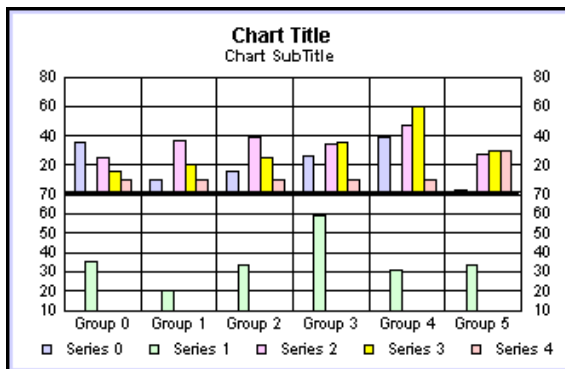
Is an object ID returned by a getObject() method (for example, getY1Axis()).

The returned integer values can be:

- ❑ 0 (left in vertical graphs/bottom in horizontal graphs).
- ❑ 1 (right in vertical graphs/top in horizontal graphs).
- ❑ 2 (both sides of the graph).

Example:

```
setGraphType(21);
setDepthRadius(0);
setAxisSide(getY1Axis(),2);
setAxisSide(getY2Axis(), getAxisSide(getY1Axis()));
```



Note: You may also use one of the axis-specific properties (for example, X1AxisSide, Y1AxisSide, and so on) to determine where an axis is imaged.

See [setAxisSide\(\)](#) on page 856, [getAxisAssignment\(\)](#) on page 716, [setAxisAssignment\(\)](#) on page 850, [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853, [O1AxisSide](#) on page 425, [X1AxisSide](#) on page 578, [Y1AxisSide](#) on page 619, [Y2AxisSide](#) on page 655.

getAxisSize()

In multi-Y graphs (3 or more Y-axes), axes can overlap. The `setAxisOffset()` and `setAxisSize()` methods can be used to create a more desirable graph layout for multi-Y axes graphs. `setAxisSize()` sets the size of the frame, `setAxisOffset()` sets the axis offset from the base of the frame.

Syntax:

```
int getAxisSize(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getY2Axis()`).

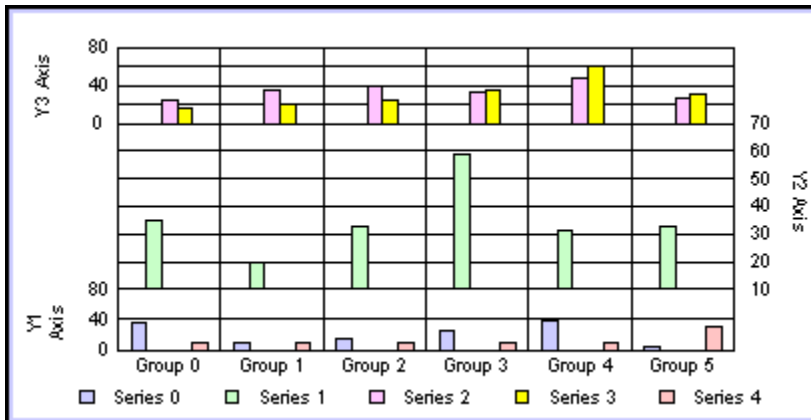
This method returns the value (0...100) set by `setAxisSize()` for a specified Y-axis. The default size is 0.

Example:

```

setAxisAssignment(getSeries(0), 0);
setAxisAssignment(getSeries(1), 1);
setAxisAssignment(getSeries(2), 2);
setAxisAssignment(getSeries(3), 2);
setAxisSize(getY1Axis(), 20);
setAxisSize(getY2Axis(), 55);
setAxisSize(getY3Axis(), 25);
setAxisOffset(getY1Axis(), 0);
setAxisOffset(getY2Axis(), getAxisSize(getY1Axis()));
setAxisOffset(getY3Axis(), 75);
setGraphType(103);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setDepthRadius(0);

```



See [setAxisSize\(\)](#) on page 857.

getBorder()

This function returns a string that describes the border of a specified object. The string can be one of the following: EmptyBorder, CompoundBorder, BevelBorder, ShadowBorder, EtchedBorder, LineBorder, or DoubleLineBorder.

Syntax:

```
cString getBorder(IdentObj anObject)
```

where:

anObject

Is an Object ID returned by a getObject() method.

Note: Currently supported objects include: an annotation object (for example, getAnnotationBox(0)), the title box object (that is, getTitleBox()), the subtitle box object (that is, getSubtitleBox()), or the footnote box object (that is, getFootnoteBox()).

See [setBorder\(\)](#) on page 859.

getBorderColor()

This method returns the border color of a specified object.

Syntax:

```
color getBorderColor (IdentObj id);
```

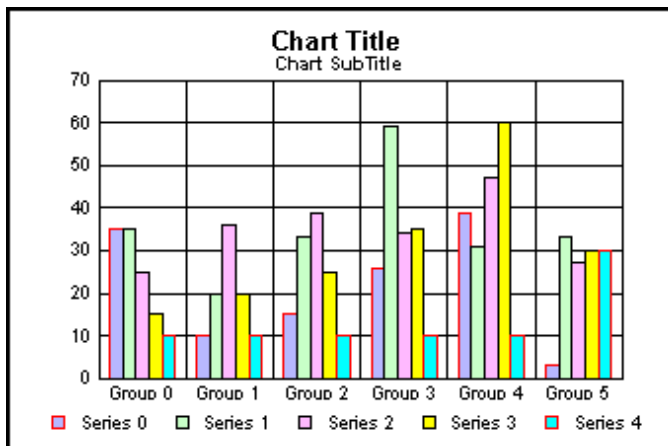
where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

Example:

```
setDepthRadius(0);
setDepthAngle(0);
setBorderColor(getSeries(4),new Color(255,0,0));
setFillColor(getSeries(4),new Color(0,255,255));
setBorderColor(getSeries(0),getBorderColor(getSeries(4)));
```



See [setBorderColor\(\)](#) on page 860, [getTransparentBorderColor\(\)](#) on page 824, [setTransparentBorderColor\(\)](#) on page 1022, [setSeriesBorderColor\(\)](#) on page 976.

getBorderType()

This method gets the border type for a specified object. The default value is NONE.

Syntax:

```
int getBorderType (IdentObj idObj);
```

where:

idObj

Is the object ID returned by [getAnnotationBox\(\)](#), [getBeveledLegend\(\)](#), [getBeveledLegendMarker\(\)](#), [getFootnoteBox\(\)](#), [getSubtitleBox\(\)](#), or [getTitleBox\(\)](#).

This method returns an integer between 0 and 5 which identifies the border type applied to a specified object.

See [Beveled Borders](#) on page 178 and [setBorderType\(\)](#) on page 861.

getBorderWidth()

This function returns the uniform insets set by [setBorderWidth\(\)](#) for all sides of an object.

Syntax:

```
Insets getBorder(IdentObj anObject)
```

where:

anObject

Is an Object ID returned by a [getObject\(\)](#) method.

Note: Currently supported objects include: an annotation object (for example, [getAnnotationBox\(0\)](#)), the title box object (that is, [getTitleBox\(\)](#)), the subtitle box object (that is, [getSubtitleBox\(\)](#)), or the footnote box object (that is, [getFootnoteBox\(\)](#)).

See [setBorderWidth\(\)](#) on page 862.

getChartFromFTP()

This method gets a graph from an FTP server. If the returned value is true, than the file was successfully retrieved, if the value is false, than the transmission failed.

Syntax:

```
boolean getChartFromFTP(
    java.lang.String strHost,
    java.lang.String strUser,
    java.lang.String strPass,
    java.lang.String strDir,
    java.lang.String strOrgFile);
```

where:

strHost

Is the host name.

strUser

Is the user name.

strPass

Is the password.

strDir

Is the directory location.

strOrgFile

Is the file name.

See [sendChartToFTP\(\)](#) on page 840, [sendGIFToFTP\(\)](#) on page 842.

getCornerArcWidth/Height()

These methods get the width and height of the corner arc of a border applied to an object. The default value is 5.

Syntax:

```
getCornerArcHeight(IdentObj id);
getCornerArcWidth(IdentObj id);
```

where:

id

Is the object ID returned by [getAnnotationBox\(\)](#), [getBeveledLegend\(\)](#), [getBeveledLegendMarker\(\)](#), [getFootnoteBox\(\)](#), [getSubtitleBox\(\)](#), or [getTitleBox\(\)](#).

These methods return the integer value of the height or width of corner arc applied to an object.

getCurveFitPolynomialOrder()

If the `setCurveFit()` method selects a polynomial fit line across a series of risers (that is, `setCurveFit(FIT_POLYNOMIAL)`), this method returns the value that is assigned by `setCurveFitPolynomialOrder()`. The `setCurveFitPolynomialOrder()` method sets the order (the largest power to which x is raised) for the polynomial fit. The default value is 3. It may be set to a value in the range 0...10.

Syntax:

```
int getCurveFitPolynomialOrder(IdentObj id)
int getCurveFitPolynomialOrder(int seriesID)
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries()`).

seriesID

Is a series number.

See [setCurveFitPolynomialOrder\(\)](#) on page 866, [getCurveFitType\(\)](#) on page 726, [setCurveFitType\(\)](#) on page 868.

getCurveFitType()

This method returns the value assigned by the `setCurveFitType()` method. The `setCurveFitType()` method selects a curve fit line and draws the selected curve type across one or more series of risers in a graph.

Syntax:

```
int getCurveFitType(IdentObj id)
int getCurveFitType(int seriesID)
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries()`).

seriesID

Is a series number.

The curve fit type selected by `setCurveFitType()` returns the following values:

CurveFitType Constant	Value	Description
FIT_NONE	0	None
FIT_LINEAR	1	Linear regression curve fit line
FIT_QUADRATIC	2	Quadratic curve fit line
FIT_POLYNOMIAL	3	Polynomial fit line
FIT_HYPERBOLIC	4	Hyperbolic curve fit line
FIT_LOGARITHMIC	5	Logarithmic curve fit line
FIT_MODHYPERBOLIC	6	Modified hyperbolic
FIT_RATIONAL	7	Rational curve fit line
FIT_EXPONENTIAL	8	Exponential curve fit line
FIT_MODEXPONENTIAL	9	Modified exponential
FIT_LOGQUADRATIC	10	Logarithmic/Quadratic curve fit line
FIT_GEOMETRIC	11	Geometric curve fit line

See [setCurveFitType\(\)](#) on page 868, [CurveFitEquationDisplay](#) on page 303, [CurveFitHighOrderFirst](#) on page 305.

getCustomDataText()

This method returns a boolean set by setCustomDataText() for a specified series object. If the returned value is true, custom data text is enabled for the specified series; false indicates that custom data text is not enabled.

Syntax:

`boolean getCustomDataText(IdentObj idSeries)`

where:

idSeries

Is an object ID returned by a getObjectID() method (for example, getSeries()).

See [setCustomDataText\(\)](#) on page 869.

getCustomDataTextFormat()

This method returns an integer that indentifies the custom data text format set by setCustomDataTextFormat() for a specified series object.

Syntax:

```
int getCustomDataTextFormat(IdentObj idSeries)
```

where:

idSeries

Is an object ID returned by a getObjectID() method (for example, getSeries()).

Returned integer values represent the following custom data text formats:

-1
setCustomDataTextFormatPattern defines the pattern.

0...21 identifies a preset number format:

0/1 = # (for example, 123 = 123)	12 = \$#B (Show B for billions) (for example, 1,234,567,891 = \$1B)
2 = %# (for example, 123 = 12,300%)	13 = #T (Show T for trillions) (for example, 1,234,567,891,234 = 1T)
3 = #.##% (for example, 123 = 12,300.0%)	14 = \$#T (Show T for trillions) (for example, 1,234,567,891,234 = \$1T)
4 = #.###% (for example, 123 = 12,300.00%)	15 = Number with thousands separators, no decimal places (for example, 1,234 = 1K)
5 = \$#.## (for example, 123 = \$123.00)	16 = Number with thousands separators, two decimal places (for example, 1,234 = 1.23K)
6 = \$# (for example, 123 = \$123)	17 = General currency format for current Locale
7 = #K (Show K for values over 999) (for example, 1,234 = 1K)	18 = Short Date Format: MM/DD/YY (for example, 10/01/03)
8 = \$#K (Show K for values over 999) (for example, 1,234 = \$1K)	19 = Medium Date Format: Mon. DD, YYYY (for example, Oct. 01, 2003)

9 = #M (Show M for millions) (for example, 1,234,567 = 1M)	20 = Long Date Format: Month DD, YYYY (for example, October 01, 2003)
10 = \$#M (Show M for millions) (for example, 1,234,567 = \$1M)	21 = Full Date Format: Day, Month DD, YYYY (for example, Wednesday, October 01, 2003)
11 = #B (Show B for billions) (for example, 1,234,567,891 = 1B)	

See [setCustomDataTextFormat\(\)](#) on page 870.

getCustomDataTextFormatPattern()

This method returns the data text format pattern set by [setCustomDataTextFormatPattern\(\)](#) for a specified series.

Syntax:

```
String getCustomDataTextFormatPattern(IdentObj idSeries)
```

where:

idSeries

Is an object ID returned by a [getObjectID\(\)](#) method (for example, [getSeries\(\)](#)).

Returned pattern strings identify the following formats:

- ❑ pattern:= subpattern{;subpattern}
- ❑ subpattern:= {prefix}integer{.fraction}{suffix}
- ❑ prefix:= '\\u0000'..'\\uFFFD' - specialCharacters
- ❑ suffix:= '\\u0000'..'\\uFFFD' - specialCharacters
- ❑ integer:= '#'* '0'* '0'
- ❑ fraction:= '0'* '#'*

Notation:

*X**

0 or more instances of X

(X|Y)

either X or Y.

X . Y

any character from X up to Y, inclusive.

S - T

characters in S, except those in T

The first subpattern is for positive numbers. The second (optional) subpattern is for negative numbers. In both cases, a comma (,) can occur inside the integer portion. The special characters used in the parts of the subpattern are:

0

a digit

#

a digit, zero shows as absent

.

a period (.) is a placeholder for decimal separator

,

a comma (,) is a placeholder for grouping separator

;

a semicolon (;) separates formats

-

a minus sign/dash (-) is the default negative prefix

%

divide by 100 and show as percentage

x

any other characters can be used in the prefix or suffix

'

a single quote (') is used to quote special characters in a prefix or suffix

If there is no explicit negative subpattern, a minus sign (-) is prefixed to the positive form (i.e., "0.00" alone is equivalent to "0.00;-0.00"). Illegal formats, such as "#.#.#" or mixing '_' and '*' in the same format, will cause a `ParseException`. The `ParseException` string will show where the error occurred. The grouping separator is commonly used for thousands, but in some countries for ten-thousands. The interval is a constant number of digits between the grouping characters, such as 100,000,000 or 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "#####,####" == "##,####,####". This class only handles localized digits where the 10 digits are contiguous in Unicode, from 0 to 9.

See [setCustomDataTextFormatPattern\(\)](#) on page 872.

getDataAsDouble()

This method returns the numeric value for any row/column intersection.

Syntax:

```
double getDataAsDouble(int row, int col);
```

where:

row

Is the row from which to retrieve data.

col

Is the column from which to retrieve data.

getDataAsString()

This method returns a value for any series/group intersection and converts it to a string.

Syntax:

```
string getDataAsString (int row, int col);
```

where:

row

Is the row from which to get the value.

col

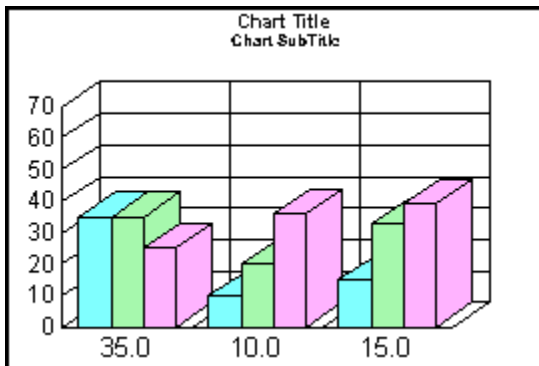
Is the column from which to get the value.

Example:

```

setScrollOffsetGroup(0);
setScrollOffsetSeries(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setGroupLabel(0,getDataAsString(0,0));
setGroupLabel(1,getDataAsString(0,1));
setGroupLabel(2,getDataAsString(0,2));
setLegendDisplay(false);

```

**getDataCoord()**

This method can be used to determine the location of an object in a graph.

Syntax:

```

int getDataCoord (IdentObj id);
int getDataCoord(IdentObj id, DataItem item, boolean bCumulative

```

where:

id

Is an object ID returned by a getObject() method (for example, getFrame()).

item

Is a data item.

bCumulative

Is either true or false.

This method returns the x,y coordinates of an object in a graph.

getDataLabel()

This method returns a label string at a specified series/group.

Syntax:

```
String getDataLabel(int s, int g);
```

where:

s

Is the series number.

g

Is the group number.

See [getGroupLabel\(\)](#) on page 761, [setGroupLabel\(\)](#) on page 914, [getSeriesLabel\(\)](#) on page 809, [setSeriesLabel\(\)](#) on page 979.

getDataTextAngle()

This method returns an integer value that identifies the angle from center point where data text is drawn.

Syntax:

```
int getDataTextAngle(IdentObj id);
```

```
int getDataTextAngle(int seriesID);
```

```
int getDataTextAngle(int seriesID, int groupID);
```

where:

id

Is an object ID returned by a getObject() method (for example, getDataText()).

seriesID

Is a series number.

groupID

Is a group number.

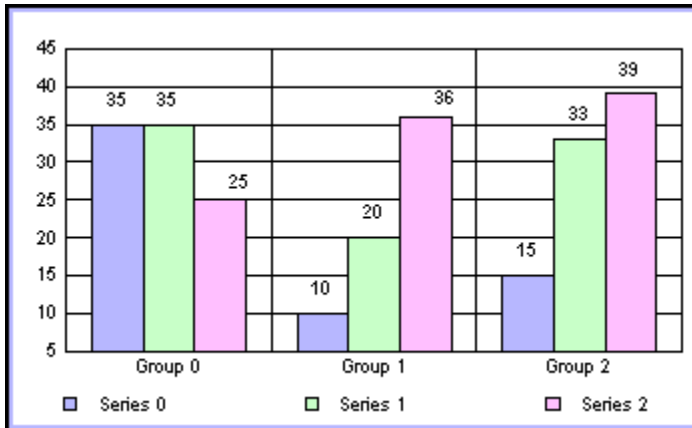
The returned integer value is the data text angle (0...360).

Example:

```

setDepthRadius(0);
setDataRange(0,0,2,2);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextAngle(2,0,30);
setDataTextAngle(2,1,getDataTextAngle(2,0));

```



Note: The `DataTextDisplay` property must be set to true in order for data text to be drawn in the graph. The `DataTextPosition` property must be set to zero.

See [setDataTextAngle\(\)](#) on page 879, [getDataTextRadius\(\)](#) on page 734, [setDataTextRadius\(\)](#) on page 881, [DataTextAngleDefault](#) on page 314, [DataTextDisplay](#) on page 316, [DataTextPosition](#) on page 322.

getDataTextRadius()

This method returns an integer value that identifies the radius from center point where data text is drawn.

Syntax:

```

int getDataTextRadius(IdentObj id);
int getDataTextRadius(int seriesID);
int getDataTextRadius(int seriesID, int groupID);

```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getDataText()`).

seriesID

Is a series number.

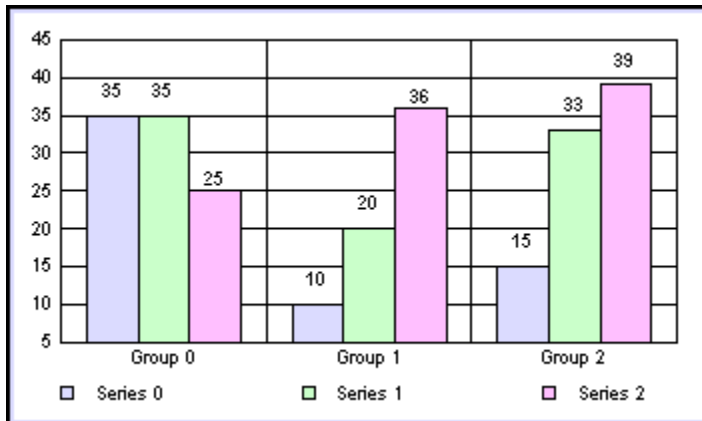
groupID

Is a group number.

The returned integer is the data text radius (0...100).

Example:

```
setDepthRadius(0);
setDataRange(0,0,2,2);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextRadius(2,0,0);
setDataTextRadius(2,1,getDataTextRadius(2,0));
```



Note: DataTextDisplay must be set to true in order for data text to be drawn in the graph.

See [setDataTextRadius\(\)](#) on page 881, [getDataTextAngle\(\)](#) on page 733, [setDataTextAngle\(\)](#) on page 879, [DataTextRadiusDefault](#) on page 324, [DataTextDisplay](#) on page 316, [DataTextPosition](#) on page 322.

getDataValue()

This method returns a data value for a data object (riser) in a graph.

Syntax:

```
double getDataValue (IdentObj id);
```

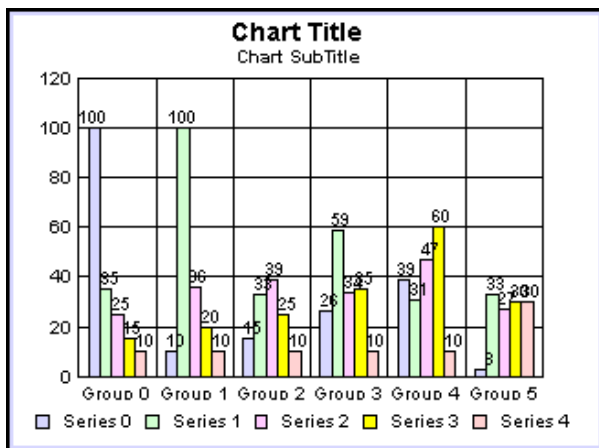
where:

id

Is an object ID returned by a getObject() method (for example, getSeriesGroup()).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setDataTextDisplay(true);
setDataValue(getSeriesGroup(0,0), 100);
setDataValue(getSeriesGroup(1,1), getDataValue(getSeriesGroup(0,0)));
```



See [setDataValue\(\)](#) on page 882.

getDisplay()

This method returns a boolean that identifies whether or not an object is visible in a graph.

Syntax:

```
boolean getDisplay(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getTitle()).

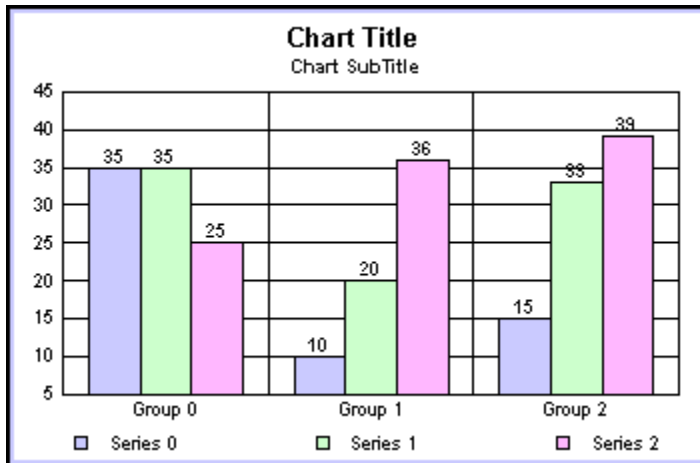
Returned values can be true (the object is displayed) or false (the object is not displayed).

Example:

```

setDepthRadius(0);
setDataRange(0,0,2,2);
setDataTextDisplay(getObjLabel());

```



See [setDisplay\(\)](#) on page 883.

getDisplayOffScale()

This method returns a boolean that identifies whether or not off-scale data points are imaged in a graph.

Syntax:

```
boolean getDisplayOffScale(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1Axis()).

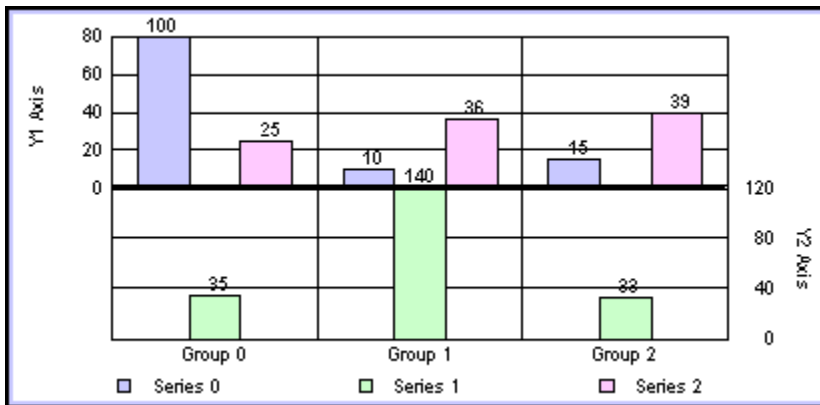
Returned values can be true (offscale values are drawn) or false (offscale values are not drawn).

Example:

```

setDepthRadius(0);
setGraphType(21);
setDataValue(getSeriesGroup(0,0),100);
setDataValue(getSeriesGroup(1,1),140);
setDataRange(0,0,2,2);
setDataTextDisplay(true);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setDisplayOffScale(getY1Axis(),true);
setDisplayOffScale(getY2Axis(), getDisplayOffScale(getY1Axis()));
setY1ScaleMax(80.0);
setY2ScaleMax(120);
setY2ScaleMaxAuto(false);
setY1ScaleMaxAuto(false);

```



Note: You may also use one of the object-specific properties to determine the off-scale display attribute of a specific object (for example, `X1OffScaleDisplay`, `Y1OffScaleDisplay`, and so on).

See [setDisplayOffScale\(\)](#) on page 884, [X1OffScaleDisplay](#) on page 606, [Y1OffScaleDisplay](#) on page 642, [Y2OffScaleDisplay](#) on page 680.

getDocumentHeightInPoints()

This method returns the document/graph height in points. It is a convenience function that converts the document height in TWIPs to points. The height in TWIPs is defined by the `DocumentHeightInTwips` property.

Syntax:

```
int getDocumentHeightInPoints();
```

Input:*None*

Note: A TWIP is 1/20th point or 1/100th millimeter. The document size is used to calculate the size of text using absolute font sizes.

See [DocumentHeightInTwips](#) on page 346, [setFontSizeAbsolute\(\)](#) on page 896.

getDocumentWidthInPoints()

This method returns the document width in points as an integer. It is a convenience function that converts the document width in TWIPs (set by `setDocumentWidthInTwips()`) to points.

Syntax:

```
int getDocumentWidthInPoints();
```

Note: A TWIP is 1/20th point or 1/100th millimeter. The document size is used to calculate the size of text using absolute font sizes.

See [DocumentWidthInTwips](#) on page 346, [setFontSizeAbsolute\(\)](#) on page 896.

getDrawTicksPerpendicular()

In 2D graphs where depth effect can be applied with `DepthAngle` and `DepthRadius` and independent tick marks are drawn, this method returns true when ticks are drawn perpendicular to axis, and false when ticks are not drawn at an angle.

Syntax:

```
boolean getDrawTicksPerpendicular(IdentObj id)
```

id

object ID returned by a `getObject()` method. It must be one of the tick mark objects (for example, `getY1MajorTick()`).

See [setDrawTicksPerpendicular\(\)](#) on page 885.

getEffectiveColorMode()

The value assigned to the `ColorMode` property defines the basic color scheme for coloring riser objects in a graph. However, some graph types (3D surface and 3D connected groups) prefer color-by-height, and most prefer color-by-series. These preferences can usually be overridden by an explicit `ColorMode` setting.

Syntax:

```
int getEffectiveColorMode()
```

This method returns the actual color mode used regardless of the ColorMode setting. Values can be:

- ❑ 0 (automatic). Use the group as the series if there is only one series. 3D connected series ribbon and areas are colored by series. 3D group ribbon and areas are colored by group.
- ❑ 1 (color by series).
- ❑ 2 (color by group).

See [ColorMode](#) on page 260.

getEffectiveGroupOffset()

This method can be used to determine whether or not all groups in the data set are imaged in the graph.

Syntax:

```
int getEffectiveGroupOffset();
```

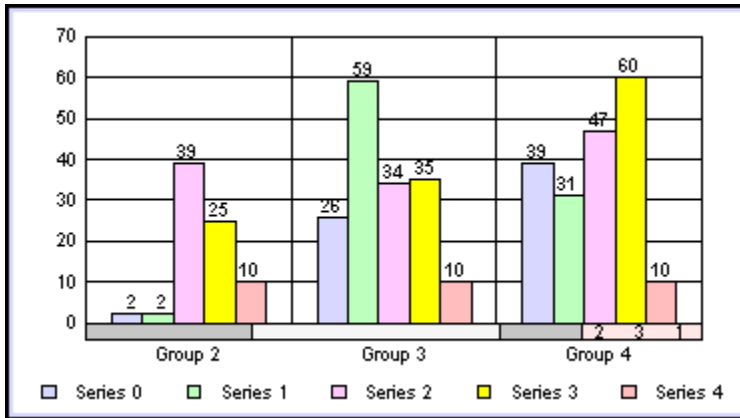
This method returns the effective group offset (that is, the total number of groups in the data set minus the number of viewable groups). The number of viewable groups is set by the ViewableGroups property.

Example:

```

setDepthRadius(0);
setViewableGroups(3);
setScrollOffsetGroup(2);
setDataTextDisplay(true);
setDataValue(getSeriesGroup(0,2), getEffectiveGroupOffset());
setDataValue(getSeriesGroup(1,2),getScrollOffsetGroup());

```



See [getEffectiveSeriesOffset\(\)](#) on page 741, [ViewableGroups](#) on page 566, [ScrollOffsetGroup](#) on page 519.

getEffectiveSeriesOffset()

This method can be used to determine whether or not all series in the data set are being imaged in the graph.

Syntax:

```
int getEffectiveSeriesOffset();
```

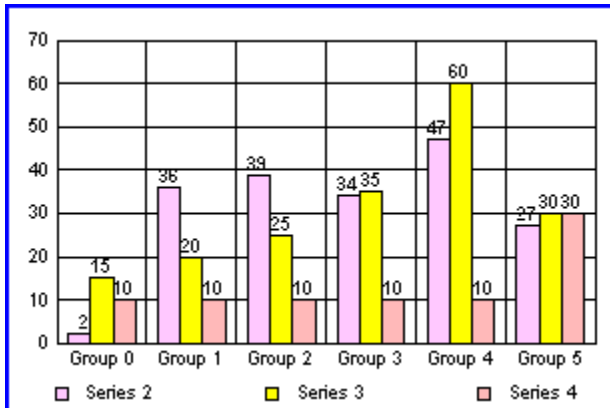
This method returns the effective series offset (that is, the total number of series in the data set minus the number of viewable series). The number of viewable series is set by the ViewableSeries property.

Example:

```

setDepthRadius(0);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setViewableSeries(3);
setScrollOffsetSeries(3);
setDataValue(getSeriesGroup(2,0), getEffectiveSeriesOffset());
setDataTextDisplay(true);

```



See [getEffectiveGroupOffset\(\)](#) on page 740, [ViewableSeries](#) on page 568, [ScrollOffsetSeries](#) on page 520.

getEffectiveViewableGroups()

This method can be used to determine whether or not data scrolling is being used in the graph.

Syntax:

```
int getEffectiveViewableGroups();
```

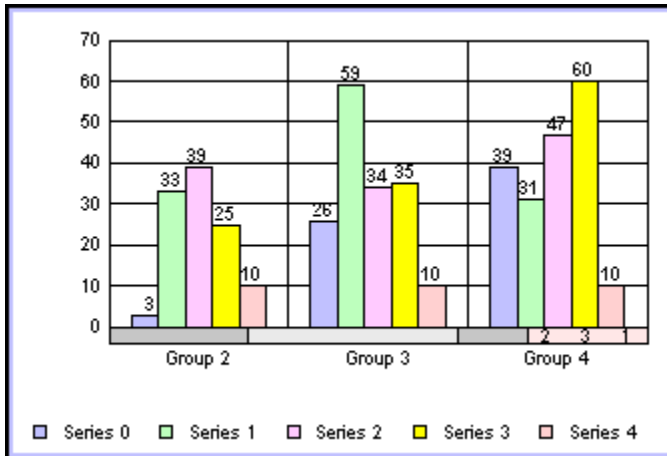
If this method returns a value greater than zero, it indicates the ViewableGroups property is set to a value greater than zero (that is, not all groups in the data set are imaged). A value of zero indicates the number of imaged groups is the same as the number of groups in the data set.

Example:

```

setDepthRadius(0);
setViewableGroups(3);
setScrollOffsetGroup(2);
setDataTextDisplay(true);
setDataValue(getSeriesGroup(0,2), getEffectiveViewableGroups());

```



See [getEffectiveGroupOffset\(\)](#) on page 740, [getEffectiveSeriesOffset\(\)](#) on page 741, [getEffectiveViewableSeries\(\)](#) on page 743, [ViewableSeries](#) on page 568, [ScrollOffsetSeries](#) on page 520.

getEffectiveViewableSeries()

This method can be used to determine whether or not data scrolling is being used in the graph.

Syntax:

```
int getEffectiveViewableSeries();
```

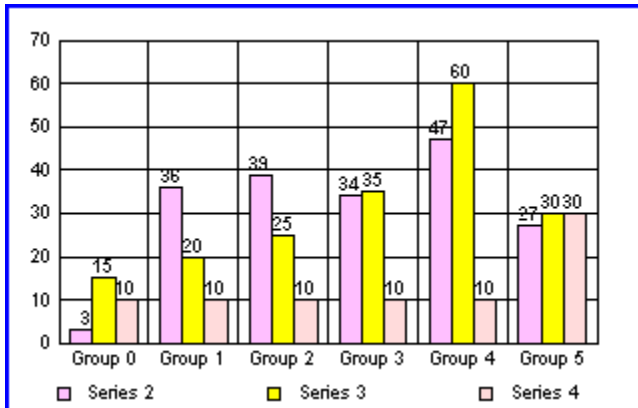
If this method returns a value greater than zero, it indicates the ViewableSeries property is set to a value greater than zero (that is, not all series in the data set are imaged). A value of zero indicates the number of imaged series is the same as the number of series in the data set (that is, ViewableSeries=0).

Example:

```

setDepthRadius(0);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setViewableSeries(3);
setScrollOffsetSeries(3);
setDataValue(getSeriesGroup(2,0), getEffectiveViewableSeries());
setDataTextDisplay(true);

```



See [getEffectiveGroupOffset\(\)](#) on page 740, [getEffectiveSeriesOffset\(\)](#) on page 741, [getEffectiveViewableGroups\(\)](#) on page 742, [ViewableSeries](#) on page 568, [ScrollOffsetSeries](#) on page 520.

getExcludeMaxLabel()

This method returns a boolean that indicates whether or not the maximum label is excluded from an axis.

Syntax:

```
boolean getExcludeMaxLabel(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1Label()).

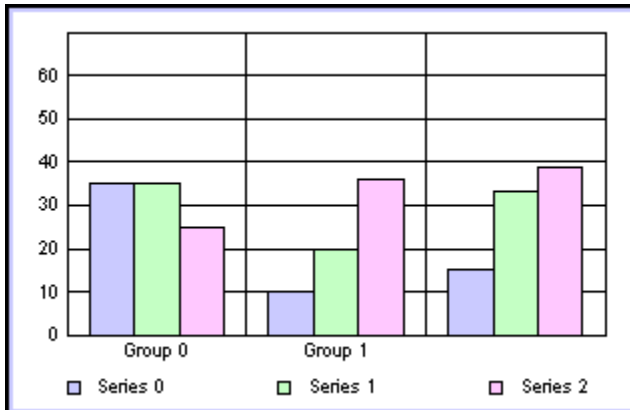
Returned values can be true (the maximum label is excluded) or false (the maximum label is included).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setO1ExcludeMaxLabel(true);
setExcludeMaxLabel(getY1Label(), getExcludeMaxLabel(getO1Label()));

```



Note: You may also use one of the axis-specific properties to determine whether or not the maximum label is excluded from a particular axis (for example, `O1ExcludeMaxLabel`, `O2ExcludeMaxLabel`, `X1ExcludeMaxLabel`, and so on).

See [setExcludeMaxLabel\(\)](#) on page 888, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMinLabel\(\)](#) on page 889, [O1ExcludeMaxLabel](#) on page 428, [O2ExcludeMaxLabel](#) on page 454, [X1ExcludeMaxLabel](#) on page 579, [Y1ExcludeMaxLabel](#) on page 621, [Y2ExcludeMaxLabel](#) on page 657.

getExcludeMinLabel()

This method returns a boolean that indicates whether or not the minimum label is excluded from an axis in a graph.

Syntax:

```
boolean getExcludeMinLabel(IdentObj id);
```

where:

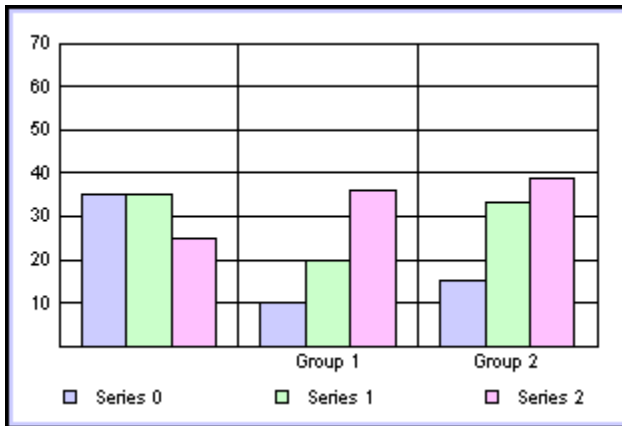
id

Is an object ID returned by a `getObject()` method (for example, `getY1Label()`).

Returned values can be true (the minimum label is excluded) or false (the minimum label is included).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setO1ExcludeMinLabel(true);
setExcludeMinLabel(getY1Label(),getExcludeMinLabel(getO1Label()));
```



Note: You may also use one of the axis-specific properties to determine whether or not minimum labels are excluded on an axis (for example, `O1ExcludeMinLabel`, `O2ExcludeMinLabel`, `X1ExcludeMinLabel`, and so on).

See [setExcludeMinLabel\(\)](#) on page 889, [getExcludeMaxLabel\(\)](#) on page 744, [setExcludeMaxLabel\(\)](#) on page 888, [O1ExcludeMinLabel](#) on page 429, [O2ExcludeMinLabel](#) on page 456, [X1ExcludeMinLabel](#) on page 581, [Y1ExcludeMinLabel](#) on page 622, [Y2ExcludeMinLabel](#) on page 658.

getFillColor()

This method returns the value of a fill color for an object in a graph. If the value is null, this means that no fill color is applied.

Syntax:

```
Color getFillColor (IdentObj id);
```

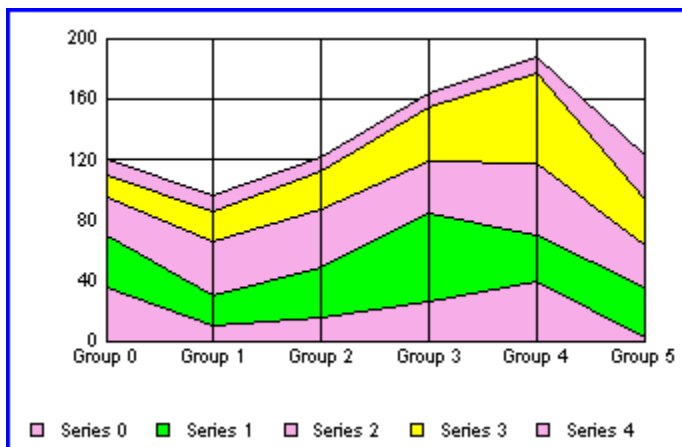
where:

id

Is an object ID returned by a `getObject()` method (for example, `getSeries()`).

Example:

```
setDepthRadius(0);
setGraphType(32);
setFillType(getSeries(0), 1);
setFillColor(getSeries(0), new Color(245,175,224));
setFillType(getSeries(1), 1);
setFillColor(getSeries(2), getFillColor(getSeries(0)));
setFillType(getSeries(4), 1);
setFillColor(getSeries(4), getFillColor(getSeries(0)));
```



See [setFillColor\(\)](#) on page 890, [getBorderColor\(\)](#) on page 723, [setBorderColor\(\)](#) on page 860, [getFillType\(\)](#) on page 747, [setFillType\(\)](#) on page 892, [getTransparentFillColor\(\)](#) on page 825, [setTransparentFillColor\(\)](#) on page 1022, [setSeriesFillColor\(\)](#) on page 977.

getFillType()

This method returns an integer that identifies the fill type assigned to an object.

Syntax:

```
int getFillType (IdentObj id);
```

where:

id

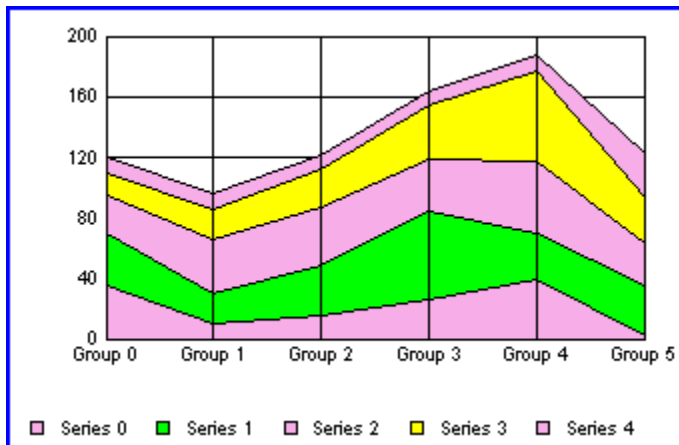
Is an object ID returned by a `getObject()` method (for example, `getSeries()`).

Returned fill type values can be:

- ☐ 1 (color)
- ☐ 2 (gradient)
- ☐ 3 (texture)

Example:

```
setDepthRadius(0);
setGraphType(32);
setFillType(getSeries(0), 1);
setFillColor(getSeries(0), new Color(245,175,224));
setFillType(getSeries(1), getFillType(getSeries(0)));
setFillColor(getSeries(2), getFillColor(getSeries(0)));
setFillType(getSeries(4), getFillType(getSeries(0)));
setFillColor(getSeries(4), getFillColor(getSeries(0)));
```



See [setFillType\(\)](#) on page 892, [getFillColor\(\)](#) on page 746, [setFillColor\(\)](#) on page 890.

getFontName()

This method returns a string that identifies the font name assigned to a text object. If the string is null, then no font name has been applied.

Syntax:

```
String getFontName(IdentObj id);
```

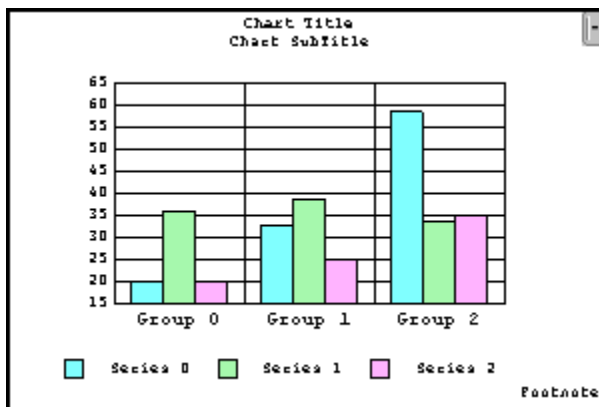

where:

id

Is an object ID returned by a `getObject()` method (for example, `getFootnote()`).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setFontName(getTitle(), "Courier");
setFontName(getFootnote(), getFontName(getTitle()));
setFontName(getSubtitle(), getFontName(getTitle()));
setFontName(getO1Label(), getFontName(getTitle()));
setFontName(getY1Label(), getFontName(getTitle()));
setFontName(getLegendText(), getFontName(getTitle()));
```



See [setFontName\(\)](#) on page 893, [getFontSizeVC\(\)](#) on page 751, [setFontSizeVC\(\)](#) on page 898, [getFontStyle\(\)](#) on page 752, [setFontStyle\(\)](#) on page 899, [setFontSize\(\)](#) on page 894.

getFontSize()

This method returns a font size in destination coordinates. It can be an integer value in the range 1...128 (inclusive) identifying the font size. The default is 12.

Syntax:

```
int getFontSize();
int getFontSize(IdentObj id);
```

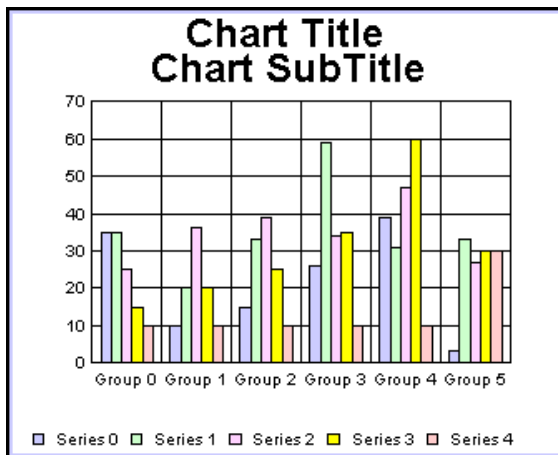
where:

id

Is an object ID returned by a getObject() method (for example, getFootnote()).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFontSizeAbsolute(getTitle(), true);
setFontSizeAbsolute(getSubtitle(), true);
setFontSize(getSubtitle(), 24);
setFontSize(getTitle(), getFontSize(getSubtitle()));
```



Note:

- ❑ If a specific object is not identified with the input parameter *id*, this method returns the associated value (if any) for the first item in the selection list.
- ❑ This method has been replaced by setFontSizeInPoints() when setFontSizeAbsolute() is true.

See [getFontSizeVC\(\)](#) on page 751, [setFontSize\(\)](#) on page 894, [setFontSizeAbsolute\(\)](#) on page 896, [setFontSizeVC\(\)](#) on page 898.

getFontSizeAbsolute()

This function returns the boolean value set by setFontSizeAbsolute(). When true, setFontSizeInPoints() must be used to set a text object's font size in points. When false, setFontSizeVC() must be used to set a text object's font size in virtual coordinates.

Syntax:

```
boolean getFontSizeAbsolute(IdentObj id)
```

where:

id

Is an object ID returned by a getObject() method (for example, getFootnote()).

See [setFontSizeAbsolute\(\)](#) on page 896.

getFontSizeInPoints()

This function returns a text object's font size in points (1...128) that has been set by setFontSizeInPoints(). The default value is 12.

Syntax:

```
int getFontSizeInPoints(IdentObj id)
```

where:

id

Is an object ID returned by a getObject() method (for example, getFootnote()).

See [setFontSizeAbsolute\(\)](#) on page 896, [setFontSizeInPoints\(\)](#) on page 897.

getFontSizeVC()

This method returns a font size in virtual coordinates.

Syntax:

```
int getFontSizeVC(IdentObj id);
```

where:

id

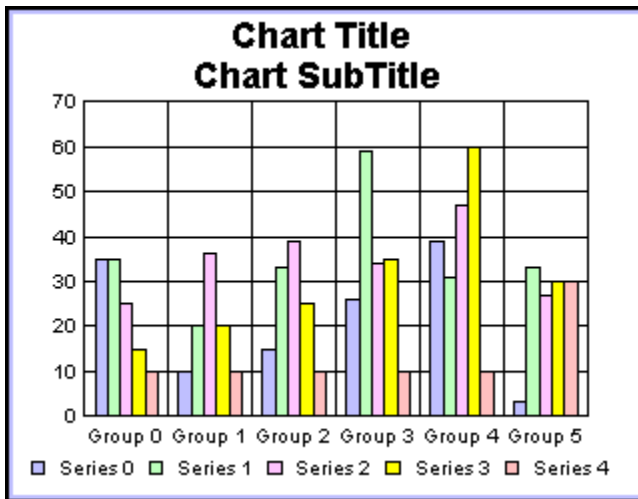
Is an object ID returned by a getObject() method (for example, getTitle()).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFontSizeAbsolute(getTitle(), false);
setFontSizeAbsolute(getSubtitle(), false);
setFontSizeVC(getSubtitle(), 2000);
setFontSizeVC(getTitle(), getFontSizeVC(getSubtitle()));

```



See [setFontSizeVC\(\)](#) on page 898, [getFontSize\(\)](#) on page 749, [setFontSize\(\)](#) on page 894, [setFontSizeAbsolute\(\)](#) on page 896.

getFontStyle()

This method returns an integer that identifies the font style assigned to a text object.

Syntax:

```
int getFontStyle(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getFootnote()).

Returned font styles can be:

0 = Plain

1 = *Italic*

2 = **Bold**

3 = ***Italic/Bold***

4 =

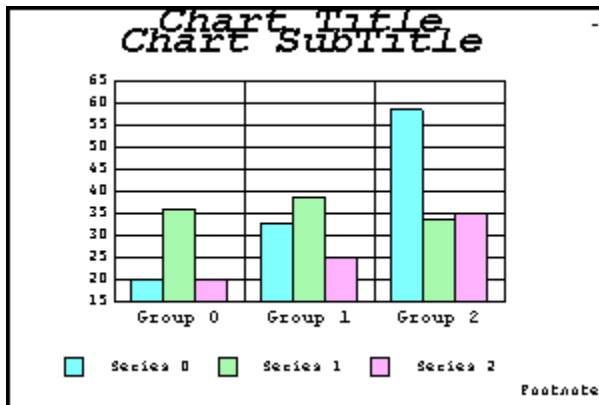
5 = *Italic/Underline*

6 = **Bold/Underline**

7 = ***Bold/Italic/Underline***

Example:

```
setAutofit(getTitle(),false);
setFontSizeVC(getTitle(),3400);
setAutofit(getSubtitle(),false);
setFontSizeVC(getSubtitle(), getFontSizeVC(getTitle()));
setFontStyle(getTitle(),3);
setFontStyle(getSubtitle(), getFontStyle(getTitle()));
setDepthAngle(0);
setDepthRadius(0);
```



See [setFontStyle\(\)](#) on page 899, [getFontSizeVC\(\)](#) on page 751, [setFontSizeVC\(\)](#) on page 898, [getFontName\(\)](#) on page 748, [setFontName\(\)](#) on page 893, [setFontSize\(\)](#) on page 894.

getFrameDepthOffset()

This method returns the depth offset of a graph frame where a DepthAngle has been applied to form a 3D effect (2.5D) on a 2D graph.

Syntax:

```
Point getFrameDepthOffset ();
```

See [DepthAngle](#) on page 333, [DepthRadius](#) on page 334.

getFrameRect()

This method returns the frame rectangle translated to destination coordinates. It makes the rectangle smaller by one pixel in each dimension and translates it to virtual coordinates.

Syntax:

```
Rectangle getFrameRect(boolean bInner);
```

where:

bInner

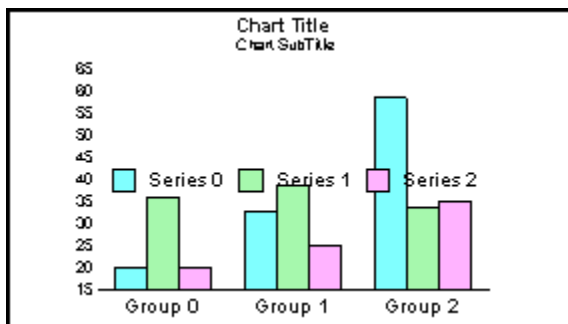
Can be one of the following:

`true` for the inside edge of frame.

`false` for the outside edge of frame.

Example:

```
setDisplay(getY1AxisLine(), false);
setDisplay(getY1MajorGrid(), false);
setDisplay(getO1MajorGrid(), false);
setDisplay(getFrame(), false);
setLegendRect(getFrameRect(false));
setDataRange(1, 1, 3, 3);
setDepthAngle(0);
setDepthRadius(0);
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
```



See [getRect\(\)](#) on page 797, [setRect\(\)](#) on page 963.

getGradientDirection()

This method returns an integer that identifies the direction of a gradient.

Syntax:

```
int getGradientDirection(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

Returned integer values can be one of the following gradient directions:

1 = right	9 = radial
2 = left	10 = radial/top/left
3 = down	11 = radial/top/right
4 = up	12 = radial/bottom/left
5 = down/left	13 = radial/bottom/right
6 = up/left	16 = radial pie
7 = down/right	17 = inverted radial pie
8 = up/right	

See [setGradientDirection\(\)](#) on page 902.

getGradientNumPins()

This method returns the number of pins in a gradient.

Syntax:

```
int getGradientNumPins(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

See [setGradientNumPins\(\)](#) on page 903.

getGradientPinLeftColor()

This method returns the color used by the left pin of a gradient.

Syntax:

```
Color getGradientPinLeftColor(IdentObj id, int nIndex);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

nIndex

Is the number of pins in the gradient.

See [setGradientPinLeftColor\(\)](#) on page 904.

getGradientPinPosition()

This method returns a value that identifies the pin position of a gradient (0.0 to 1.0).

Syntax:

```
double getGradientPinPosition(IdentObj id, int nIndex);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

nIndex

Is the number of pins in the gradient.

See [setGradientPinPosition\(\)](#) on page 906.

getGradientPinRightColor()

This method returns the color used by the right pin of a gradient.

Syntax:

```
Color getGradientPinRightColor(IdentObj id, int nIndex);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

nIndex

Is the number of pins in the gradient.

See [setGradientPinRightColor\(\)](#) on page 908.

getGridCount()

This method returns the number of minor grid lines on the primary ordinal (O1) axis.

Syntax:

```
int getGridCount(IdentObj id);
int getGridCount(int objectID);
```

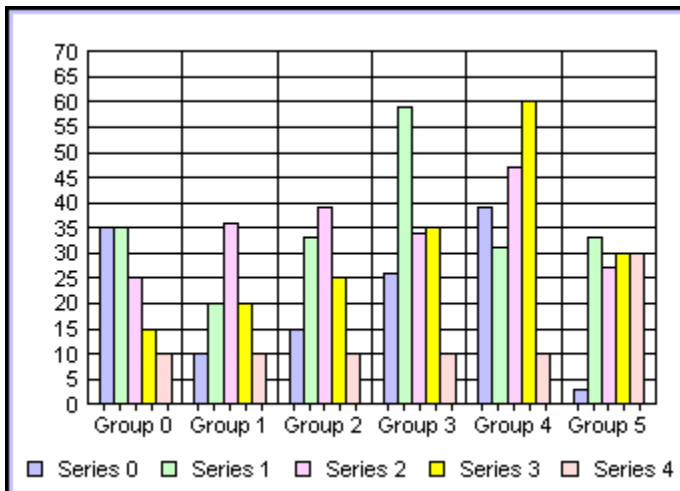
where:

id

Is an object ID returned by a getObject() method (for example, getO1MinorGrid()).

Example:

```
setDepthRadius(0);
setDisplay(getO1MinorGrid(), true);
setGridStyle(getO1MinorGrid(), 4);
setGridCount(getO1MinorGrid(), 5);
setY1MajorGridStepAuto(false);
setY1MajorGridStep(getGridCount(getO1MinorGrid()));
```



Note: You may also use O1MinorGridCount to determine the count of minor grid lines.

See [setGridCount\(\)](#) on page 909, [O1MinorGridCount](#) on page 446.

getGridStep()

This method returns the number of grid steps assigned to a numeric axis.

Syntax:

```
double getGridStep(IdentObj id);
```

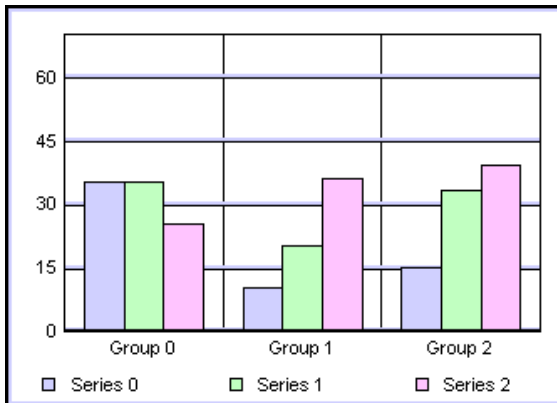
where:

id

Is an object ID returned by a getObject() method (for example, getY1MajorGrid()).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setY1MajorGridStepAuto(false);
setGridStep(getY1MajorGrid(), getGridStep(getO1MajorGrid()));
setLineWidth(getY1MajorGrid(), 2);
setFillColor(getY1MajorGrid(), new Color (0,0,255));
```



Note: You may also use one of the axis-specific properties to determine the number of grid steps assigned on an axis (for example, X1MajorGridStep, X1MinorGridStep, Y1MajorGridStep, and so on).

See [setGridStep\(\)](#) on page 910, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911.

getGridStepAuto()

This method returns a boolean that indicates whether or not grid steps are automatically calculated for a numeric axis.

Syntax:

```
boolean getGridStepAuto(IdentObj id);
```

where:

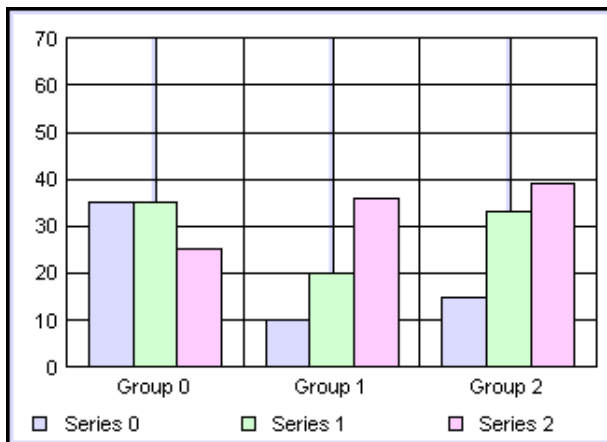
id

Is an object ID returned by a getObject() method (for example, getY1MajorGrid()).

Returned values can be true (grid steps are automatically calculated) or false (grid steps are not automatically calculated).

Example:

```
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDisplay(getO1MinorGrid(), getGridStepAuto(getY1MajorGrid()));
setLineWidth(getO1MinorGrid(),2);
setFillColor(getO1MinorGrid(), new Color (0,0,255));
```



Note: You may also use one of the axis-specific properties to determine whether or not grid steps are automatically calculated on an axis (for example, X1MajorGridStepAuto, X1MinorGridStepAuto, and so on).

See [setGridStepAuto\(\)](#) on page 911, [X1MajorGridStepAuto](#) on page 594, [X1MinorGridStepAuto](#) on page 601, [Y2MajorGridStepAuto](#) on page 670, [Y2MinorGridStepAuto](#) on page 676, [Y1MajorGridStepAuto](#) on page 633, [Y1MinorGridStepAuto](#) on page 638.

getGridStyle()

This method returns an integer that identifies the style of grid lines assigned to an axis.

Syntax:

```
int getGridStyle(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1MajorGrid()).

Returned integer values can be one of the following grid styles:

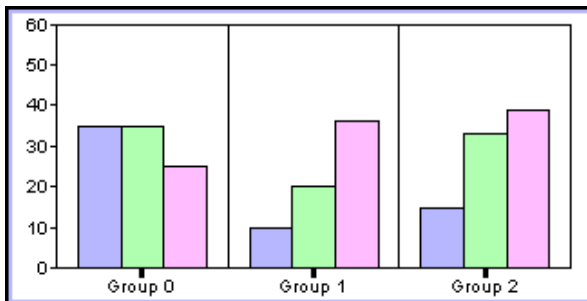
- ❑ 0 (no grid)
- ❑ 1 (normal grid)
- ❑ 2 (grids and ticks)
- ❑ 3 (inner ticks)
- ❑ 4 (outer ticks)
- ❑ 5 (spanned ticks)

Example:

```

setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDisplay(getO1MinorGrid(), getGridStepAuto(getY1MajorGrid()));
setLineWidth(getO1MinorGrid(), 2);
setFillColor(getO1MinorGrid(), new Color (0,0,255));
setGridStyle(getO1MinorGrid(), 3);
setGridStyle(getY1MajorGrid(), getGridStyle(getO1MinorGrid()));

```



See [setGridStyle\(\)](#) on page 913, [O1MajorGridStyle](#) on page 444, [O1MinorGridStyle](#) on page 449, [X1MajorGridStyle](#) on page 596, [X1MinorGridStyle](#) on page 603, [Y1MajorGridStyle](#) on page 634, [Y1MinorGridStyle](#) on page 639, [Y2MajorGridStyle](#) on page 671, [Y2MinorGridStyle](#) on page 677.

getGroupLabel()

This method returns a group label string for a specific object ID or group.

Syntax:

```
String getGroupLabel(IdentObj id);
```

```
String getGroupLabel(int groupID);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getGroup()`).

groupID

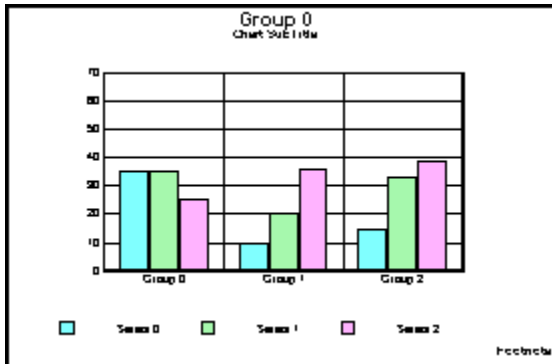
Is a group number.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setTitleString(getGroupLabel\(\));

```



See [setGroupLabel\(\)](#) on page 914, [getDataLabel\(\)](#) on page 733, [setDataLabel\(\)](#) on page 876, [getSeriesLabel\(\)](#) on page 809, [setSeriesLabel\(\)](#) on page 979.

getHasGraphics2D()

This method indicates whether or not the graph contains 2D graphics. Returned values can be true (the graph has 2D graphics) or false (the graph does not have 2D graphics).

Syntax:

```
boolean getHasGraphics2D\(\)
```

getIgnoreSeries()

This method indicates whether or not a particular series is set to be ignored by [setIgnoreSeries\(\)](#). Returned values can be true (the series is ignored) or false (the series is not ignored).

Syntax:

```
boolean getIgnoreSeries (IdentObj id);
```

```
boolean getIgnoreSeries (int seriesID);
```

where:

id

Is an object ID returned by a [getObject\(\)](#) method (for example, [getSeries\(1\)](#)).

seriesID

Is a series number.

See [setIgnoreSeries\(\)](#) on page 915.

getInsets()

This method returns the uniform insets for all sides of an annotation, title box, subtitle box, or footnote box object.

Syntax:

`Insets getInsets(IdentObj anObject)`

where:

anObject

Is an object ID returned by a getObject() method.

Note: Currently supported objects include: an annotation object (for example, getAnnotationBox(0)), the title box object (that is, getTitleBox()), the subtitle box object (that is, getSubtitleBox()), or the footnote box object (that is, getFootnoteBox()).

See [setInset\(\)](#) on page 916.

get...inset()

These methods get the top/bottom and right/left insets of borders of a specified object. The default value is 1 (which is essentially off).

Syntax:

`getBottomInset(IdentObj idObj);`

`getLeftInset(IdentObj idObj);`

`getRightInset(IdentObj idObj);`

`getTopInset(IdentObj idObj);`

where:

idObj

Is the object ID returned by getAnnotationBox(), getBeveledLegend(), getBeveledLegendMarker(), getFootnoteBox(), getSubtitleBox(), or getTitleBox().

Return:

int; For get...Inset(), the inset of the border in virtual coordinates.

See [Beveled Borders](#) on page 178.

getLabelStagger()

This method indicates whether or not labels are staggered on an axis. Returned values can be true (labels are staggered) or false (labels are not staggered).

Syntax:

```
boolean getLabelStagger();
```

```
boolean getLabelStagger(IdentObj id);
```

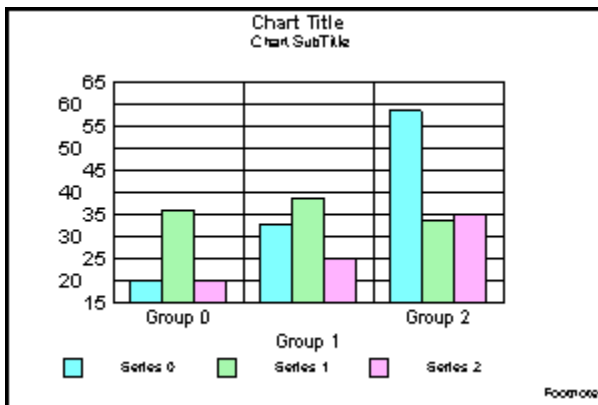
where:

id

Is an object ID returned by a getObject() method (for example, getO1Label()).

Example:

```
setViewableSeries(3);  
setDataScrollerPresenceSeries(0);  
setViewableGroups(3);  
setDataScrollerPresenceGroups(0);  
setLabelStagger(getO1Label(), true);  
setLabelStagger(getY1Label(), getLabelStagger(  
    getO1Label()));
```



Note:

- ❑ If a specific object is not identified with the input parameter id, this method returns the associated value (if any) for the first item in the selection list.
- ❑ You may also use one of the axis-specific properties to determine whether or not labels are staggered on a particular axis (for example, `X1LabelStagger`, `Y1LabelStagger`, `Y2LabelStagger`, and so on).

See [setLabelStagger\(\)](#) on page 918, [O1LabelStagger](#) on page 439, [X1LabelStagger](#) on page 588, [Y1LabelStagger](#) on page 628, [Y2LabelStagger](#) on page 664.

getLegendMinWidth()

This method returns the smallest width (in virtual coordinates) such that none of the legend labels will wrap or truncate. Note that since it specifically calculates a width, it is applicable to vertical legends only. If you try to call this method on a horizontal legend, or before the legend has been created, this method returns zero.

Syntax:

```
int getLegendMinWidth ( );
```

getLegendRect()

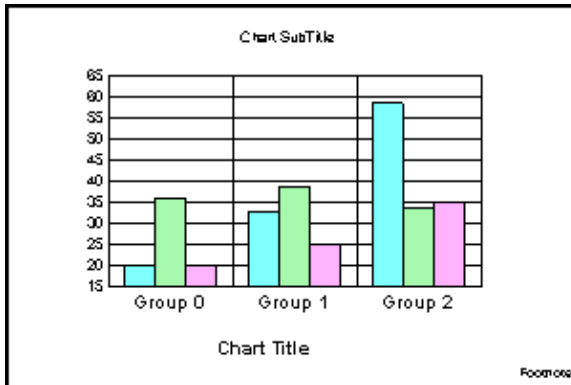
This method returns the size and location of the legend rectangle.

Syntax:

```
Rectangle getLegendRect( );
```

Example:

```
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setRect(getTitle(), getLegendRect());
setLegendDisplay(false);
```



See [LegendAutomatic](#) on page 397, [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400, [LegendReverse](#) on page 404.

getLevelDateFormat()

This function returns the date format string set by `setLevelDateFormat()` for a graph that uses a time scale axis.

Syntax:

```
String getLevelDateFormat(int level)
```

where:

int

Is the level (0...2) from which to get the date format.

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [setLevelDateFormat\(\)](#) on page 923 and [Time Scale Axis Graphs](#) on page 221.

getLevelFillColor()

This function returns the fill color assigned to the specified object or level, set by `setLevelFillColor()` for a graph that uses a time scale axis.

Syntax:

```
Color getLevelFillColor(IdentObj id)
```

```
Color getLevelFillColor(int level)
```

id

Is an object ID of the level from which to get the fill color (that is, `getTimeScaleLevel(n)`).

level

Is the level (0...2) from which to get the fill color.

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [setLevelFillColor\(\)](#) on page 924 and [Time Scale Axis Graphs](#) on page 221.

getLevelHeight()

This function returns the level height set by `setLevelHeight()` for a graph that uses a time scale axis.

Syntax:

```
int getLevelHeight(int level)
```

where:

level

Is the level (0...2) from which to get the level height.

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [setLevelHeight\(\)](#) on page 925 and [Time Scale Axis Graphs](#) on page 221.

getLevelInterval()

This function returns the level interval set by `setLevelInterval()` for a graph that uses a time scale axis.

Syntax:

```
int getLevelInterval(int level)
```

where:

level

Is the level (0...2) from which to get the interval.

```
TIME_INTERVAL_UNDEFINED = 0
TIME_INTERVAL_SECONDS = 1
TIME_INTERVAL_MINUTES = 2
TIME_INTERVAL_HOURS = 3
TIME_INTERVAL_DAYS = 4
TIME_INTERVAL_WEEKS = 5
TIME_INTERVAL_MONTHS = 6
TIME_INTERVAL_QUARTERS = 7
TIME_INTERVAL_YEARS = 8
```

Note: A time scale axis is only defined in clustered/stacked bar, line, and scatter graphs.

See [setLevelInterval\(\)](#) on page 925 and [Time Scale Axis Graphs](#) on page 221.

getLevelTextFormat()

This function returns the level text format integer set by `setLevelTextFormat()` for a graph that uses a time scale axis.

Syntax:

```
int getLevelTextFormat(int level)
```

level

Is the level from which to get the text format (0...2).

Note: A time scale axis is only defined in clustered/stacked bar, line, and scatter graphs.

See [setLevelTextFormat\(\)](#) on page 927 and [Time Scale Axis Graphs](#) on page 221.

getLineBasicStrokeType()

This method returns the line basic stroke type set by `setLineBasicStrokeType()` for a specified line object.

Syntax:

```
int getLineBasicStrokeType(IdentObj id)
```

id

Is an object ID of a line object returned by a getObject() method (that is, getY1MajorGrid())

Returned integer (0...5) values can be one of the following basic stroke types:

0 = Don't use Java2D. Draw line with regular JDK 1.1.8 code

1 = Use Java2D BasicStroke() which constructs a new BasicStroke with defaults for all attributes.

2 = Use Java2D BasicStroke(float width) which constructs a solid BasicStroke with the specified line width and default values for the cap and join styles.

3 = Use Java2D BasicStroke(float width, int cap, int join) which constructs a solid BasicStroke with the specified attributes.

4 = Use Java2D BasicStroke(float width, int cap, int join, float miterlimit) which constructs a solid BasicStroke with the specified attributes.

5 = Use Java2D BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase) which constructs a new BasicStroke with the specified attributes

See [setLineBasicStrokeType\(\)](#) on page 928.

getLineCap()

This method returns the cap style that has been applied to a specified line object by setLineCap().

Syntax:

```
int getLineCap(IdentObj id)
```

where:

id

Is the object ID of a line object returned by a getObject() method (that is, getY1MajorGrid())

The method return an integer that defines the decoration that has been applied to the ends of unclosed subpaths and dash segments of a specified line as defined by the Java 2D BasicStroke object (for example, CAP_BUTT, CAP_ROUND or CAP_SQUARE). See the java.awt.BasicStroke constructors for values that may be returned by this method.

See [setLineCap\(\)](#) on page 929.

getLineDash()

This method returns a vector (java.lang.Float) of floating point numbers that defines the dash pattern set by setLineDash() for a specified line object.

Syntax:

```
Vector getLineDash(IdentObj id)
```

where:

id

Is an object ID of a line object returned by a getObject() method (that is, getY1MajorGrid()).

See [setLineDash\(\)](#) on page 930.

getLineDashFloat()

This method returns an array representing the lengths of the dash segments specified by setLineDash() for a specified line object. It translates a Vector of Floats into a float[] array.

Syntax:

```
float[] getLineDashFloat(IdentObj id)
```

id

Is an object ID of a line object returned by a getObject() method (that is, getY1MajorGrid())

See [setLineDash\(\)](#) on page 930.

getLineDashPhase()

The method returns the current dash phase. The dash phase is a distance specified in user coordinates that represents an offset into the dashing pattern. In other words, the dash phase defines the point in the dashing pattern that will correspond to the beginning of the stroke. This value is set by setLineDashPhase().

Syntax:

```
float getLineDashPhase(IdentObj id)
```

id

Is an object ID of a line object returned by a getObject() method (that is, getY1MajorGrid())

See [setLineDashPhase\(\)](#) on page 931.

getLineJoin()

This method returns an integer that defines a line object's join decoration applied by `setLineJoin()`. A join decoration is a decoration that has been applied where path segments meet (for example, `JOIN_ROUND`, `JOIN_BEVEL`, or `JOIN_MITER`).

Syntax:

```
int getLineJoin(IdentObj id)
```

where:

id

Is an object ID of a line object returned by a `getObject()` method (that is, `getY1MajorGrid()`)

Note: See the `java.awt.BasicStroke` constructors for constants that can be specified in *NewValue*.

See [setLineJoin\(\)](#) on page 931.

getLineMiterLimit()

When `setLineJoin()` applies `JOIN_MITER` to a line object, this method returns a value that defines the limit to trim the miter join. This value is set by `setLineMiterLimit()`

Syntax:

```
float getLineMiterLimit(IdentObj id)
```

id

Is an object ID of a line object returned by a `getObject()` method (that is, `getY1MajorGrid()`).

See [getLineJoin\(\)](#) on page 771, [setLineJoin\(\)](#) on page 931, [setLineMiterLimit\(\)](#) on page 932.

getLineWidth()

This method returns the thickness (in pixels) of a series line.

Syntax:

```
int getLineWidth (IdentObj id)
```

where:

id

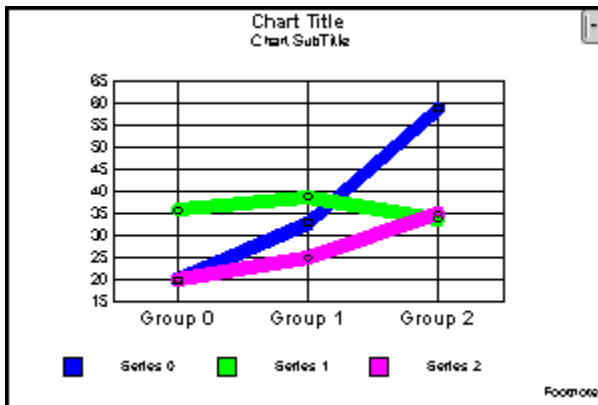
Is the object ID of a line object.

Example:

```

setGraphType(41);
setY1MajorGridStep(10.0);
setLineWidth(getSeries(0),10);
setLineWidth(getSeries(1), getLineWidth(getSeries(0)));
setLineWidth(getSeries(2), getLineWidth(getSeries(0)));
setVisibleSeries(3);
setDataScrollerPresenceSeries(0);
setVisibleGroups(3);
setDataScrollerPresenceGroups(0);

```



See [setLineWidth\(\)](#) on page 937.

getLogScale()

This method returns a boolean that indicates whether a numeric axis uses a logarithmic (true) or linear (false) scale.

Syntax:

```
boolean getLogScale(IdentObj id);
```

where:

id

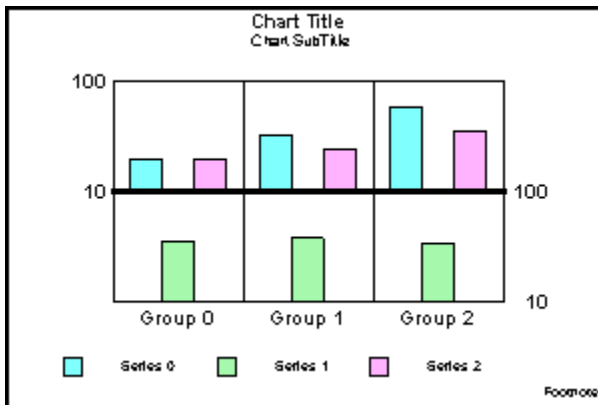
Is an object ID returned by a getObject() method (for example, getY1Axis()).

Example:

```

setDataRange(0,0,2,2);
setDepthAngle(0);
setDepthRadius(0);
setGraphType(21);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setDataScrollerPresenceGroups(0);
setLogScale(getY1Axis(),true);
setLogScale(getY2Axis(),getLogScale(getY1Axis()));

```



Note: You may also use one of the axis-specific properties to determine whether logarithmic or linear scale is used on a particular axis (for example, `X1LogScale`, `Y1LogScale`, `Y2LogScale`, and so on).

See [setLogScale\(\)](#) on page 938, [X1LogScale](#) on page 589, [Y1LogScale](#) on page 629, [Y2LogScale](#) on page 666.

getLogScaleBase()

When logarithmic scaling is used on a numeric axis in a graph, this property returns a value that identifies the lowest/base value drawn on the axis.

Syntax:

```
double getLogScaleBase(IdentObj id)
```

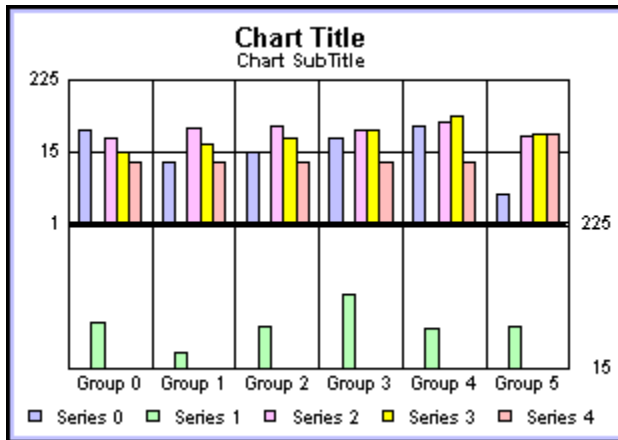
where:

id

Is an object ID returned by a getObject() method (for example, getY1Axis()).

Example:

```
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setY1LogScale(true);
setY2LogScale(true);
setLogScaleBase(getY1Axis(), 15);
setLogScaleBase(getY2Axis(), getLogScaleBase(getY1Axis()));
```



See [getLogScale\(\)](#) on page 772, [setLogScaleBase\(\)](#) on page 940.

getMarkerShape()

This method returns an integer value that identifies the shape of a marker for a specified object.

Syntax:

```
int getMarkerShape();
int getMarkerShape(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

Returned values can be 0...32. Zero is for no shape. Marker shapes 1...18 identify one of the following marker shapes:

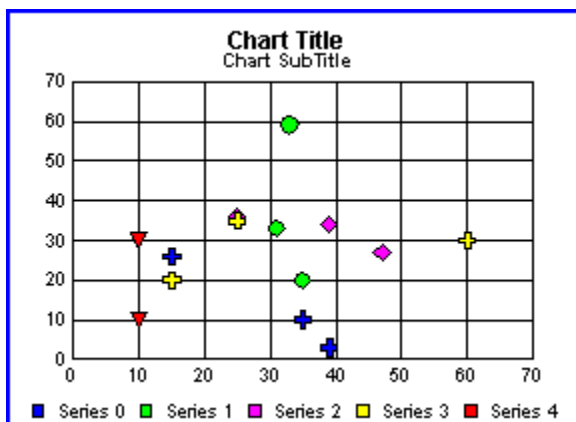
1	2	3	4	5	6	7	8	9
								
10	11	12	13	14	15	16	17	18
								

Example:

```

setDepthRadius(0);
setDepthAngle(0);
setGraphType(61);
setMarkerSizeDefault(60);
setMarkerShape(getSeries(0),getMarkerShape(getSeries(3)));

```



Note: If an object ID (id) is not specified, this method returns the marker shape (if any) for the first item in the selection list.

See [getMarkerSize\(\)](#) on page 776, [setMarkerSize\(\)](#) on page 942, [getMarkerTemplate\(\)](#) on page 777, [MarkerDisplay](#) on page 411, [MarkerSizeDefault](#) on page 412.

getMarkerSize()

This method returns an integer value that identifies the size of a marker (0...100).

Syntax:

```
int getMarkerSize(IdentObj id);  
int getMarkerSize(int seriesID);  
int getMarkerSize(int seriesID, int groupID);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

seriesID

Is a series number.

groupID

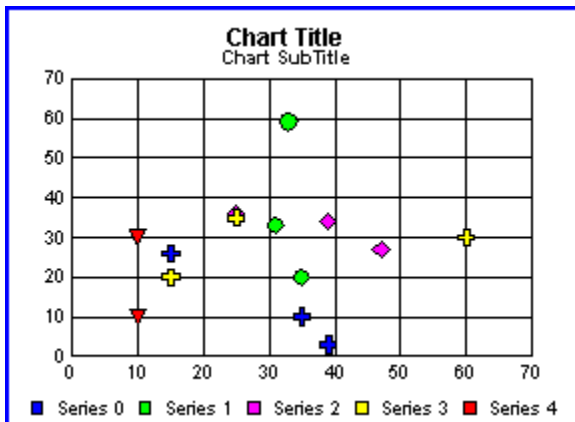
Is a group number.

Example:

```

setGraphType(61);
setMarkerSize(getSeries(0),90);
setMarkerSize(getSeries(1), getMarkerSize(getSeries(0)));
setMarkerSize(getSeries(2), getMarkerSize(getSeries(0)));
setMarkerSize(getSeries(3), getMarkerSize(getSeries(0)));
setMarkerShape(getSeries(0),4);
setMarkerShape(getSeries(1), getMarkerShape(getSeries(0)));
setMarkerShape(getSeries(2), getMarkerShape(getSeries(0)));
setMarkerShape(getSeries(3), getMarkerShape(getSeries(0)));

```



See [setMarkerSize\(\)](#) on page 942, [getMarkerShape\(\)](#) on page 774, [setMarkerShape\(\)](#) on page 940, [MarkerDisplay](#) on page 411, [MarkerSizeDefault](#) on page 412.

getMarkerTemplate()

This method returns the marker template (0...32) that may be assigned to an object. The marker template can be assigned to an object or all objects in a graph by the [setMarkerShape\(\)](#) method.

Syntax:

```
MarkerTemplate getMarkerTemplate(IdentObj id);
```

where:

id

Is an object ID returned by a [getObject\(\)](#) method (for example, [getSeries\(\)](#)).

See [getMarkerShape\(\)](#) on page 774, [setMarkerShape\(\)](#) on page 940, [MarkerDisplay](#) on page 411, [MarkerSizeDefault](#) on page 412.

getMaxNumAnnotations()

The following method can be used to get the maximum number of annotations.

Syntax:

```
int getMaxNumAnnotations();
```

For annotations in excess of the default (30), the `setTransparentFillColor()` and `setTransparentBorderColor()` methods should be used to set transparent fill and border colors for the associated annotation box as desired:

- ❑ `setTransparentFillColor(getAnnotationBox(nnn), true | false);`
- ❑ `setTransparentBorderColor(getAnnotationBox(nnn), true | false);`

getMillisFromDate()

This function returns the time in milliseconds since January 1, 1970, 0:00:00 GMT for the date/time specified by the input parameters.

Syntax:

```
long getMillisFromDate(  
    int year,  
    int month,  
    int day,  
    int hour,  
    int minute,  
    int second)
```

where:

`year`

Is a year number (for example, 2004).

`month`

Is a month number 1...12 (1=January, 12=December).

`day`

Is a day in month (that is, 1...31).

`hour`

Is an hour in day (1...24).

`minute`

Is a minute in hour (1...60).

`second`

Is a second in minute (1...60).

getMinimumSize()

This method returns the minimum dimensions (height and width) of a graph.

Syntax:

```
java.awt.Dimension getMinimumSize();
```

getMinNumGroups()

This method returns the minimum number of groups in a specified graph type.

Syntax:

```
int getMinNumGroups(int nIntegerGraphType)
```

```
int getMinNumGroups()
```

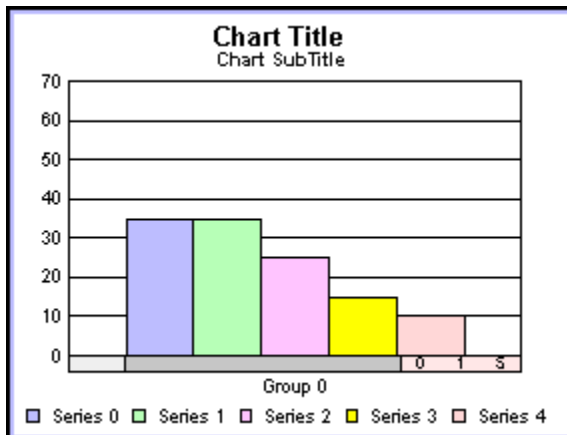
where:

nIntegerGraphType

Is a graph type number (0...106).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(getMinNumGroups(17));
```



Note: If a graph type is not specified with `nIntegerGraphType`, this method returns the minimum number of groups required for the current graph type.

See [getMinNumSeries\(\)](#) on page 780.

getMinNumSeries()

This method returns the minimum number of series in a specified graph type.

Syntax:

```
int getMinNumSeries()
int getMinNumSeries(int nIntegerGraphType)
```

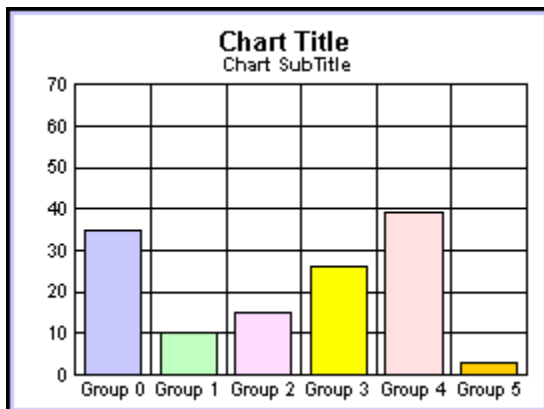
where:

nIntegerGraphType

Is a graph type number (0...106).

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setViewableSeries(getMinNumSeries(17));
```



Note: If a graph type is not specified with `nIntegerGraphType`, this method returns the minimum number of series required for the current graph type.

See [getMinNumGroups\(\)](#) on page 779.

getNonZeroBaseline()

This method indicates whether or not a non-zero baseline has been set by `setNonZeroBaseline()` on a numeric axis. Returned values can be true (a non-zero baseline is set) or false (a zero baseline is set).

Syntax:

```
boolean getNonZeroBaseline(IdentObj id)
```

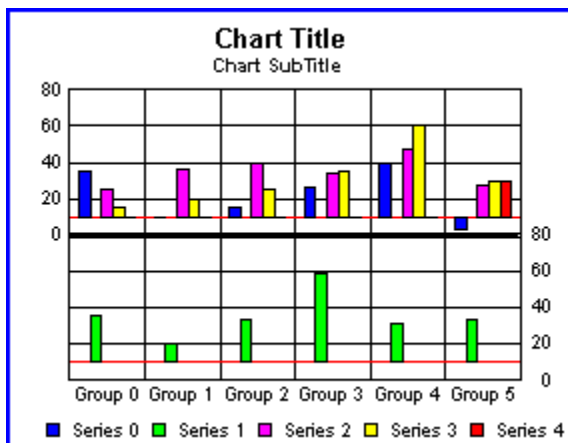
where:

id

Is an object ID returned by a `getObject()` method (for example, `getY1Axis()`).

Example:

```
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setNonZeroBaseline(getY1Axis(), true);
setNonZeroBaselineValue(getY1Axis(), 10);
setNonZeroBaseline(getY2Axis(), getNonZeroBaseline(getY1Axis()));
setNonZeroBaselineValue(getY2Axis(),
getNonZeroBaselineValue(getY1Axis()));
setFillColor(getNonZeroBaseLineZeroLine(), new Color (255,0,0));
```



See [getNonzeroBaselineValue\(\)](#) on page 782, [setNonZeroBaseline\(\)](#) on page 944, [setNonZeroBaselineValue\(\)](#) on page 945.

getNonzeroBaselineValue()

This method returns the non-zero baseline value set by `setNonZeroBaselineValue()` for a numeric axis.

Syntax:

```
double getNonZeroBaselineValue(IdentObj id)
```

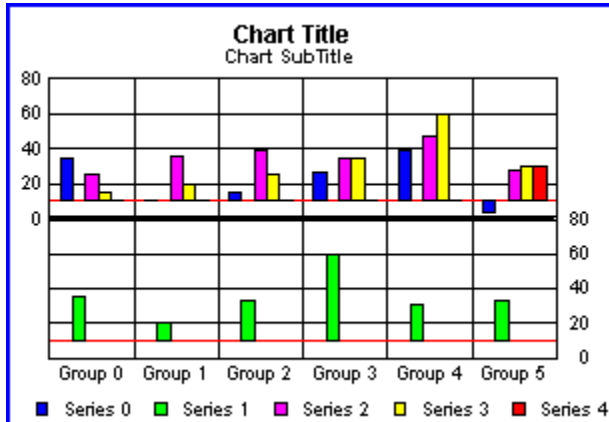
where:

id

Is an object ID returned by a `getObject()` method (for example, `getY1Axis()`).

Example:

```
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setNonZeroBaseline(getY1Axis(),true);
setNonZeroBaselineValue(getY1Axis(),10);
setNonZeroBaseline(getY2Axis(), getNonZeroBaseline(getY1Axis()));
setNonZeroBaselineValue(getY2Axis(),
getNonZeroBaselineValue(getY1Axis()));
setFillColor(getNonZeroBaselineZeroLine(), new Color (255,0,0));
```



See [getNonZeroBaseline\(\)](#) on page 781, [setNonZeroBaseline\(\)](#) on page 944, [setNonZeroBaselineValue\(\)](#) on page 945.

getNumNonIgnoredSeries()

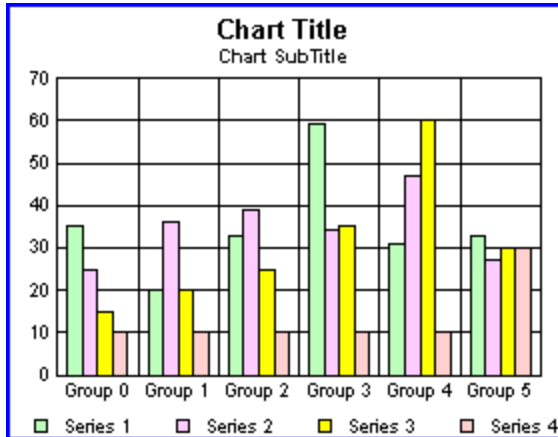
This method returns an integer value that identifies the number of series in the graph that have not been ignored by `setIgnoreSeries()`.

Syntax:

```
int getNumNonIgnoredSeries();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setIgnoreSeries(getSeries(0), true);
setIgnoreSeries(getNumNonIgnoredSeries(), true);
```



See [setIgnoreSeries\(\)](#) on page 915.

getNumColumnsInGroup()

This method returns the number of groups required per data column for a particular graph type.

Syntax:

```
int getNumColumnsInGroup(int nIntegerGraphType)
```

```
int getNumColumnsInGroup()
```

where:

nIntegerGraphType

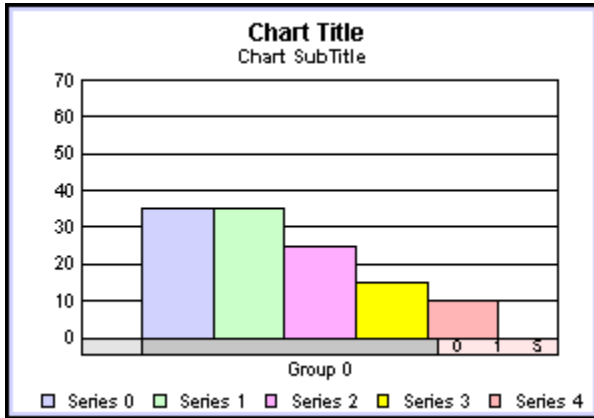
Is a graph type number.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setVisibleGroups(getNumColumnsInGroup(17));

```



Note: If a graph type is not specified with `nIntegerGraphType`, this method returns the number of groups required per data column for the current graph type.

See [getMinNumGroups\(\)](#) on page 779, [getMinNumSeries\(\)](#) on page 780.

getNumGroups()

This method returns an integer value that identifies the number of viewable groups in a graph. The number of viewable groups is the number of groups that are visible in a graph after data scrolling and/or zooming. Use `getNumTotalGroups()` to determine the total number of groups in a graph's data set.

Syntax:

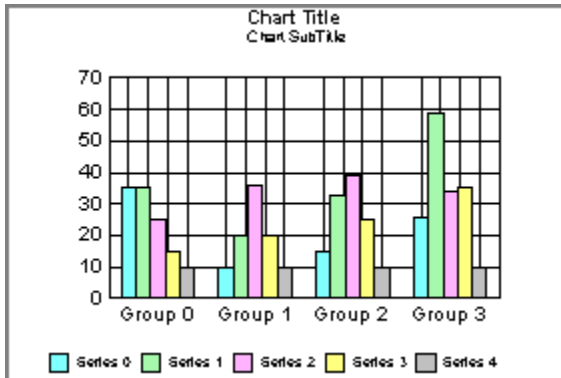
```
int getNumGroups();
```

Example:

```

setViewableGroups(4);
setDataScrollerPresenceGroups(0);
set01MinorGridCount(getNumGroups());
set01MinorGridDisplay(true);
set01MajorGridDisplay(false);
setDepthRadius(0);
setDepthAngle(0);

```



See [getNumTotalGroups\(\)](#) on page 786, [getNumSeries\(\)](#) on page 785, [getNumTotalSeries\(\)](#) on page 787.

getNumSeries()

This method returns an integer value that identifies the number of viewable series in a graph. The number of viewable series is the number of series that are visible in a graph after any data scrolling and/or zooming. Use [getNumTotalSeries\(\)](#) to determine the total number of series in a graph's data set.

Syntax:

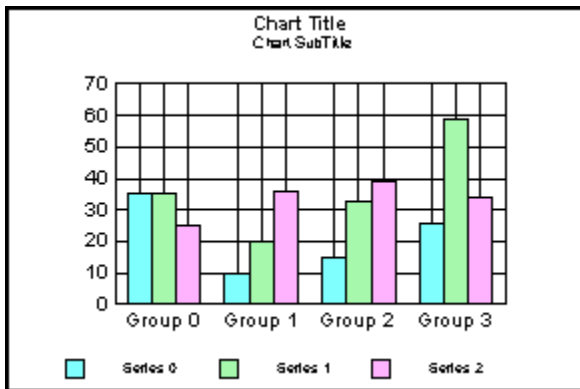
```
int getNumSeries();
```

Example:

```

setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setO1MinorGridCount(getNumSeries());
setO1MinorGridDisplay(true);
setO1MajorGridDisplay(false);
setDepthRadius(0);
setDepthAngle(0);

```



See [getNumGroups\(\)](#) on page 784, [getNumTotalGroups\(\)](#) on page 786, [getNumTotalSeries\(\)](#) on page 787.

getNumTotalGroups()

This method returns an integer value that identifies the total number of groups in a graph's entire data range. It does not consider any data scrolling or zooming operations that may have been applied to a graph. Use [getNumGroups\(\)](#) if you want to get the number of viewable groups in a graph (after any data scrolling or zooming).

Syntax:

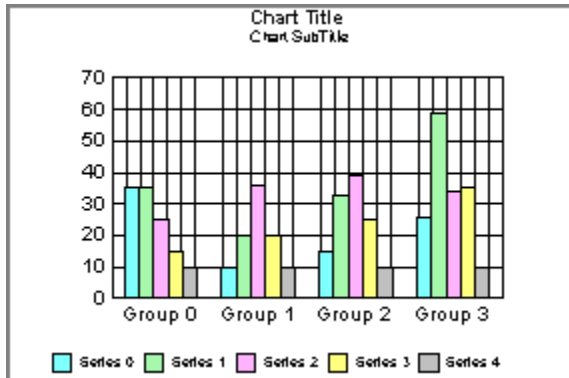
```
int getNumTotalGroups();
```

Example:

```

setViewableGroups(4);
setDataScrollerPresenceGroups(0);
set01MinorGridCount(getNumTotalGroups\(\));
set01MinorGridDisplay(true);
set01MajorGridDisplay(false);
setDepthRadius(0);
setDepthAngle(0);

```



See [getNumGroups\(\)](#) on page 784, [getNumSeries\(\)](#) on page 785, [getNumTotalSeries\(\)](#) on page 787.

getNumTotalSeries()

This method returns an integer value that identifies the total number of series in a graph's entire data range. It does not consider any data scrolling or zooming operations that may have been applied to a graph. Use [getNumSeries\(\)](#) if you want to get the number of viewable series in a graph (after any data scrolling or zooming).

Syntax:

```
int getNumTotalSeries\(\);
```

See [getNumSeries\(\)](#) on page 785, [getNumGroups\(\)](#) on page 784, [getNumTotalGroups\(\)](#) on page 786.

get01MajorGridColorBand1()

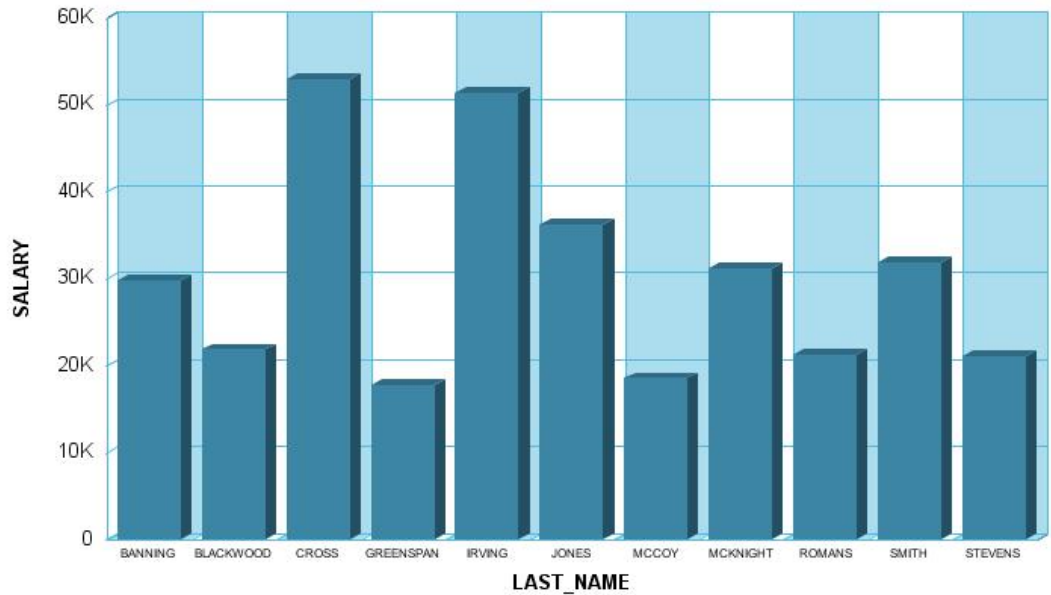
This method returns the object ID of the X-axis Color Band 1, which allows color banded styling to be applied to the X-axis grid in a graph.

Syntax:

```
IdentObj get01MajorGridColorBand1\(\)
```

Example:

```
setDisplay(get01MajorGridColorBand1(),true);  
setFillColor(get01MajorGridColorBand1(), new Color (171,221,239));
```

**get01MajorGridColorBand2()**

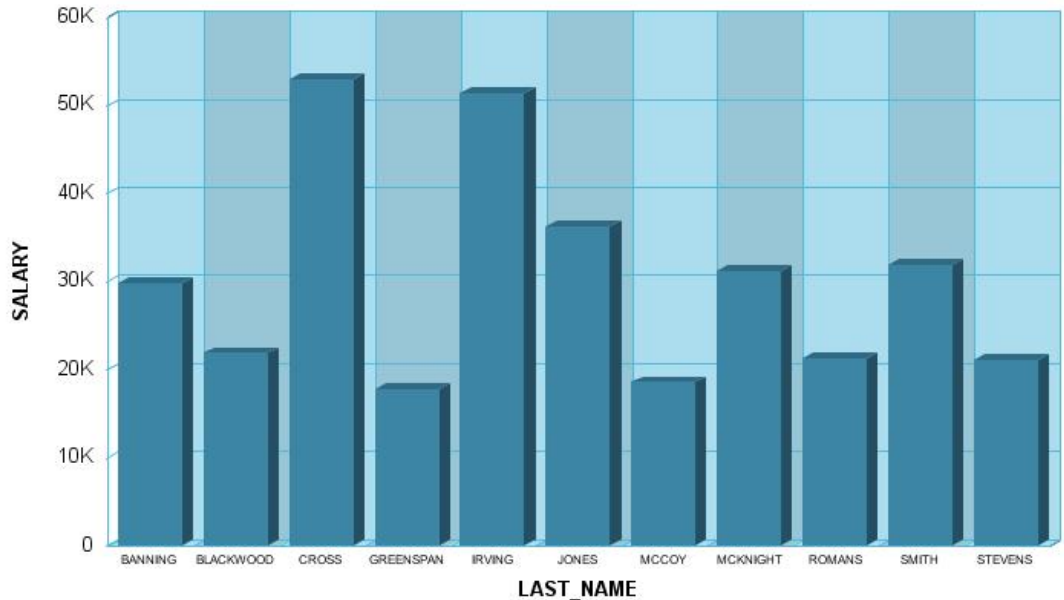
This method returns the object ID of the X-axis Color Band 2, which allows color banded styling to be applied to the X-axis grid in a graph.

Syntax:

```
IdentObj get01MajorGridColorBand2()
```


Example:

```
setDisplay(get01MajorGridColorBand1(),true);
setFillColor(get01MajorGridColorBand1(), new Color (171,221,239));
setDisplay(get01MajorGridColorBand2(),true);
setFillColor(get01MajorGridColorBand2(), new Color (153,198,214));
```

**getPieSliceDelete()**

This method returns a boolean that identifies whether or not a pie slice has been deleted from a pie graph.

Syntax:

```
boolean getPieSliceDelete(IdentObj id);
boolean getPieSliceDelete(int seriesID);
boolean getPieSliceDelete(int seriesID,int groupID);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

seriesID

Is a series number.

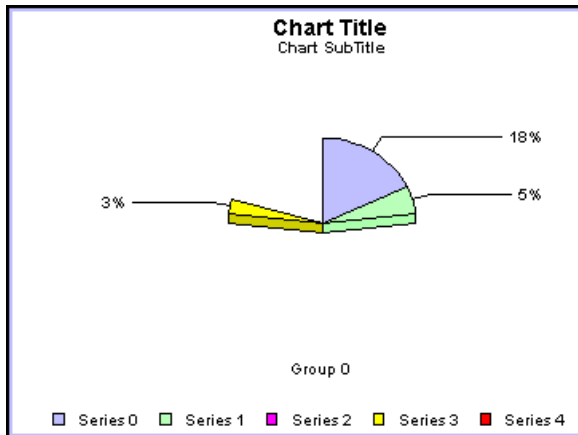
groupID

Is a group number.

Returned values can be true (the slice has been deleted) or false (the slice has not been deleted).

Example:

```
setGraphType(55);
setPieSliceDelete(4,0,true);
setPieSliceDelete(2,0,getPieSliceDelete(4,0));
```



See [getPieSliceDelete\(\)](#) on page 789, [getPieSliceDetach\(\)](#) on page 790, [setPieSliceDetach\(\)](#) on page 949.

getPieSliceDetach()

This method returns an integer value that identifies the distance (0...100) a slice is detached from a pie graph.

Syntax:

```
int getPieSliceDetach(IdentObj id);
int getPieSliceDetach(int seriesID);
int getPieSliceDetach(int seriesID, int groupID);
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

seriesID

Is a series number.

groupID

Is a group number.

See [getPieSliceDelete\(\)](#) on page 789, [setPieSliceDelete\(\)](#) on page 948.

getPlaceAlign()

This method returns the horizontal alignment set by [setPlaceAlign\(\)](#) for a specified object.

Syntax:

```
int getPlaceAlign(IdentObj id)
```

where:

id

Is an object ID of a text object returned by a [getObject\(\)](#) method (for example, [getO1Label\(\)](#)).

Returned values can be 0 (do not allow horizontal alignment of the specified graph object) or 1 (allow horizontal alignment of the specified graph object).

Note:

- ❑ All [getPlace...\(\)](#) and [setPlace...\(\)](#) methods are only valid when the automatic graph layout manager has been enabled with [setPlace\(true\)](#).
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that realignment is needed to produce a better graph layout.

See [setPlaceAlign\(\)](#) on page 950, [Place](#) on page 496.

getPlaceMargin()

This function returns a rectangle that defines the margin around a graph applied by [setPlaceMargin\(\)](#). The default margin is new Rectangle (-15500, -15500, 24000, 31000).

Syntax:

```
Rectangle getPlaceMargin()
```

Note: All [getPlace...\(\)](#) and [setPlace...\(\)](#) methods are only valid when the automatic graph layout manager has been enabled with [setPlace\(true\)](#).

See [Place](#) on page 496, [setPlaceMargin\(\)](#) on page 951.

getPlaceMaxNumWordWrapLines()

This method returns the maximum number of word wrap lines set by `setPlaceMaxNumWordWrapLines()` for a specified text object.

Syntax:

```
int getPlaceMaxNumWordWrapLines(IdentObj id)
```

where:

id

Is an object ID of a text object returned by a `getObject()` method (for example, `getO1Label()`).

This method returns the maximum number of word wrap lines (0...10). The default value is 3.

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that word wrapping is needed to produce a better graph layout.

See [setPlaceMaxNumWordWrapLines\(\)](#) on page 951, [Place](#) on page 496.

getPlaceMinFontSize()

This method returns the minimum font point size set by `setPlaceMinFontSize()` for a specified text object.

Syntax:

```
int getPlaceMinFontSize(IdentObj id)
```

where:

id

Is an object ID of a text object returned by a `getObject()` method (for example, `getO1Label()`).

The minimum font size can be a value from 4...96. The default value is 4.

Note:

- ❑ All `getPlace...`() and `setPlace...`() methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that font resizing is needed to produce a better graph layout.

See [setPlaceMinFontSize\(\)](#) on page 952, [Place](#) on page 496.

getPlaceRect()

This method returns the rectangle set by `setPlaceRect()` for a specified object. In the current implementation, only object IDs return by `getPlaceVertO1LabelsTitle()` or `getPlaceHorzO1LabelsTitle()` can be used.

Syntax:

```
Rectangle getPlaceRect(IdentObj id)
```

where:

id

Is an object ID returned by `getPlaceVertO1LabelsTitle()` or `getPlaceHorzO1LabelsTitle()`.

Note: All `getPlace...`() and `setPlace...`() methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.

See [setPlaceRect\(\)](#) on page 954, [Place](#) on page 496.

getPlaceResize()

This method returns the resizing value set by `setPlaceResize()` for a specified object.

Syntax:

```
int getPlaceResize(IdentObj id)
```

where:

id

Is the object ID of a graph object returned by a `getObject()` method (for example, `getO1Label()`).

Returned values can be 0 (do not allow resizing of graph objects) or 1 (allow resizing of graph objects).

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that resizing is needed to produce a better graph layout.

See [setPlaceResize\(\)](#) on page 954, [Place](#) on page 496.

getPlaceSkip()

This method returns the label skip value set by `setPlaceSkip()` for labels on the primary ordinal axis.

Syntax:

```
int getPlaceSkip(IdentObj id)
```

where:

id

Is the object ID of primary ordinal axis labels (for example, `getO1Label()`).

Returned values can be 0 (do not allow label skipping) or 1 (allow label skipping).

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that label skipping is needed to produce a better graph layout.

See [setPlaceSkip\(\)](#) on page 957, [Place](#) on page 496.

getPlaceTruncateCount()

This method returns the truncate count (0...1096) set by `setPlaceTruncateCount()` for a specified text object. The default value is 0.

Syntax:

```
int getPlaceTruncateCount(IdentObj id)
```

where:

id

Is an object ID of a text object returned by a `getObject()` method (for example, `getO1Label()`).

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that truncation is needed to produce a better graph layout.

See [setPlaceTruncateCount\(\)](#) on page 957, [Place](#) on page 496.

getPlaceWordWrap()

This method returns the value set by `setPlaceWordWrap()` for a specified text object.

Syntax:

```
int getPlaceWordWrap(IdentObj id)
```

where:

id

Is an object ID of a text object returned by a `getObject()` method (for example, `getO1Label()`).

Returned values can be 0 (do not allow word wrapping) or 1 (allow word wrapping). The default value is 1.

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that word wrapping is needed to produce a better graph layout.

See [setPlaceWordWrap\(\)](#) on page 959, [Place](#) on page 496.

getQuadrantLineValueX()

This method returns the scale value where the vertical quadrant line identified by `nIndex` is located. If the returned value is `NO_LINEVALUE`, then automatic scaling is enabled.

Syntax:

```
double getQuadrantLineValueX(int nIndex)
```

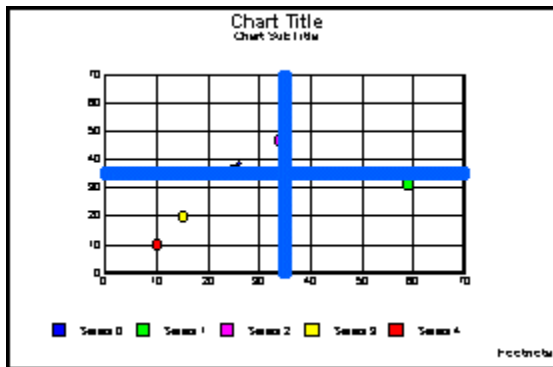
where:

nIndex

Is a quadrant line number (zero to number of quadrant lines drawn).

Example:

```
setGraphType(89);
setGridStep(getX1MajorGrid(),10.0);
setGridStep(getY1MajorGrid(),10.0);
setQuadrantLineValueY(0, getQuadrantLineValueX(0));
```

**getQuadrantLineValueY()**

This method returns the scale value where the horizontal quadrant line identified by *nIndex* is located. If the returned value is NO_LINEVALUE, then automatic scaling is enabled.

Syntax:

```
double getQuadrantLineValueY(int nIndex)
```

where:

nIndex

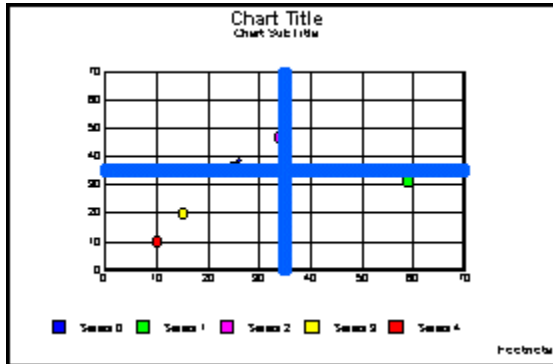
Is a quadrant line number (zero to number of quadrant lines drawn).

Example:

```

setGraphType(89);
setGridStep(getX1MajorGrid(),10.0);
setGridStep(getY1MajorGrid(),10.0);
setQuadrantLineValueX(0, getQuadrantLineValueY(0));

```

**getRect()**

This method returns the bounding rectangle for a user-movable object in virtual coordinates (x, y, width, height). See [Virtual Coordinate System](#) on page 47 for a description of virtual coordinates and a list of user-movable objects.

Syntax:

```
Rectangle getRect(IdentObj id);
```

where:

id

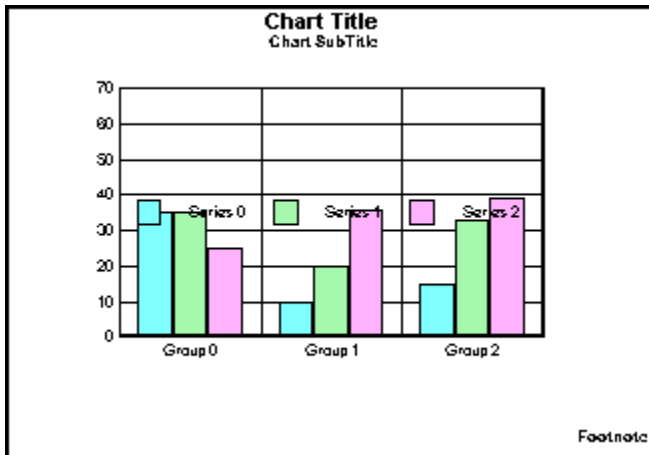
Is an object ID returned by a getObject() method (for example, getFrame()).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setLegendOrient(1);
setLegendRect(getRect(getFrame()));

```



See [setRect\(\)](#) on page 963, [getFrameRect\(\)](#) on page 754, [getLegendRect\(\)](#) on page 765, [setLegendRect\(\)](#) on page 921.

getReferenceLineValue()

This method returns the location/value at which a specified reference line (*nIndex*) is drawn on a specified axis (*axisObj*).

Syntax:

```
double getReferenceLineValue(IdentObj axisObj, int nIndex);
```

where:

axisObj

Is an object ID returned by a `getObject()` method (for example, `getY1Axis()`).

nIndex

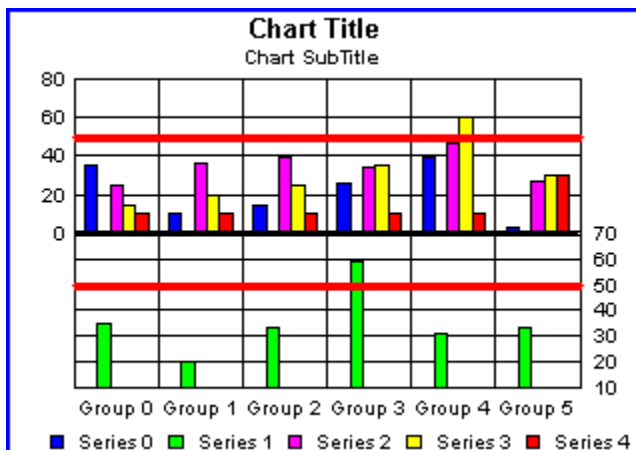
Is a reference line index number (0...2).

Example:

```

setGraphType(21);
setDepthRadius(0);
setReferenceLine(getY1Axis(),0,50);
setLineWidth(getReferenceLine(getY1Axis(),0),3);
setFillColor(getReferenceLine(getY1Axis(),0),
    new Color(255,0,0));
setReferenceLine(getY2Axis(), 1,
    getReferenceLineValue(getY1Axis(), 0));
setLineWidth(getReferenceLine(getY2Axis(),1),3);
setFillColor(getReferenceLine(getY2Axis(),1),
    new Color(255,0,0));

```



See [setReferenceLine\(\)](#) on page 964.

getReferenceLineXY()

This method returns the object ID of a XY reference line. The XY reference line is defined by the [setX1\(\)](#), [setY1\(\)](#), [setX2\(\)](#), and [setY2\(\)](#) methods.

Syntax:

```
IdentObj getReferenceLineXY(IdentObj XYObj, int nIndex);
```

where:

XYObj

Is the XY object ID.

nIndex

Is the integer reference line index number.

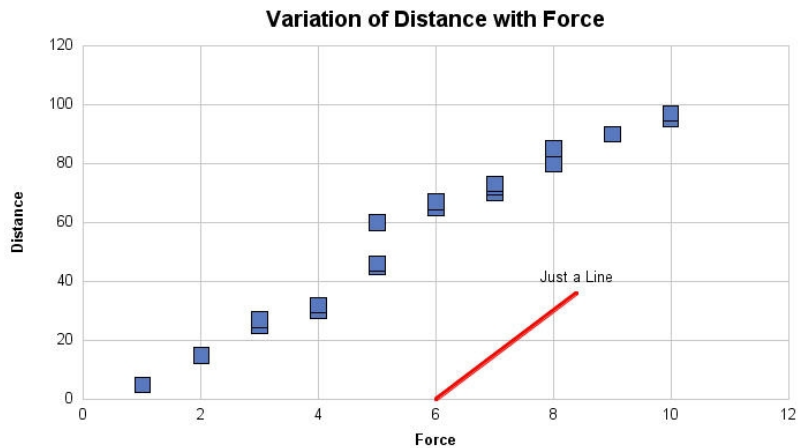
Example:

```

setBorderColor(getReferenceLineXY(0),new Color(255,0,0));
setDisplay(getReferenceLineXY(0),true);
setFillColor(getReferenceLineXY(0),new Color(255,0,0));
setLineWidth(getReferenceLineXY(0),3); setX1(getReferenceLineXY(0), 0.5);
setY1(getReferenceLineXY(0), 0); setX2(getReferenceLineXY(0), 0.7);
setY2(getReferenceLineXY(0),
0.3);setReferencePercentageMode(getReferenceLineXY(0),true);

setDisplay(getReferenceLineXYText(0), true);
setReferenceTextPosition(getReferenceLineXYText(0), 0);
setTextString(getReferenceLineXYText(0), "Just a Line");

```



See [setBorderColor\(\)](#) on page 860, [setDisplay\(\)](#) on page 883, [setFillColor\(\)](#) on page 890, [setLineWidth\(\)](#) on page 937, [setTextString\(\)](#) on page 1002, [ReferenceTextPosition](#) on page 502, [ReferencePercentageMode](#) on page 501, [setX1\(\)](#) on page 1029, [setY1\(\)](#) on page 1030, [setX2\(\)](#) on page 1030, [setY2\(\)](#) on page 1031.

getReferenceMarker()

This method returns the object ID of a custom marker. The XY reference marker is defined by the [setX1\(\)](#) and [setY1\(\)](#) methods.

Syntax:

```
IdentObj getReferenceMarker(IdentObj XYObj, int nIndex);
```

where:

XYObj

Is the marker object ID.

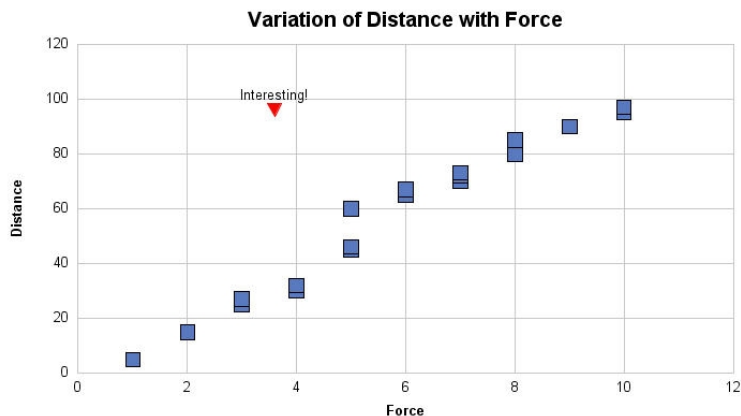
nIndex

Is the integer reference marker index number.

Example:

```
setBorderColor(getReferenceMarker(0),new Color(255,0,0));
setDisplay(getReferenceMarker(0),true);
setFillColor(getReferenceMarker(0),new Color(255,0,0));
setReferenceMarkerShape(getReferenceMarker(0),5);
setReferenceMarkerSize(getReferenceMarker(0), 15);
setX1(getReferenceMarker(0), .3);
setY1(getReferenceMarker(0), .8);setReferencePercentageMode(getReferenceMarker(0),
true);

setDisplay(getReferenceMarkerText(0), true);
setReferenceTextPosition(getReferenceMarkerText(0), 0);
setTextString(getReferenceMarkerText(0), "Interesting!");
```



See [setBorderColor\(\)](#) on page 860, [setDisplay\(\)](#) on page 883, [setFillColor\(\)](#) on page 890, [setTextString\(\)](#) on page 1002, [ReferenceTextPosition](#) on page 502, [ReferencePercentageMode](#) on page 501, [setX1\(\)](#) on page 1029, [setY1\(\)](#) on page 1030.

getReferenceRect()

This method returns the object ID of a custom rectangle. The reference rectangle is defined by the [setX1\(\)](#), [setY1\(\)](#), [setX2\(\)](#), and [setY2\(\)](#) methods.

Syntax:

```
IdentObj getReferenceRect(IdentObj XYObj, int nIndex);
```

where:

XYObj

Is the rectangle object ID.

nIndex

Is the integer reference rectangle index number.

Example:

```
setBorderColor(getReferenceRect(0),new Color(255,111,111));
setDisplay(getReferenceRect(0),true);
setFillColor(getReferenceRect(0),new Color(255,111,111));
setX1(getReferenceRect(0),
0.0); setY1(getReferenceRect(0), 0.0); setX2(getReferenceRect(0),
0.5); setY2(getReferenceRect(0),
0.5);setReferencePercentageMode(getReferenceRect(0),true);

setBorderColor(getReferenceRect(1),new Color(255,255,111));
setDisplay(getReferenceRect(1),true);
setFillColor(getReferenceRect(1),new Color(255,255,111));
setX1(getReferenceRect(1),
0.0); setY1(getReferenceRect(1), 0.5); setX2(getReferenceRect(1),
0.5); setY2(getReferenceRect(1),
1);setReferencePercentageMode(getReferenceRect(1),true);

setBorderColor(getReferenceRect(2),new Color(255,0,0));
setDisplay(getReferenceRect(2),true);
setFillColor(getReferenceRect(2),new Color(255,0,0));
setX1(getReferenceRect(2),
0.5); setY1(getReferenceRect(2), 0); setX2(getReferenceRect(2),
1); setY2(getReferenceRect(2),
0.5);setReferencePercentageMode(getReferenceRect(2),true);

setBorderColor(getReferenceRect(3),new Color(111,255,111));
setDisplay(getReferenceRect(3),true);
setFillColor(getReferenceRect(3),new Color(111,255,111));
setX1(getReferenceRect(3),
0.5); setY1(getReferenceRect(3), 0.5); setX2(getReferenceRect(3),
1); setY2(getReferenceRect(3),
1);setReferencePercentageMode(getReferenceRect(3),true)

setDisplay(getReferenceRectText(0), true);
setReferenceTextPosition(getReferenceRectText(0), 6);
setTextString(getReferenceRectText(0), "Custom Peach ");
```

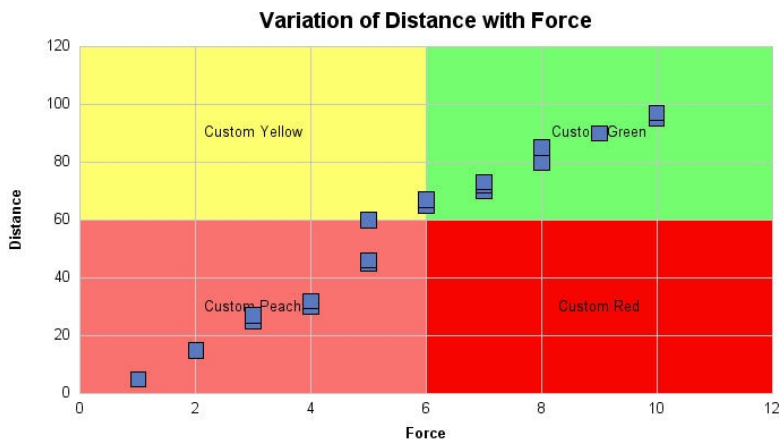
```

setDisplay(getReferenceRectText(1), true);
setReferenceTextPosition(getReferenceRectText(1), 6);
setTextString(getReferenceRectText(1), "Custom Yellow");

setDisplay(getReferenceRectText(2), true);
setReferenceTextPosition(getReferenceRectText(2), 6);
setTextString(getReferenceRectText(2), "Custom Red");

setDisplay(getReferenceRectText(3), true);
setReferenceTextPosition(getReferenceRectText(3), 6);
setTextString(getReferenceRectText(3), "Custom Green");

```



See [setBorderColor\(\)](#) on page 860, [setDisplay\(\)](#) on page 883, [setFillColor\(\)](#) on page 890, [setTextString\(\)](#) on page 1002, [ReferenceTextPosition](#) on page 502, [ReferencePercentageMode](#) on page 501, [setX1\(\)](#) on page 1029, [setY1\(\)](#) on page 1030, [setX2\(\)](#) on page 1030, [setY2\(\)](#) on page 1031.

getScaleMax()

This method returns the maximum scale value assigned to a numeric axis.

Syntax:

```
double getScaleMax(IdentObj id);
```

where:

id

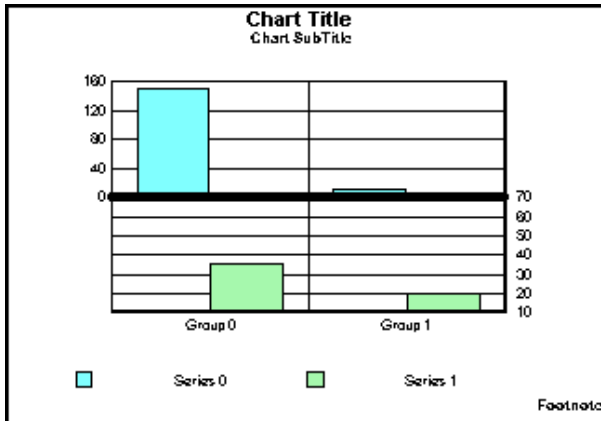
Is an object ID returned by a `getObject()` method (for example, `getY1Axis()`).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setViewableGroups(2);
setViewableSeries(2);
setGraphType(21);
setDataScrollerPresenceGroups(0);
setData(0,0,setScaleMax(getY1Axis()));

```



Note: You may also use one of the axis-specific properties to get the maximum scale on a particular axis (for example, `X1ScaleMax`, `Y1ScaleMax`, `Y2ScaleMax`, and so on).

See [setScaleMax\(\)](#) on page 968, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMaxAuto\(\)](#) on page 969, [getScaleMin\(\)](#) on page 805, [setScaleMin\(\)](#) on page 970, [getScaleMinAuto\(\)](#) on page 806, [setScaleMinAuto\(\)](#) on page 971, [X1ScaleMax](#) on page 608, [X1ScaleMaxAuto](#) on page 610, [Y1ScaleMax](#) on page 643, [Y1ScaleMaxAuto](#) on page 645, [Y2ScaleMax](#) on page 682, [Y2ScaleMaxAuto](#) on page 683.

getScaleMaxAuto()

This method returns a boolean that indicates whether automatic or manual scaling is used for the maximum value on a numeric axis. Returned values can be true (the maximum value is automatically calculated (automatic scaling)) or false (the maximum value is not automatically calculated (manual scaling)).

Syntax:

```
boolean getScaleMaxAuto(IdentObj id);
```


where:

id

Is an object ID returned by a getObject() method (for example, getY1Axis()).

Note: You may also use one of the axis-specific properties to determine automatic/manual scaling on a particular axis (for example, X1ScaleMaxAuto, Y1ScaleMaxAuto, and so on).

See [getScaleMax\(\)](#) on page 803, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969, [getScaleMin\(\)](#) on page 805, [setScaleMin\(\)](#) on page 970, [getScaleMinAuto\(\)](#) on page 806, [setScaleMinAuto\(\)](#) on page 971, [X1ScaleMax](#) on page 608, [X1ScaleMaxAuto](#) on page 610, [Y1ScaleMax](#) on page 643, [Y1ScaleMaxAuto](#) on page 645, [Y2ScaleMax](#) on page 682, [Y2ScaleMaxAuto](#) on page 683.

getScaleMin()

This method returns the minimum scale value assigned to a numeric axis.

Syntax:

```
double getScaleMin(IdentObj id);
```

where:

id

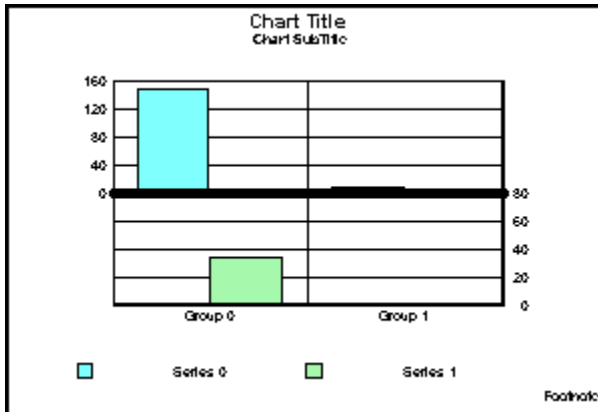
Is an object ID returned by a getObject() method (for example, getY1Axis()).

Example:

```

setDataScrollerPresenceGroups(0);
setDepthRadius(0);
setViewableGroups(2);
setViewableSeries(2);
setGraphType(21);
setData(0,0,getScaleMax());
setData(1,1,setScaleMin(getY1Axis()));

```



Note: You may also use one of the axis-specific properties to get the minimum scale on a particular axis (for example, `X1ScaleMin()`, `Y1ScaleMin()`, `Y2ScaleMin()`, and so on).

See [setScaleMin\(\)](#) on page 970, [getScaleMax\(\)](#) on page 803, [setScaleMax\(\)](#) on page 968, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMaxAuto\(\)](#) on page 969, [getScaleMinAuto\(\)](#) on page 806, [setScaleMinAuto\(\)](#) on page 971, [X1ScaleMin](#) on page 611, [X1ScaleMinAuto](#) on page 613, [Y1ScaleMin](#) on page 646, [Y1ScaleMinAuto](#) on page 648, [Y2ScaleMin](#) on page 685, [Y2ScaleMinAuto](#) on page 686.

getScaleMinAuto()

This method returns a boolean that identifies whether manual or automatic scaling is used for the minimum value on a numeric axis. Returned values can be true (the minimum value is automatically calculated) or false (the minimum value is not automatically calculated).

Syntax:

```
boolean getScaleMinAuto(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getY1Axis()`).

Note: You may also use one of the axis-specific properties to determine automatic/manual scaling on a particular axis (for example, [X1ScaleMinAuto\(\)](#), [Y1ScaleMinAuto\(\)](#), [Y2ScaleMinAuto\(\)](#), and so on).

See [setScaleMinAuto\(\)](#) on page 971, [getScaleMax\(\)](#) on page 803, [setScaleMax\(\)](#) on page 968, [getScaleMaxAuto\(\)](#) on page 804, [setScaleMaxAuto\(\)](#) on page 969, [getScaleMin\(\)](#) on page 805, [setScaleMin\(\)](#) on page 970, [X1ScaleMin](#) on page 611, [X1ScaleMinAuto](#) on page 613, [Y1ScaleMin](#) on page 646, [Y1ScaleMinAuto](#) on page 648, [Y2ScaleMin](#) on page 685, [Y2ScaleMinAuto](#) on page 686.

getScaleMustIncludeZero()

This method returns a boolean that identifies whether or not a given axis must include zero. Returned values can be true (the axis must include zero) or false (zero is not required on this axis).

Syntax:

```
boolean getScaleMustIncludeZero(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1Axis()).

Note: You may also use one of the axis-specific properties to determine if a particular axis must include zero (for example, [X1MustIncludeZero](#), [Y1MustIncludeZero](#), [Y2MustIncludeZero](#), and so on).

See [setScaleMustIncludeZero\(\)](#) on page 972, [X1MustIncludeZero](#) on page 605, [Y1MustIncludeZero](#) on page 641, [Y2MustIncludeZero](#) on page 679.

getSeriesBorderColor()

This method returns the border color of a particular series in a graph.

Syntax:

```
Color getSeriesBorderColor (int s);
```

where:

s

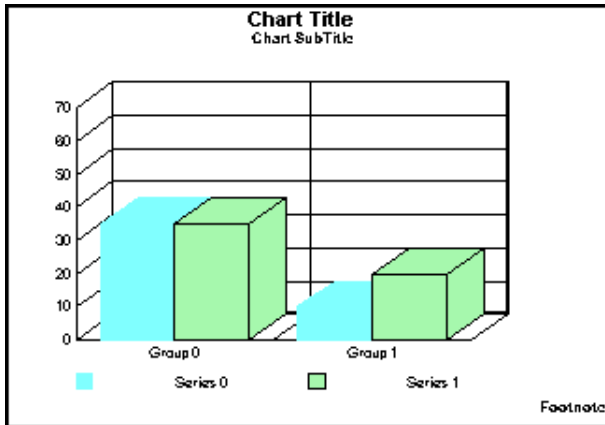
Is a series number.

Example:

```

setViewableGroups(2);
setViewableSeries(2);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setSeriesBorderColor(0,new Color(0,0,239));
setSeriesFillColor(0, getSeriesBorderColor(0));

```



See [getSeriesFillColor\(\)](#) on page 808, [setSeriesBorderColor\(\)](#) on page 976.

getSeriesFillColor()

This method returns the fill color of a particular series in a graph.

Syntax:

```
Color getSeriesFillColor (int s);
```

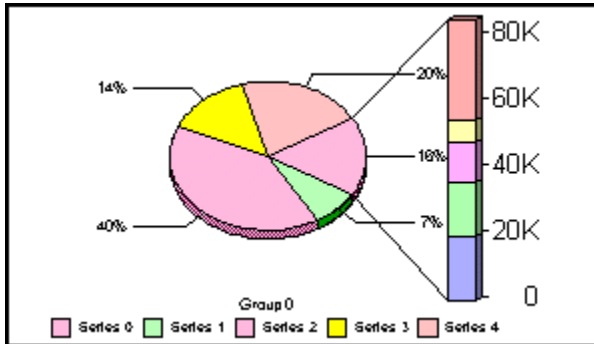
where:

s

Is a series number.

Example:

```
setSeriesFillColor(0, new Color(206,0,121));
setSeriesFillColor(2, getSeriesFillColor(0));
setGraphType(93);
```



See [setSeriesBorderColor\(\)](#) on page 976, [setSeriesFillColor\(\)](#) on page 977.

getSeriesLabel()

This method returns a label string associated with a series in a graph.

Syntax:

```
string getSeriesLabel(IdentObj id);
string getSeriesLabel(int seriesID);
```

where:

seriesID

Is any valid series number.

id

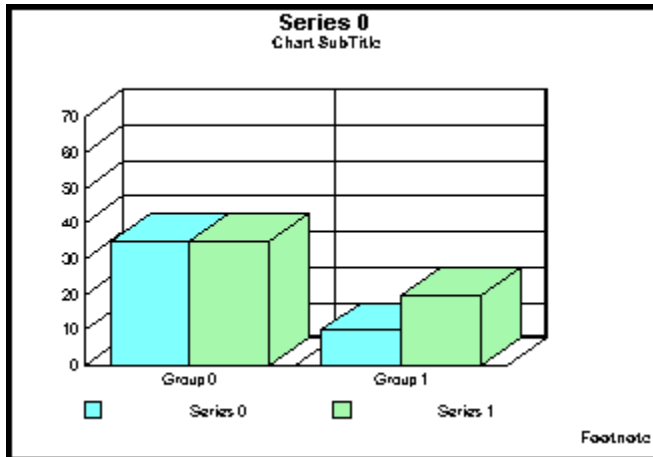
Is an object ID returned by a getObject() method (for example, getSeries()).

Example:

```

setViewableGroups(2);
setViewableSeries(2);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setTitleString(getSeriesLabel(0));

```



See [setSeriesLabel\(\)](#) on page 979.

getSeriesType()

This method returns an integer value that identifies the series type for a specific object or a specific series.

Syntax:

```

int getSeriesType(IdentObj id);
int getSeriesType(int seriesID);

```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

seriesID

Is a series number.

The integer value for the series type can be:

- ❑ 0 (riser unspecified)

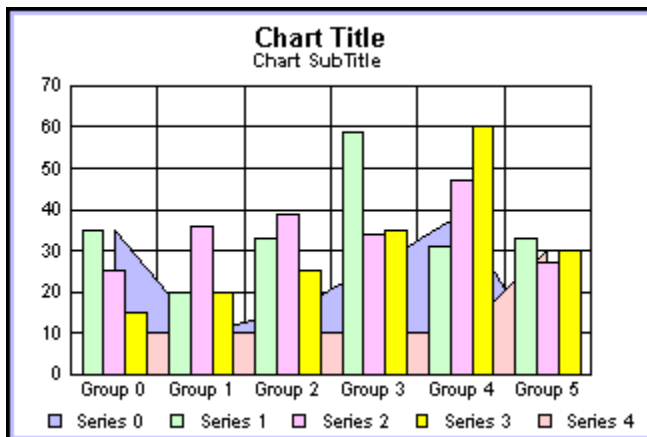
- ❑ 1 (riser bar)
- ❑ 2 (riser marker)
- ❑ 3 (riser area)

Example:

```

setDepthRadius(0);
setDepthAngle(0);
setSeriesType(getSeries(0),3);
setSeriesType(getSeries(4), getSeriesType(getSeries(0)));

```



See [setSeriesType\(\)](#) on page 981.

getShadowColor()

This method returns the color value of a drop shadow. A returned value of null means no drop shadow color has been applied.

Syntax:

```
Color getShadowColor(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getFrame()`).

See [setShadowColor\(\)](#) on page 984, [getShadowDisplay\(\)](#) on page 812, [setShadowDisplay\(\)](#) on page 985, [getShadowXOffset\(\)](#) on page 814, [setShadowXOffset\(\)](#) on page 989, [getShadowYOffset\(\)](#) on page 814, [setShadowYOffset\(\)](#) on page 990.

getShadowDisplay()

This method indicates whether or not a drop shadow is applied to an object in a graph. Returned values can be true (the drop shadow is drawn) or false (the drop shadow is not drawn).

Syntax:

```
boolean getShadowDisplay(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getFrame()).

See [setShadowDisplay\(\)](#) on page 985, [getShadowColor\(\)](#) on page 811, [setShadowColor\(\)](#) on page 984, [getShadowXOffset\(\)](#) on page 814, [setShadowXOffset\(\)](#) on page 989, [getShadowYOffset\(\)](#) on page 814, [setShadowYOffset\(\)](#) on page 990.

getShadowFillType()

This method returns the shadow fill type set by setShadowFillType() for a specified Java2D text object.

Syntax:

```
int getShadowFillType(IdentObj id)
```

where:

id

Is an object ID returned by a getObject() method (for example, getTitle()).

The integer value for the shadow fill type can be:

- ❑ 1 (FILL_COLOR). This value is the default.
- ❑ 2 (FILL_GRADIENT_LEFTRIGHT).
- ❑ 3 (FILL_TEXTURE)

See [setShadowFillType\(\)](#) on page 986.

getShadowGradientEndColor()

This method returns the shadow gradient end color set by setShadowGradientEndColor() for a specified Java2D text object. The default value is Green.

Syntax:

```
Color getShadowGradientEndColor(IdentObj idObj)
```


where:

idObj

Is the object ID returned by a *getObject()* method (for example, *getTitle()*).

See [setShadowGradientEndColor\(\)](#) on page 986.

getShadowGradientStartColor()

This method returns the shadow gradient start color set by *setShadowGradientStartColor()* for a specified text object. The default value is Blue.

Syntax:

`Color getShadowGradientStartColor(IdentObj idObj)`

where:

idObj

Is an object ID returned by a *getObject()* method (for example, *getTitle()*).

See [setShadowGradientStartColor\(\)](#) on page 987.

getShadowTextureURL()

This method returns the texture that has been applied to a Java 2D text object by *setShadowTextureURL()*.

Syntax:

`Image getShadowTextureURL(String szURL)`

This form of this method returns a *java.awt.Image* set by *setShadowTextureURL()* for a specified text object.

`String getShadowTextureURL(IdentObj id)`

This form of this method returns the URL specification string that identifies a texture file, set by *setShadowTextureURL()* for a specified text object.

where:

id

Is the object ID of a text object returned by a *getObject()* method.

See [setShadowTextureURL\(\)](#) on page 988.

getShadowXOffset()

This method returns the offset in the X-direction of a drop shadow that has been applied to an object. If the returned value is null, than no drop shadow has been applied.

Syntax:

```
int getShadowXOffset(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method.

See [setShadowXOffset\(\)](#) on page 989, [getShadowColor\(\)](#) on page 811, [setShadowColor\(\)](#) on page 984, [getShadowDisplay\(\)](#) on page 812, [setShadowDisplay\(\)](#) on page 985, [getShadowYOffset\(\)](#) on page 814, [setShadowYOffset\(\)](#) on page 990.

getShadowYOffset()

This method returns the offset in the Y-direction of a drop shadow that is assigned to an object. If the returned value is null, than no drop shadow has been applied.

Syntax:

```
int getShadowYOffset(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method.

See [setShadowYOffset\(\)](#) on page 990, [getShadowColor\(\)](#) on page 811, [setShadowColor\(\)](#) on page 984, [getShadowDisplay\(\)](#) on page 812, [setShadowDisplay\(\)](#) on page 985, [getShadowXOffset\(\)](#) on page 814, [setShadowXOffset\(\)](#) on page 989.

getSkipBegin()

This method returns the label skip begin value. This value represents the first label to be skipped when manual skip mode is enabled with setO1LabelAutoSkip(2).

Syntax:

```
int getSkipBegin(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getO1Label()).

See [setSkipBegin\(\)](#) on page 991, [getSkipCount\(\)](#) on page 815, [setSkipCount\(\)](#) on page 992, [O1LabelAutoSkip](#) on page 432.

getSkipCount()

This method returns a skip interval value. This value represents the interval of labels to be skipped when manual skip mode is enabled with `setO1LabelAutoSkip(2)`.

Syntax:

```
int getSkipCount(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getO1Label()`).

See [setSkipCount\(\)](#) on page 992, [getSkipBegin\(\)](#) on page 814, [O1LabelAutoSkip](#) on page 432.

getTextBackgroundColor()

This method returns the background color value set by `setTextBackgroundColor()` for a specified text object. The default value is White.

Syntax:

```
Color getTextBackgroundColor(IdentObj idObj)
```

where:

idObj

Is an object ID returned by a `getObject()` method (for example, `getTitle()`).

getTextFormatPattern()

This method can be used to determine the standard Java number format pattern (if any) assigned to a data text or label object.

Syntax:

```
string getTextFormatPattern(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getY1Label()`).

This method returns a pattern string in the following format:

```
pattern:= subpattern{;subpattern}  
subpattern:= {prefix}integer{.fraction}{suffix}  
prefix:= '\\u0000'..'\\uFFFF' - specialCharacters  
suffix:= '\\u0000'..'\\uFFFF' - specialCharacters  
integer:= '#'* '0'* '0'  
fraction:= '0'* '#'*
```

See [Formatting Numbers and Numeric Labels](#) on page 130 for more detailed information about this format pattern.

See [setTextFormatPattern\(\)](#) on page 994, [getTextFormatPreset\(\)](#) on page 816, [setTextFormatPreset\(\)](#) on page 995, [DataTextFormat](#) on page 318, [PieFeelerTextFormat](#) on page 478, [PieRingTotalFormat](#) on page 486, [X1LabelFormat](#) on page 584, [Y1LabelFormat](#) on page 625, [Y2LabelFormat](#) on page 661.

getTextFormatPreset()

This method returns an integer value that identifies the format of a numeric text object. Returned values can be -2, -1 (the format of the number is defined by one of the ...FormatPattern() properties or methods) or 0...21, 24...30 (the format of the number is defined by a preset format). See [Preset Number Formats](#) on page 130.

Syntax:

```
int getTextFormatPreset(IdentObj id);
```

where:

id

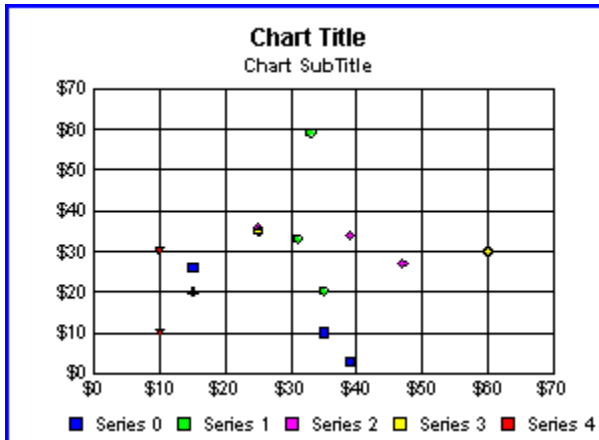
Is an object ID returned by a getObject() method (for example, getY1Label()).

Example:

```

setGraphType(61);
setDepthRadius(0);
setDepthAngle(0);
setTextFormatPreset(getY1Label(), 6);
setTextFormatPreset(getX1Label(), getTextFormatPreset(getY1Label()));

```



See [setTextFormatPreset\(\)](#) on page 995, [getTextFormatPattern\(\)](#) on page 815, [setTextFormatPattern\(\)](#) on page 994.

getTextJustHoriz()

This method returns an integer value that identifies the horizontal alignment of a text object.

Syntax:

```
int getTextJustHoriz(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getY1Label()`).

Returned values can be:

- ☐ 0 (left)
- ☐ 1 (center)
- ☐ 2 (right)

See [setTextJustHoriz\(\)](#) on page 996, [getTextJustVert\(\)](#) on page 818, [setTextJustVert\(\)](#) on page 998.

getTextJustVert()

This method returns an integer value that identifies the vertical alignment of a text object.

Syntax:

```
int getTextJustVert(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1Label()).

Returned values can be:

- ❑ 0 (top)
- ❑ 1 (center)
- ❑ 2 (bottom)

See [setTextJustVert\(\)](#) on page 998, [getTextJustHoriz\(\)](#) on page 817, [setTextJustHoriz\(\)](#) on page 996.

getTextRotation()

This method returns an integer value that identifies the rotation angle of a text object.

Syntax:

```
int getTextRotation(IdentObj id);
```

where:

id

Is an object ID returned by a getObject() method (for example, getY1Label()).

Returned values can be:

- ❑ 0 (none)
- ❑ 1 (90 degrees)
- ❑ 2 (270 degrees)

See [setTextRotation\(\)](#) on page 1000, [O1LabelRotate](#) on page 435, [O2LabelRotate](#) on page 459, [X1LabelRotate](#) on page 586, [Y1LabelRotate](#) on page 627, [Y2LabelRotate](#) on page 663.

getTextStrikethrough()

This method returns the value set by setTextStrikethrough() and identifies whether or not strike-through has been applied to a specified text object.

Returned values can be true (strike-through has been applied to the specified text object) or false (strike-through not applied). The default value is false.

Syntax:

```
boolean getTextStrikethrough(IdentObj idObj)
```

where:

idObj

Is an object ID returned by a *getObject()* method (for example, *getTitle()*).

getTextString()

This method returns a text string assigned to a text object.

Syntax:

```
string getTextString(IdentObj id);
```

where:

id

Is an object ID returned by a *getObject()* method (for example, *getTitle()*).

Example:

```
setDataScrollerPresenceGroups(0);
setDepthRadius(0);
setViewableGroups(3);
setViewableSeries(3);
setTitleString(getTextString(getFootnote()));
```

See [setTextString\(\)](#) on page 1002, [FootnoteString](#) on page 357, [O1TitleString](#) on page 454, [O2TitleString](#) on page 466, [SubtitleString](#) on page 550, [TitleString](#) on page 555, [X1TitleString](#) on page 615, [Y1TitleString](#) on page 650, [Y2TitleString](#) on page 689.

getTexture()

This method can be used to determine if a texture has been applied to an object in a graph. When a texture name (that is, a .GIF file name) is supplied as an input parameter, this method identifies whether or not the named texture is applied to any object in the graph. If an object ID is specified (for example, *getTexture (getSeries(1);)*, this method will identify whether or not a texture is applied to the specific object. A returned value of null means that texture is not in the look.

Syntax:

```
Image getTexture(String szURL);
```

```
String getTexture(IdentObj id);
```

where:

szURL

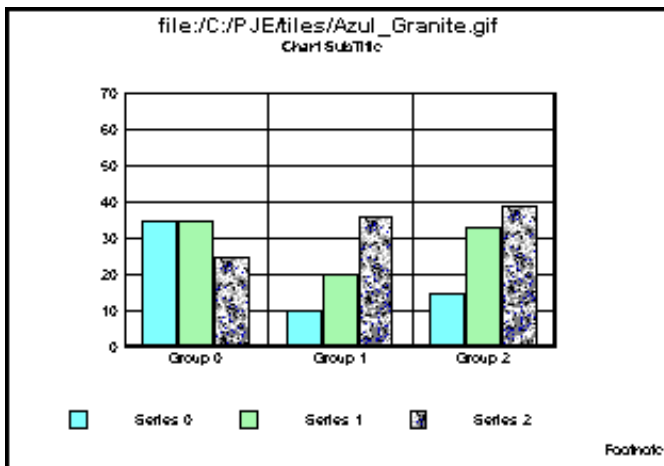
Is the name of an image to look for in the graph.

id

Is an object ID returned by a getObject() method (for example, getSeries()).

Example:

```
setDepthRadius(0);
setFillType(getSeries(2),3);
setTextureDisplayMode(getSeries(2),0);
setTextureURL(getSeries(2),"/images/tdg/Azul_Granite.gif");
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
setTitleString(getTexture(getSeries(2)));
```



See [getTextureDisplayMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004, [getFillType\(\)](#) on page 747, [setFillType\(\)](#) on page 892, [setTextureURL\(\)](#) on page 1007, [DisplayBarAsPictograph](#) on page 342.

getTextureBGColor()

This method returns the texture background color set by setTextureBGColor() for a specified object.

Syntax:

```
Color getTextureBGColor(IdentObj id)
Color getTextureBGColor(int seriesID)
Color getTextureBGColor(int seriesID, int groupID)
```

where:

IdentObj

Is an object ID returned by a getObject() method.

seriesID

Is a series number.

groupID

Is a group number.

See [setTextureBGColor\(\)](#) on page 1003.

getTextureDisplayMode()

This method returns the texture display mode for a specified object. Returned values can be texture display mode 0 (stretched) or 1 (tiled).

Syntax:

```
int getTextureDisplayMode(IdentObj id);
int getTextureDisplayMode()
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

See [getTexture\(\)](#) on page 819, [getTextureDisplayMode\(\)](#) on page 821, [getTextureFlipMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004, [setTextureFlipMode\(\)](#) on page 1005, [setTextureResource\(\)](#) on page 1006, [setTextureURL\(\)](#) on page 1007.

getTextureFlipMode()

This method returns the texture flip mode for a specified object. Returned values are a texture flip mode index number, 0...3. The default value is 0.

Syntax:

```
int getTextureFlipMode(IdentObj id)
int getTextureFlipMode()
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getO1Label()`).

See [getTexture\(\)](#) on page 819, [getTextureDisplayMode\(\)](#) on page 821, [getTextureFlipMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004, [setTextureFlipMode\(\)](#) on page 1005, [setTextureResource\(\)](#) on page 1006, [setTextureURL\(\)](#) on page 1007.

getTextWrap()

This method indicates whether or not line wrapping is enabled for a text object. Returned values can be true (text wrapping is enabled) or false (text wrapping is not enabled).

Syntax:

```
boolean getTextWrap(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getO1Label()`).

See [setTextWrap\(\)](#) on page 1008, [O1LabelWrap](#) on page 441, [O2LabelWrap](#) on page 463.

getTickCount()

This method returns the number of tick marks on the specified axis set by `setTickCount()`.

Syntax:

```
int getTickCount(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method. It must identify a tick mark object (for example, `getY1MajorTick()`).

See [setTickCount\(\)](#) on page 1009.

getTickLength()

This method returns the length of tick marks set by `setTickLength()`, in virtual coordinates.

Syntax:

```
int getTickLength(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method. It must identify a tick mark object (for example, `getY1MajorTick()`)

See [setTickLength\(\)](#) on page 1011.

getTickStep()

This method returns the number of tick steps set by `setTickStep()` on a specified axis.

Syntax:

```
double getTickStep(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method. It must identify a tick mark object (for example, `getY1MajorTick()`).

See [setTickStep\(\)](#) on page 1011.

getTickStepAuto()

This method returns the boolean set by `setTickStepAuto()` for a specified axis. It indicates whether or not automatic calculation of tick steps is enabled.

Syntax:

```
boolean getTickStepAuto(IdentObj id)
```

where:

id

Is an object ID returned by a `getObject()` method. It must identify a tick mark object (for example, `getY1MajorTick()`).

See [setTickStepAuto\(\)](#) on page 1012.

getTickStyle()

This method returns the tick mark style set by `setTickStyle()`.

Syntax:

```
int getTickStyle(IdentObj id)
```

where:

id

Is an object ID returned by a *getObject()* method. It must identify a tick mark object (for example, *getY1MajorTick()*).

Returned values can be:

0 for small tick marks from frame edge outward

1 for small tick marks from frame edge outward

2 for small tick marks from frame edge inward

3 for small tick marks from frame edge outward

4 Small tick marks span across the frame edge

See *setTickStyle()* on page 1013.

getTickWidth()

This method returns the width of tick marks set by *setTickWidth()* in device coordinates.

Syntax:

```
int getTickWidth(IdentObj id)
```

where:

id

Is an object ID returned by a *getObject()* method. It must identify a tick mark object (for example, *getY1MajorTick()*)

See *setTickWidth()* on page 1013.

getTransparentBorderColor()

This method indicates whether or not a transparent border color is applied to an object. Returned values can be true (transparent border color is applied) or false (transparent border color is not applied).

Syntax:

```
boolean getTransparentBorderColor(IdentObj id);
```

where:

id

Is an object ID returned by a *getObject()* method (for example, *getSeries()*).

See [setTransparentBorderColor\(\)](#) on page 1022, [getBorderColor\(\)](#) on page 723, [setBorderColor\(\)](#) on page 860, [setSeriesBorderColor\(\)](#) on page 976.

getTransparentFillColor()

This method indicates whether or not a transparent fill color is applied to an object. Returned values can be true (transparent fill color is applied) or false (transparent fill color is not applied).

Syntax:

```
boolean getTransparentFillColor(IdentObj id);
```

where:

id

Is an object ID returned by a `getObject()` method (for example, `getSeries()`).

See [setTransparentFillColor\(\)](#) on page 1022, [getFillColor\(\)](#) on page 746, [setFillColor\(\)](#) on page 890, [setSeriesFillColor\(\)](#) on page 977.

getUseTextBackgroundColor()

This method returns the value set by `setUseTextBackgroundColor()` for a specified text object. If this method returns true, background color can be applied to the specified text object with `setTextBackgroundColor()`. If this method returns false, background color is disabled for the specified text object. The default value is false.

Syntax:

```
boolean getUseTextBackgroundColor(IdentObj idObj)
```

where:

idObj

Is an object ID returned by a `getObject()` method (for example, `getTitle()`).

getWaterfallGroupMode()

This method returns the mode (normal, subtotal, total, or extra) assigned to a group in a waterfall graph.

Syntax:

```
int getWaterfallGroupMode(int nGroup)
```

where:

nGroup

Is a group number.

Returned values can be one of the following waterfall group mode numbers:

- ☐ 0 (normal)
- ☐ 1 (subtotal)
- ☐ 2 (total)
- ☐ 3 (extra)

See [setWaterfallGroupMode\(\)](#) on page 1027.

getX1MajorGridStepAutoValue()

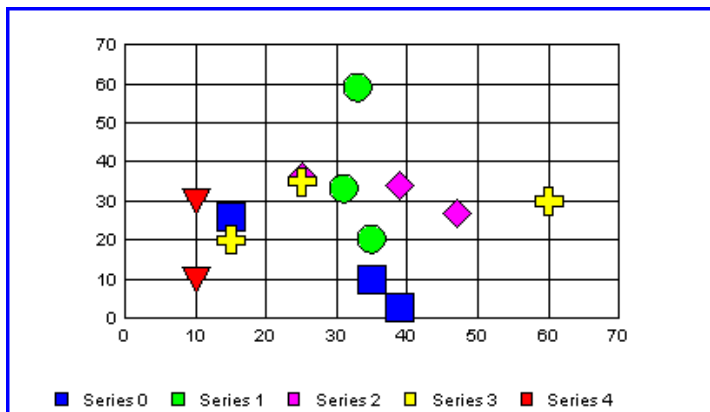
When the X1MajorGridStepAuto property is true (enabling automatic calculation of major grid steps on the X1 axis), this method returns a Java double-precision number that identifies the interval between major grid steps on the X1 axis.

Syntax:

```
double getX1MajorGridStepAutoValue()
```

Example:

```
setGraphType(61);
setDepthRadius(0);
setX1MajorGridStepAuto(false);
setX1MajorGridStepAutoValue(10);
setX1MajorGridStep(getX1MajorGridStepAutoValue());
setMarkerSizeDefault(100);
```



See [X1MajorGridStepAuto](#) on page 594.

getX1ScaleMaxAutoValue()

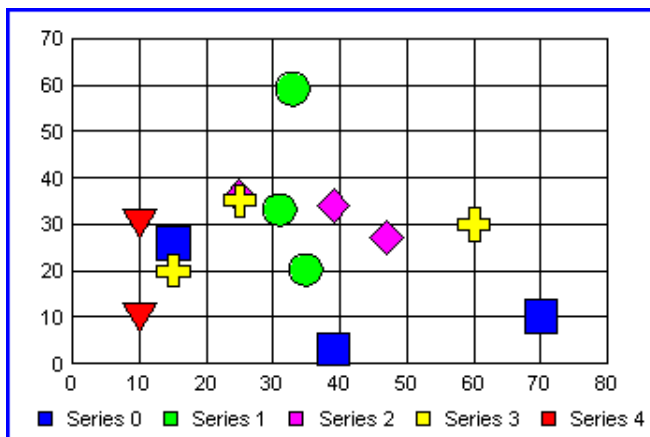
When the X1ScaleMaxAuto property is true (enabling automatic calculation of the maximum scaling value on the X1 axis), this method returns a Java double-precision number that identifies the maximum scale value that will be used on the X1 axis.

Syntax:

```
double getX1ScaleMaxAutoValue()
```

Example:

```
setDepthRadius(0);  
setGraphType(61);  
setMarkerSizeDefault(100);  
setDataValue(getSeriesGroup(0,0),  getX1ScaleMaxAutoValue());
```



See [getX1ScaleMinAutoValue\(\)](#) on page 827, [X1ScaleMaxAuto](#) on page 610.

getX1ScaleMinAutoValue()

When the X1ScaleMinAuto property is true (enabling automatic calculation of the minimum scaling value on the X1 axis), this method returns a Java double-precision number that identifies the minimum scale value that will be used on the X1 axis.

Syntax:

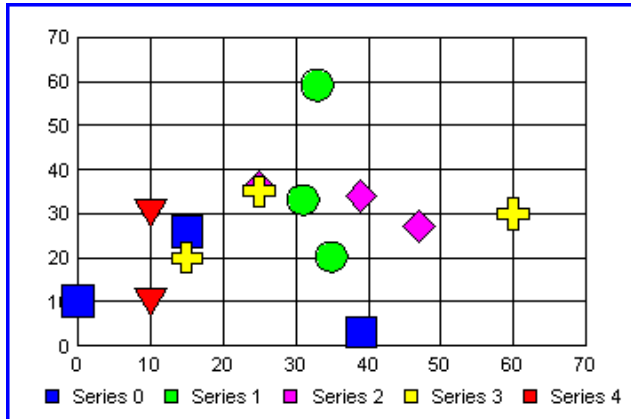
```
double getX1ScaleMinAutoValue()
```

Example:

```

setDepthRadius(0);
setGraphType(61);
setMarkerSizeDefault(100);
setDataValue(getSeriesGroup(0,0), getX1ScaleMinAutoValue());

```



See [getX1ScaleMaxAutoValue\(\)](#) on page 827, [X1ScaleMinAuto](#) on page 613.

getY1MajorGridColorBand1()

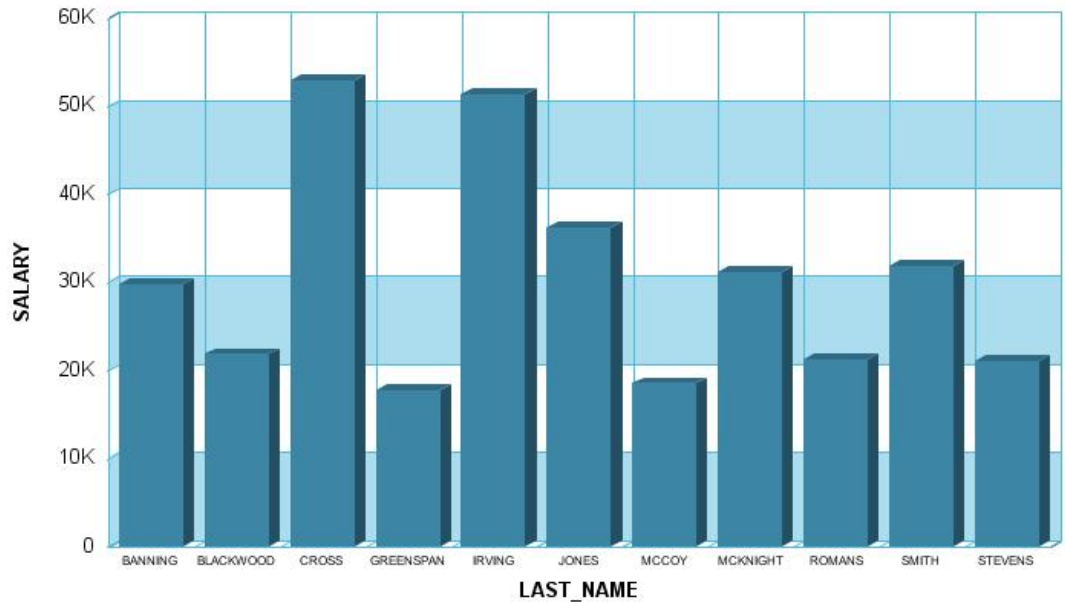
This method returns the object ID of the Y1-axis Color Band 1, which allows color banded styling to be applied to the Y1-axis grid in a graph.

Syntax:

```
IdentObj getY1MajorGridColorBand1\(\)
```


Example:

```
setDisplay(getY1MajorGridColorBand1(),true);  
setFillColor(getY1MajorGridColorBand1(), new Color (171,221,239));
```

**getY1MajorGridColorBand2()**

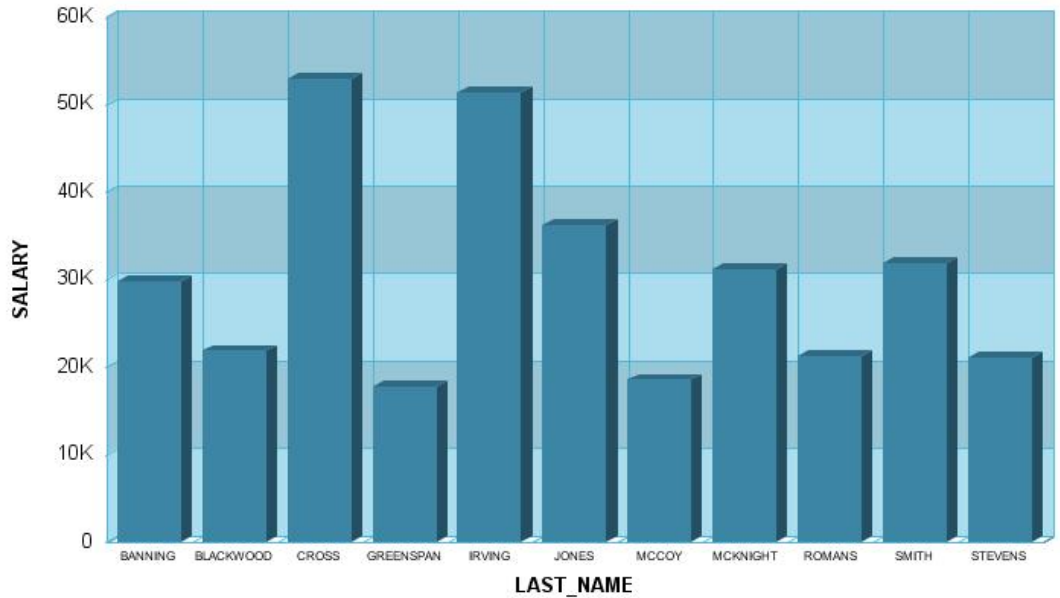
This method returns the object ID of the Y1-axis Color Band 2, which allows color banded styling to be applied to the Y1-axis grid in a graph.

Syntax:

```
IdentObj getY1MajorGridColorBand2()
```

Example:

```
setDisplay(getY1MajorGridColorBand1(),true);  
setFillColor(getY1MajorGridColorBand1(), new Color (171,221,239));  
setDisplay(getY1MajorGridColorBand2(),true);  
setFillColor(getY1MajorGridColorBand2(), new Color (153,198,214));
```

**getY1MajorGridStepAutoValue()**

When the Y1MajorGridStepAuto property is true (enabling automatic calculation of major grid steps on the Y1 axis), this method returns a Java double-precision number that identifies the interval between major grid steps on the Y1 axis.

Syntax:

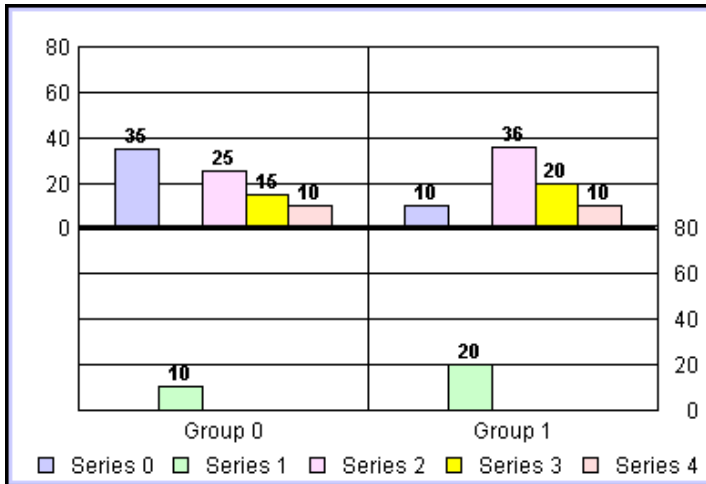
```
double getY1MajorGridStepAutoValue()
```

Example:

```

setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY1MajorGridStepAutoValue());
setDataTextDisplay(true);

```



See [Y1MajorGridStepAuto](#) on page 633.

getY1ScaleMaxAutoValue()

When the Y1ScaleMaxAuto property is true (enabling automatic calculation of the maximum scaling value on the Y1 axis), this method returns a Java double-precision number that identifies the maximum scale value that will be used on the Y1 axis.

Syntax:

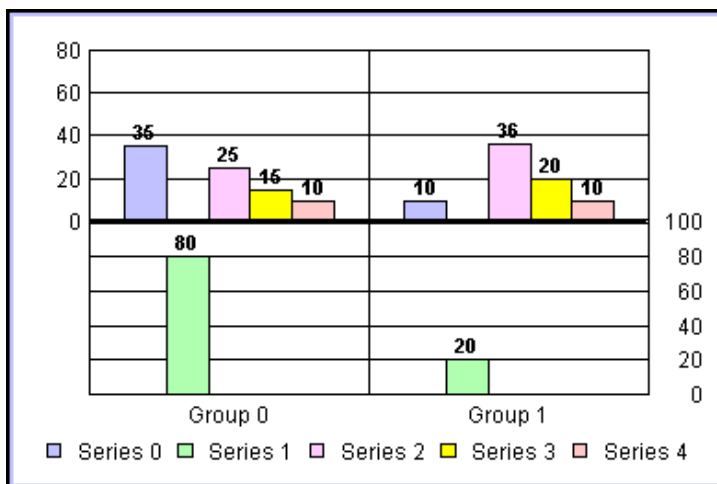
```
double getY1ScaleMaxAutoValue()
```

Example:

```

setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY1ScaleMaxAutoValue());
setDataTextDisplay(true);

```



See [getY1ScaleMinAutoValue\(\)](#) on page 832, [Y1ScaleMaxAuto](#) on page 645.

getY1ScaleMinAutoValue()

When the Y1ScaleMinAuto property is true (enabling automatic calculation of the minimum scaling value on the Y1 axis), this method returns a Java double-precision number that identifies the minimum scale value that will be used on the Y1 axis.

Syntax:

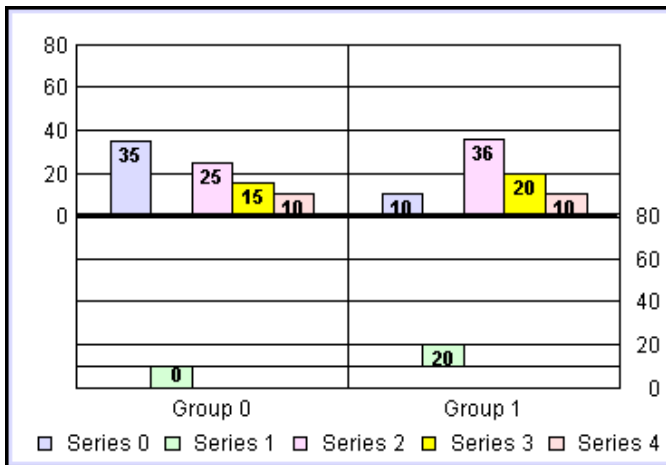
```
double getY1ScaleMinAutoValue()
```

Example:

```

setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY1ScaleMinAutoValue());
setDataTextDisplay(true);
setNonZeroBaseline(getY2Axis(),true);
setNonZeroBaselineValue(getY2Axis(),10);
setDataTextPosition(3);

```



See [getY1ScaleMaxAutoValue\(\)](#) on page 831, [Y1ScaleMinAuto](#) on page 648.

getY2MajorGridColorBand1()

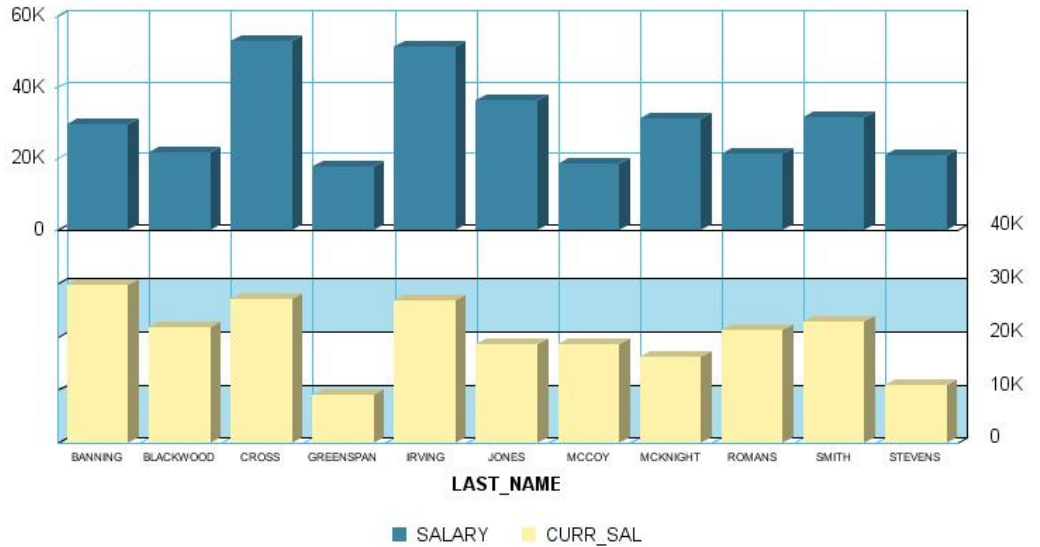
This method returns the object ID of the Y2-axis Color Band 1, which allows color banded styling to be applied to the Y2-axis grid in a graph.

Syntax:

```
IdentObj getY2MajorGridColorBand1()
```

Example:

```
setDisplay(getY2MajorGridColorBand1(),true);
setFillColor(getY2MajorGridColorBand1(), new Color (171,221,239));
```

**getY2MajorGridColorBand2()**

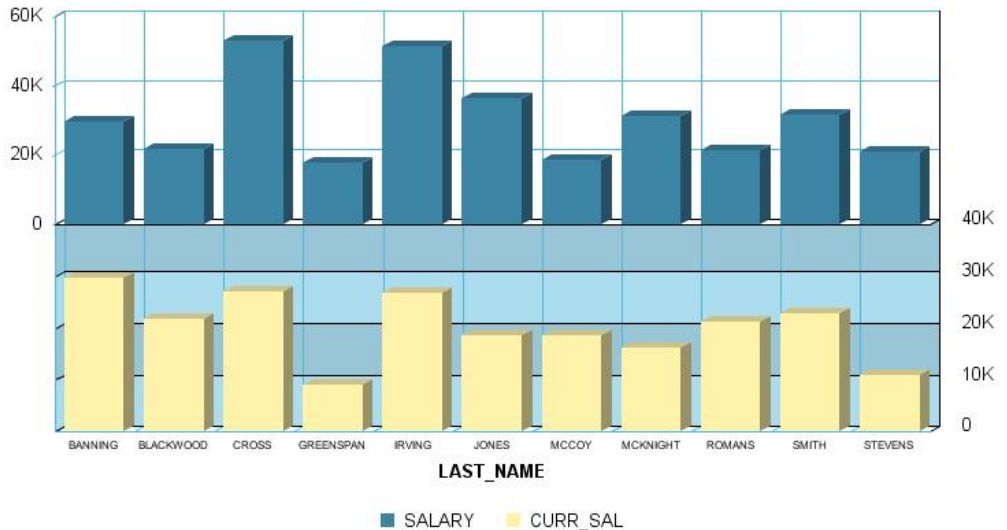
This method returns the object ID of the Y2-axis Color Band 2, which allows color banded styling to be applied to the Y2-axis grid in a graph.

Syntax:

```
IdentObj getY2MajorGridColorBand2()
```

Example:

```
setDisplay(getY2MajorGridColorBand1(),true);
setFillColor(getY2MajorGridColorBand1(), new Color (171,221,239));
setDisplay(getY2MajorGridColorBand2(),true);
setFillColor(getY2MajorGridColorBand2(), new Color (153,198,214));
```

**getY2MajorGridStepAutoValue()**

When the Y2MajorGridStepAuto property is true (enabling automatic calculation of major grid steps on the Y2 axis), this method returns a Java double-precision number that identifies the interval between major grid steps on the Y2 axis.

Syntax:

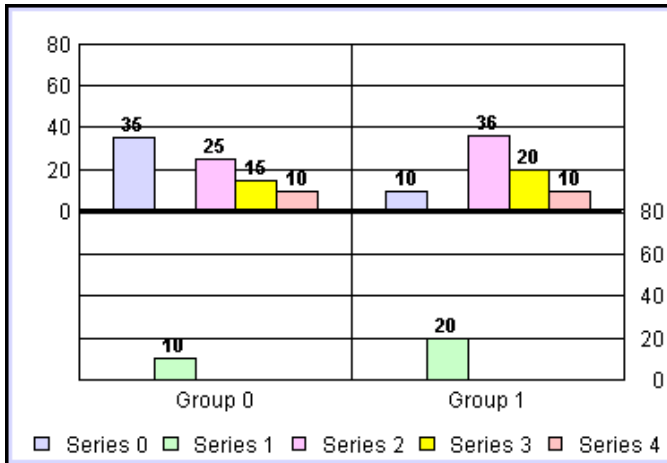
```
double getY2MajorGridStepAutoValue()
```

Example:

```

setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY2MajorGridStepAutoValue());
setDataTextDisplay(true);

```



See [Y2MajorGridStepAuto](#) on page 670.

getY2ScaleMaxAutoValue()

When the Y2ScaleMaxAuto property is true (enabling automatic calculation of the maximum scaling value on the Y2 axis), this method returns a Java double-precision number that identifies the maximum scale value that will be used on the Y2 axis.

Syntax:

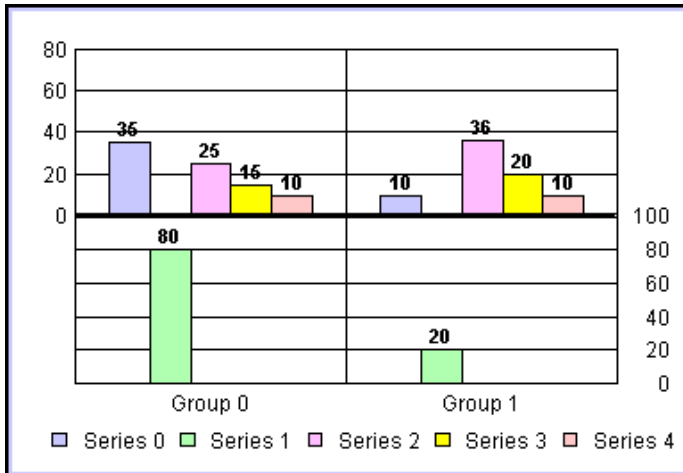
```
double getY2ScaleMaxAutoValue()
```


Example:

```

setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY2ScaleMaxAutoValue());
setDataTextDisplay(true);

```



See [getY2ScaleMinAutoValue\(\)](#) on page 837, [Y2ScaleMaxAuto](#) on page 683.

getY2ScaleMinAutoValue()

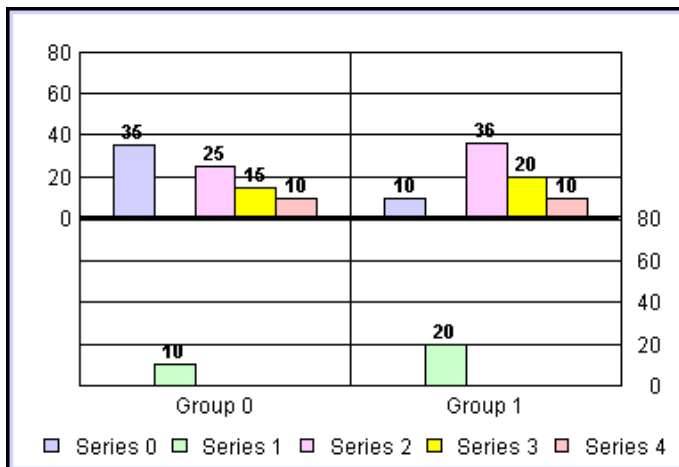
When the Y2ScaleMinAuto property is true (enabling automatic calculation of the minimum scaling value on the Y2 axis), this method returns a Java double-precision number that identifies the minimum scale value that will be used on the Y2 axis.

Syntax:

```
double getY2ScaleMinAutoValue()
```

Example:

```
setDepthRadius(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDataValue(getSeriesGroup(1,0), getY2ScaleMinAutoValue());
setDataTextDisplay(true);
```



See [getY2ScaleMaxAutoValue\(\)](#) on page 836, [Y2ScaleMinAuto](#) on page 686.

restore Methods

This section contains information on the available restore methods.

restoreAllSlices()

This method restores all deleted or detached slices to a pie graph.

Syntax:

```
void restoreAllSlices( );
```

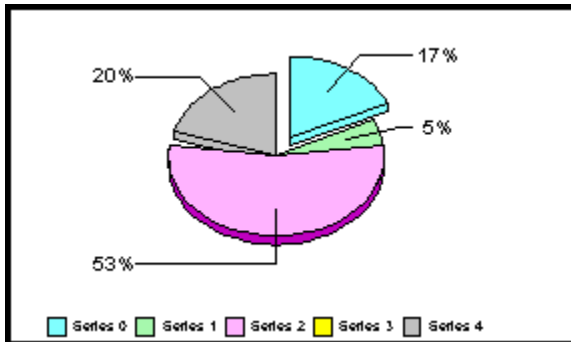
Return: *void*;

Example:

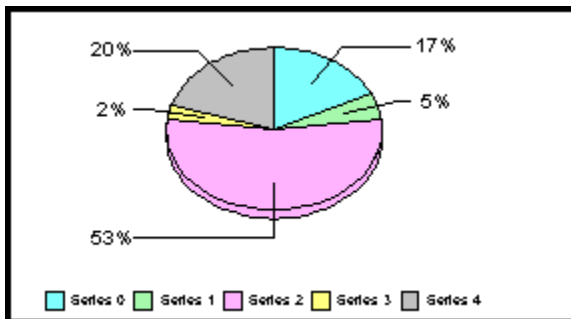
```

setPieSliceDelete(getSeries(3),true);
setPieSliceDelete(getSeriesGroup(3,0),true);
setPieSliceDelete(getSeries(0),false);
setPieSliceDelete(getSeriesGroup(0,0),false);
setPieSliceDetach(getSeries(0),100);
setPieSliceDetach(getSeriesGroup(0,0),100);

```



```
restoreAllSlices();
```



See [getPieSliceDelete\(\)](#) on page 789, [setPieSliceDelete\(\)](#) on page 948, [getPieSliceDetach\(\)](#) on page 790, [setPieSliceDetach\(\)](#) on page 949.

restoreDefaults()

This method can be used to restore a graph to its default values. It restores the default graph type (17, a vertical clustered bar graph) and the sample data associated with this graph type.

Syntax:

```
void restoreDefaults();
```

save()

This method saves a graph to an output stream. It will save all of the properties and methods required to recreate the graph in ASCII text file format. The contents of the file can be viewed or edited with any text editor.

Syntax:

```
void save(java.io.OutputStream os);  
void save(  
    java.io.OutputStream os,  
    String ChartObjectName);
```

where:

os

Is a Java output stream.

ChartObjectName

Is a graph object name.

Return: *void*;

send Methods

This section contains information on the available send methods. The methods are listed in alphabetical order.

sendChartToFTP()

This method sends a graph to an FTP server as an ASCII text file. This method returns true (success) or false (failed).

Syntax:

```
boolean sendChartToFTP(  
    String strHost,  
    String strUser,  
    String strPass,  
    String strDir,  
    String strDestFile);
```

where:

strHost

Is the host name string.

strUser

Is the user name string.

strPass

Is the password string.

strDir

Is the directory name string.

strDestFile

Is the destination file name string.

See [getChartFromFTP\(\)](#) on page 724, [sendGIFToFile\(\)](#) on page 841, [sendGIFToFTP\(\)](#) on page 842, [sendGIFToStream\(\)](#) on page 843.

sendGIFToFile()

This method sends the current graph to a file as a GIF image.

Note: When running graphs in GIF format on a UNIX platform with the HEADLESS configuration, the graph may not display properly. You may see a red X instead of your image. This is due to Sun's methodology of creating images without a head, which does not currently support GIF images.

Syntax:

```
void sendGIFToFile(os OutputStream);
```

where:

os

Is the output stream.

Return: *void*;

Example:

```
String gifFile = new String("SaveChartGIFDemo.gif"); sendGIFToFile new
FileOutputStream(gifFile));
```

See [sendChartToFTP\(\)](#) on page 840, [sendGIFToFTP\(\)](#) on page 842, [sendGIFToStream\(\)](#) on page 843.

sendGIFToFileExc()

This method sends the current graph to a file as a GIF image and throws the appropriate exception(s).

Syntax:

```
void sendGIFTToFileExc(OutputStream outs)
```

where:

outs

Is the output stream.

Return: *void*;

Note: Exceptions are not caught in this version of code.

See [sendChartToFTP\(\)](#) on page 840, [sendGIFTToFile\(\)](#) on page 841, [sendGIFTToFile\(\)](#) on page 841, [sendJPEGToStream\(\)](#) on page 844, [sendPNGToFileExc\(\)](#) on page 845, [sendPNGToStreamExc\(\)](#) on page 846, [sendGIFTToFTP\(\)](#) on page 842, [sendGIFTToStream\(\)](#) on page 843.

sendGIFTToFTP()

This method sends the current graph to an FTP server as a GIF image.

Syntax:

```
void sendGIFTToFTP(  
    String strHost,  
    String strUser,  
    String strPass,  
    String strDir,  
    String strDestFile);
```

where:

strHost

Is the host name string.

strUser

Is the user name string.

strPass

Is the password string.

strDir

Is the directory name string.

strDestFile

Is the destination file name string.

Return: *void*;

See [sendChartToFTP\(\)](#) on page 840, [sendGIFToFile\(\)](#) on page 841, [sendGIFToStream\(\)](#) on page 843.

sendGIFToStream()

This method sends the current graph to an output stream as a GIF image.

Syntax:

```
void sendGIFToStream (os OutputStream);
```

where:

os

Is the output stream.

Return: *void*;

Example:

```
//
// Save a Perspective graphtype to a GIF filestream
//
private void saveGIFStream(int nGraphType, String strFilename)
{
    TDGErrorManager errMgr =
        m_Perspective.getErrorManager();
    try {
        FileOutputStream fos =
            new FileOutputStream(strFilename);
        m_Perspective.setGraphType(nGraphType);
        m_Perspective.sendGIFToStream(fos);
        fos.close();
    } catch (FileNotFoundException fnf) {
        errMgr.notifyErrorListener(this,
            TDGError.FTP_MESSAGES,
            "File Not Found Exception:" +
            fnf.toString(), null);
    } catch (IOException e) {
        errMgr.notifyErrorListener(this,
            TDGError.FTP_MESSAGES, "I/O Exception:" +
            e.toString(), null);
    }
}
```

See [sendChartToFTP\(\)](#) on page 840, [sendGIFToFile\(\)](#) on page 841, [sendGIFToFTP\(\)](#) on page 842.

sendImageMapToStream()

This method creates a map of the image that forms a graph and sends it to a Java I/O `PrintWriter` object.

Syntax:

```
void sendImageMapToStream(  
    java.io.PrintWriter outWriter,  
    String mapName,  
    String pathName,  
    boolean bPage)
```

where:

outWriter

Is a Java I/O PrintWriter object.

mapName

Is the name of the ImageMap.

pathName

Is the name of the path to write the ImageMap to.

bPage

Can be one of the following values:

true (pageable)

false (non-pageable)

Return: *void*;

See [sendGIFToStream\(\)](#) on page 843.

sendJPEGToStream()

This method sends the graph to an output stream in JPEG format.

Syntax:

```
void sendJPEGToStream(  
    OutputStream outs, int nJPEGQuality)
```

where:

outs

Is the output stream.

nJPEGQuality

Is the JPEG quality number.

Return: *void*;

See [sendChartToFTP\(\)](#) on page 840, [sendGIFToFile\(\)](#) on page 841, [sendGIFToFileExc\(\)](#) on page 841, [sendGIFToFTP\(\)](#) on page 842, [sendPNGToFileExc\(\)](#) on page 845, [sendPNGToStreamExc\(\)](#) on page 846, [sendGIFToFTP\(\)](#) on page 842, [sendGIFToStream\(\)](#) on page 843.

sendPNGToFile()

This method writes the graph image to the specified file in Portable Network Graphics (.png) format. Any errors are passed to TDGErrorListeners with an ID of TDGError.WRITE_ERROR.

Syntax:

```
void sendPNGToFile (java.io.File pngFile)
```

where:

pngFile

Is the file to write the graph image to.

Return: *void*;

See [sendPNGToFileExc\(\)](#) on page 845, [sendPNGToStream\(\)](#) on page 845.

sendPNGToFileExc()

This method saves the graph in a file in Portable Network Graphics (.png) format and throws the appropriate exception(s).

Syntax:

```
void sendPNGToFileExc (File pngFile)
```

where:

pngFile

Is the file where the graph image will be stored.

Return: *void*;

See [sendPNGToFile\(\)](#) on page 845, [sendPNGToStream\(\)](#) on page 845, [sendPNGToStreamExc\(\)](#) on page 846.

sendPNGToStream()

This method writes the graph image to the specified output stream in Portable Network Graphics (.png) format. Any errors are passed to TDGErrorListeners with an ID of TDGError.WRITE_ERROR.

Syntax:

```
void sendPNGToStream (java.io.OutputStream pngStream)
```

where:

pngStream

Is the stream to write the graph image to.

Return: *void*;

See [sendPNGToFile\(\)](#) on page 845.

sendPNGToStreamExc()

This method writes the graph image to the specified stream in Portable Network Graphics (.png) format and throws the appropriate exception(s).

Syntax:

`void sendPNGToStreamExc (OutputStream pngStream)`

where:

pngStream

Is the output to stream.

Return: *void*;

See [sendPNGToFileExc\(\)](#) on page 845.

set Methods

This section contains information on the available set methods.

setAltFmtFrameNumColors()

This method sets the number of colors/sections to be used for alternate formatting of an axis. Alternate formatting can be used to assign different colors to individual sections of an axis and axis labels. You may also define a text string label that is shown with the alternate formatted section.

Syntax:

`setAltFmtFrameNumColors (IdentObj idObj, int newValue);`

where:

idObj

Is the object ID of an alternate formatting axis object (for example, `getY1AltFmtFrame()`, `getO1AltFmtFrame()`, and so on).

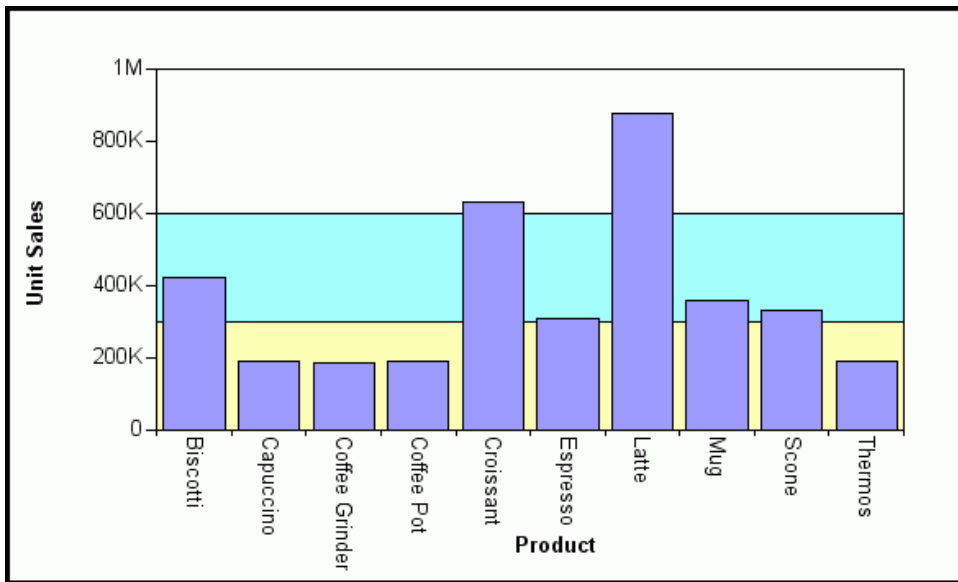
newValue

For `setAltFmtFrameNumColors()`, number of colors/sections to use for alternate formatting of the specified axis (`idObj`). It can be a value in the range 1...100. Default value is 2.

This method returns the number of colors/sections that are used for alternate formatting of the specified axis (`idObj`) for `getAltFmtFrameNumColors()` as an integer.

Example:

```
setAltFmtFrameNumColors(getY1AltFmtFrame(),3);
setAltFmtFrameSeparator(getY1AltFmtFrame(0),300000.0);
setFillColor(getY1AltFmtFrame(0),new Color(251,255,181));
setAltFmtFrameSeparator(getY1AltFmtFrame(1),600000.0);
setFillColor(getY1AltFmtFrame(1),new Color(164,255,255));
```



Note: The number of separator lines displayed in the graph above is equal to the number of alternate formatting colors minus one. Separators exist only where two alternate formatting colors meet.

Also see [Alternate Formatting](#) on page 174.

setAltFmtFrameSeparator()

This method sets the location on an axis where alternate formatting is applied.

Syntax:

```
setAltFmtFrameSearator(IdentObj idObj, double newValue);
```

where:

idObj

Is the object ID of an alternate formatting axis object (for example, `getY1AltFmtFrame()`, `getO1AltFmtFrame()`, and so on).

newValue

For `setAltFmtFrameSeparator()`, the location on the axis where alternate formatting will be applied. For a numeric axis, specify a value that is within the range of values on the specified axis. For the ordinal (O1) axis, specify a group number that is within the range of groups on the ordinal axis.

This method returns a double value for `getAltFmtFrameSeparator()`, the location on the axis where alternate formatting has been applied.

See [Alternate Formatting](#) on page 174.

setAutoColorLerpFactor()

This method sets a lightening or darkening factor for shading groups of risers. It is used in conjunction with color mode 4 and `setSeriesLooping`. For information, see [Risers and Markers](#) on page 92.

Syntax:

```
void setAutoColorLerpFactor(double newValue);
```

where:

newValue

Is a double-precision number between -1 and 1 that represents the factor for lightening or darkening each successive group of risers. A positive factor specifies that each successive group of risers will be lighter than the previous group. A negative number specifies that each successive group of risers will be darker than the previous group. Zero indicates that each successive group of risers will be shaded exactly the same as the previous group.

Return: *void*;

See [Risers and Markers](#) on page 92, [ColorMode](#) on page 260, [setSeriesLooping\(\)](#) on page 980.

setAutoSkip()

This method sets or disables label skip for labels on the O1 axis. If this method selects manual label skip (2), use setSkipBegin() to define the first label to skip and setSkipCount() to define the skip interval. If this method selects automatic skip mode (1), labels will automatically be omitted/skipped when the graph size causes the labels to be drawn in less than 8-point type. Automatic skip mode may omit all but two labels if the graph is reduced to its minimum size.

Syntax:

```
void setAutoSkip(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1Label()).

newValue

Can be one of the following values:

0 (no label skipping)

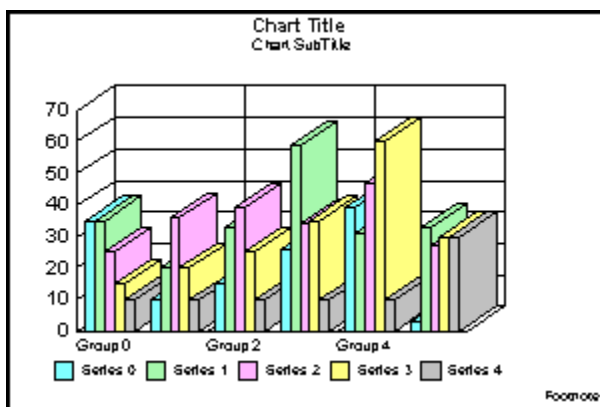
1 (automatic label skipping)

2 (manual skipping as defined by setSkipBegin() and setSkipCount())

Return: void;

Example:

```
setAutoSkip(getO1Label(),2);
setSkipBegin(getO1Label(),1);
setSkipCount(getO1Label(),1);
```



Note: If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [setSkipBegin\(\)](#) on page 991 and [setSkipCount\(\)](#) on page 992.

setAxisAssignment()

In multi-axis graphs, this method can be used to assign individual series to the Y1, Y2, Y3, Y4, and Y5 axes.

Syntax:

```
void setAxisAssignment (IdentObj id, int newValue);  
void setAxisAssignment (int seriesID, int newValue);
```

For merged multi-axis graphs:

```
void setAxisAssignment ([$] seriesID, newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries(1)`).

seriesID

Is a series number.

\$seriesID

Automatically assigns an axis to each series by using the number of measures as an increment that ensures that all series related to the same measure use the same axis. This automatic series increment is also available for [setSeriesType\(\)](#) on page 981.

newValue

Can be one of the following values:

0 (Y1 axis)

1 (Y2 axis)

2 (Y3 axis)

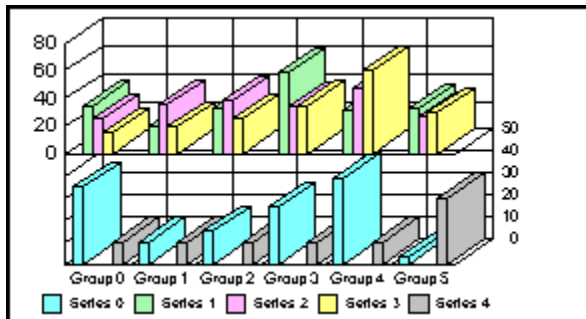
3 (Y4 axis)

4 (Y5 axis)

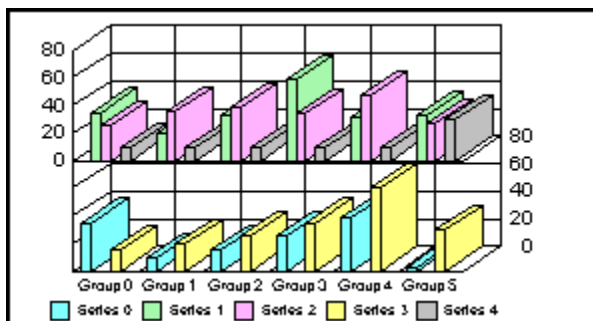
Return: *void*;

Example:

```
setGraphType(21);  setAxisAssignment(getSeries(1),0);
setAxisAssignment(getSeries(0),1);
```



```
setAxisAssignment(getSeries(3),1);  setAxisAssignment(getSeries(4),0);
```



Example for a Multi-Axis Merged Graph

The following request uses a bi-polar bar chart with two measures (CURR_SAL and SALARY) and two sort fields (BY DEPARTMENT and ACROSS LAST_NAME). This makes it a multi-axis merged graph request:

```
GRAPH FILE EMPLOYEE
SUM EMPLOYEE.EMPINFO.CURR_SAL
EMPLOYEE.PAYINFO.SALARY
BY EMPLOYEE.EMPINFO.DEPARTMENT
ACROSS EMPLOYEE.EMPINFO.LAST_NAME
ON GRAPH SET LOOKGRAPH VBAR2AXS
ON GRAPH SET 3D OFF
ON GRAPH SET BARNUMB OFF
ON GRAPH SET GRID ON
ON GRAPH SET GRAPHEDIT OFF
ON GRAPH SET GRMERGE ON
ON GRAPH SET VZERO ON
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET

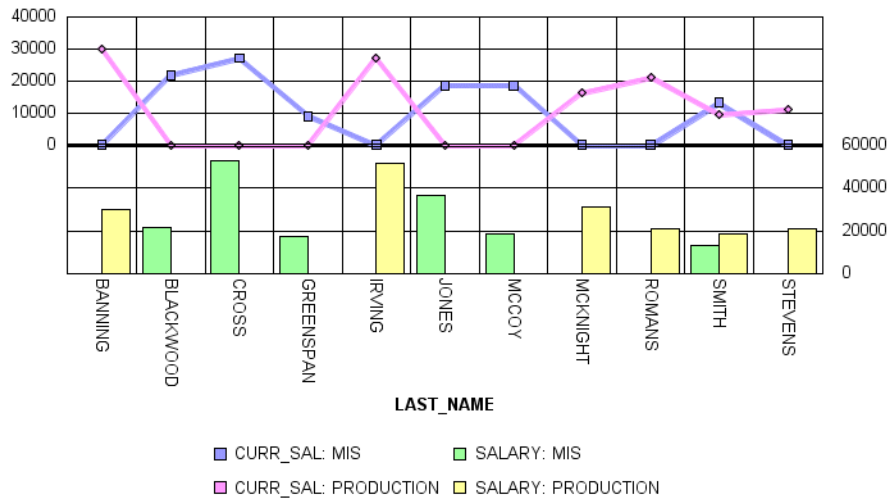
STYLE *
*GRAPH_SCRIPT  -* ***Axis Assignments***
  setAxisAssignment($0,0);
setAxisAssignment($1,1);setSeriesType($0,2);
setSeriesType($1,1);
*END
```

Since DEPARTMENT has two values and there are two measures, there are four series in the graph request: CURR_SAL for the MIS department, CURR_SAL for the PRODUCTION department, SALARY for the MIS department, and SALARY for the PRODUCTION department

The axis assignments are setAxisAssignment(\$0,0); and setAxisAssignment(\$1,1). These are translated to the following API calls that make sure that series 0 and 2 (based on CURR_SAL) are assigned to the Y1 (left axis) and that series 1 and 3 (based on SALARY) are assigned to the Y2 (right) axis:

```
setAxisAssignment(0,0);
setAxisAssignment(2,0);
setAxisAssignment(1,1);
setAxisAssignment(3,1);
```


The output is:



See [getAxisAssignment\(\)](#) on page 716, [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856, [Y1AxisSide](#) on page 619, [Y2AxisSide](#) on page 655.

setAxisDescending()

This method can be used to draw an axis in ascending or descending order.

Syntax:

```
void setAxisDescending(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getY1Axis()`).

newValue

Can be one of the following values:

`true` (descending)

`false` (ascending)

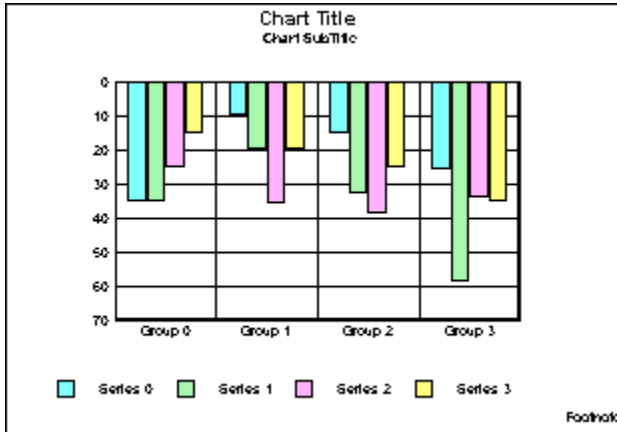
Return: `void`;

Example:

```

setDepthRadius(0);
setViewableGroups(4);
setViewableSeries(4);
setDataScrollerPresenceGroups(0); setAxisDescending(getY1Axis(), true);

```



Note: You may also use one of the axis-specific properties (for example, `X1AxisDescending`, `Y1AxisDescending`, and so on) to set the descending attribute of an axis.

See [getAxisAssignment\(\)](#) on page 716, [setAxisAssignment\(\)](#) on page 850, [getAxisSide\(\)](#) on page 720, [setAxisSide\(\)](#) on page 856, [X1AxisDescending](#) on page 575, [Y1AxisDescending](#) on page 617, [Y2AxisDescending](#) on page 653.

setAxisOffset()

In multi-Y graphs (3 or more Y-axes), axes can overlap. This method and `setAxisSize()` can be used to create a more desirable graph layout for multi-Y axes graphs. `setAxisSize()` sets the size of the frame, this method sets the axis offset from the base of the frame.

Syntax:

```

void setAxisOffset(int newValue)
void setAxisOffset(IdentObj id, int newValue)

```

where:

newValue

Is the axis offset value 0...100 (0=automatic sizing).

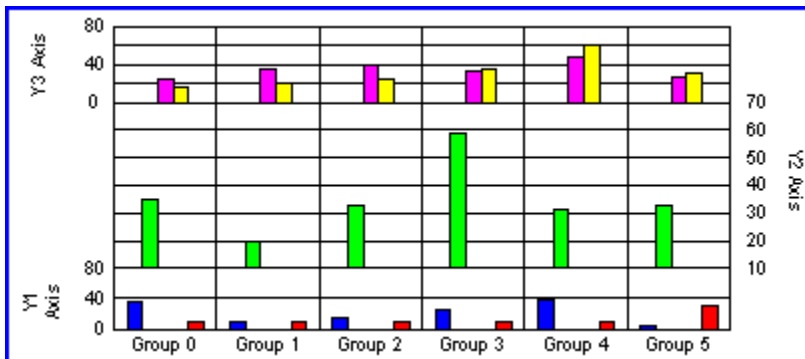
id

Is an object ID of a Y-axis returned by `getY1Axis()`, `getY2Axis()`, `getY3Axis()`, `getY4Axis()`, or `getY5Axis()`.

Return: *void*;

Example:

```
setAxisAssignment(getSeries(0), 0);
setAxisAssignment(getSeries(1), 1);
setAxisAssignment(getSeries(2), 2);
setAxisAssignment(getSeries(3), 2);  setAxisOffset(getY1Axis(), 0);
setAxisOffset(getY2Axis(), 20);  setAxisOffset(getY3Axis(), 75);
setAxisSize(getY1Axis(), 20);
setAxisSize(getY2Axis(), 55);
setAxisSize(getY3Axis(), 25);
setGraphType(103);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setDepthRadius(0);
```



Note:

- ❑ If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).
- ❑ If `setAxisSize()` sets the size of the axis to zero (automatic), the axis offset will also be automatically calculated regardless of the offset set by this method.

See [getAxisOffset\(\)](#) on page 718 and [getAxisSize\(\)](#) on page 721.

setAxisSide()

This method defines the side of a graph where an axis is imaged.

Syntax:

```
void setAxisSide(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Axis()).

newValue

Can be a value of 0...2. If the value is:

0, the axis is imaged on the left (or bottom for horizontal graphs).

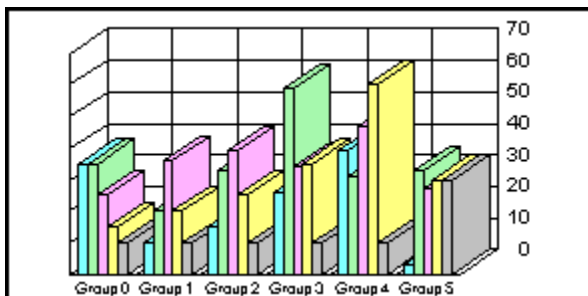
1, the axis is imaged on the right (or top for horizontal graphs).

2, the axis is imaged on both sides of graph.

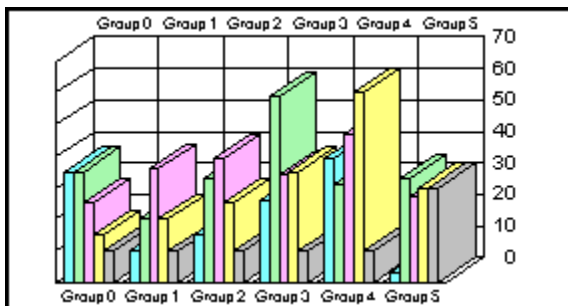
Return: *void*;

Example:

```
setAxisSide(getY1Axis(),1);
```



```
setAxisSide(getO1Axis(),2);
```



Note: You may also use one of the axis-specific properties (for example, `X1AxisSide`, `Y1AxisSide`, and so on) to assign an axis to a side.

See [getAxisAssignment\(\)](#) on page 716, [setAxisAssignment\(\)](#) on page 850, [getAxisDescending\(\)](#) on page 717, [setAxisDescending\(\)](#) on page 853, [O1AxisSide](#) on page 425, [X1AxisSide](#) on page 578, [Y1AxisSide](#) on page 619, [Y2AxisSide](#) on page 655.

setAxisSize()

In multi-Y graphs (3 or more Y-axes), axes can overlap. This method and `setAxisOffset()` can be used to create a more desirable graph layout for multi-Y axes graphs. This method sets the size of the frame, `setAxisOffset()` sets the axis offset from the base of the frame.

Syntax:

```
void setAxisSize(IdentObj id, int newValue)
```

where:

newValue

Is the axis size 0...100 (0 = automatic).

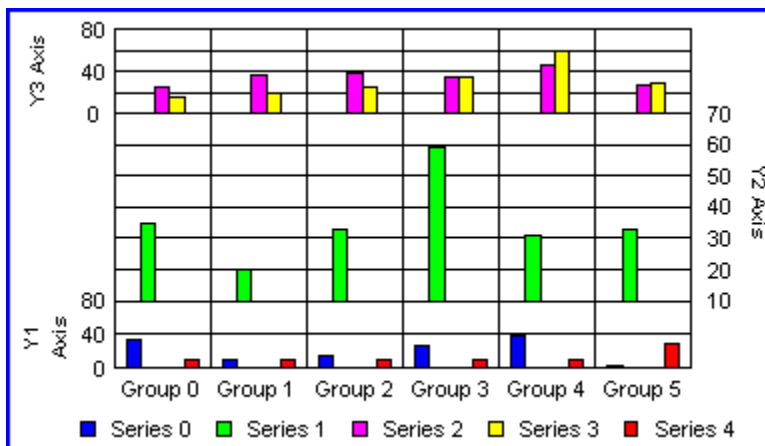
id

Is an object ID of a Y-axis returned by `getY1Axis()`, `getY2Axis()`, `getY3Axis()`, `getY4Axis()`, or `getY5Axis()`.

Return: *void*;

Example:

```
setAxisAssignment(getSeries(0), 0);
setAxisAssignment(getSeries(1), 1);
setAxisAssignment(getSeries(2), 2);
setAxisAssignment(getSeries(3), 2);
setAxisOffset(getY1Axis(), 0);
setAxisOffset(getY2Axis(), 20);
setAxisOffset(getY3Axis(), 75); setAxisSize(getY1Axis(), 20);
setAxisSize(getY2Axis(), 55); setAxisSize(getY3Axis(), 25);
setGraphType(103);
setDepthRadius(0);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
```



Note: If you set the axis size to zero (automatic) with this method, the axis offset will also be automatically calculated regardless of the value set by `setAxisOffset()`.

See [getAxisSize\(\)](#) on page 721.

setBorder()

This method draws a specified border on an annotation, title box, subtitle box, or footnote box object. The default border width is 200. Therefore, a border can be assigned without setting a width.

Syntax:

```
void setBorder(  
    IdentObj anObject,  
    String aBorder  
)  
  
void setBorder(  
    IdentObj anObject,  
    String aBorder,  
    int aWidth  
)
```

where:

anObject

Is an object ID returned by a getObject() method.

aBorder

Is one of the following border types:

- ☐ EmptyBorder
- ☐ CompoundBorder
- ☐ BevelBorder
- ☐ ShadowBorder
- ☐ EtchedBorder
- ☐ LineBorder
- ☐ DoubleLineBorder

aWidth

Is the width of the border in pixels.

Return: *void*;

Note:

- ❑ Currently supported objects include: an annotation object (for example, `getAnnotationBox(0)`), the title box object (that is, `getTitleBox()`), the subtitle box object (that is, `getSubtitleBox()`), or the footnote box object (that is, `getFootnoteBox()`).
- ❑ Borders supersede Insets as the mechanism for creating a (decorated or plain) area around the edge of an object.

See [getBorder\(\)](#) on page 722.

setBorderColor()

This method sets the border color of an object in a graph.

Syntax:

```
void setBorderColor(new Color(newValue));  
void setBorderColor(IdentObj id, new Color(newValue));
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries(1)`).

newValue

Is the value of the border color, in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

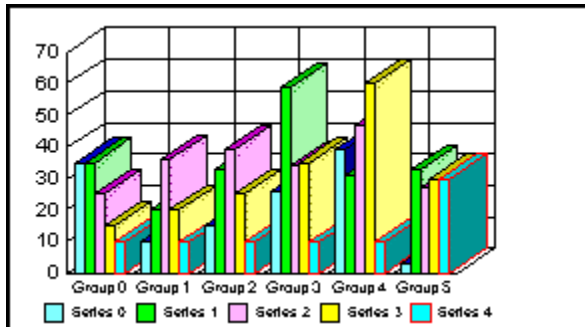
Return: *void*;

Example:

```
setBorderColor(getSeries(4),new Color(255,0,0));  
setFillColor(getSeries(4),new Color(0,255,255));
```


or, using hexadecimal color codes:

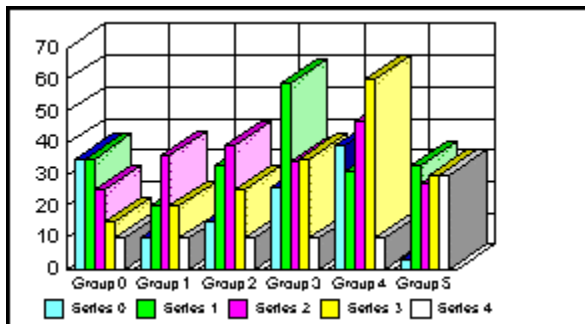
```
setBorderColor(getSeries(4),new Color(#FF0000));
setFillColor(getSeries(4),new Color(#00FFFF));
```



```
setBorderColor(getSeries(4),new Color(0,0,0));
setFillColor(getSeries(4),new Color(255,255,255));
```

or, using hexadecimal color codes:

```
setBorderColor(getSeries(4),new Color(#000000));
setFillColor(getSeries(4),new Color(#FFFFFF));
```



Note: If an object ID (id) is not specified, `newValue` is assigned to the first item in the selection list (if applicable).

See [getFillColor\(\)](#) on page 746, [setFillColor\(\)](#) on page 890, [getTransparentBorderColor\(\)](#) on page 824, [setTransparentBorderColor\(\)](#) on page 1022, [setSeriesBorderColor\(\)](#) on page 976, [ColorMode](#) on page 260.

setBorderType()

This method sets the border type for a specified object. The default value is NONE.

Syntax:

```
setBorderType(IdentObj IdObj, int nBorderType);
```

where:

IdObj

Is the object ID returned by `getAnnotationBox()`, `getBeveledLegend()`, `getBeveledLegendMarker()`, `getFootnoteBox()`, `getSubtitleBox()`, or `getTitleBox()`.

nBorderType

One of the following border types:

- ❑ `BORDER_NONE = 0;`
- ❑ `BORDER_BEVEL = 1;`
- ❑ `BORDER_BEVEL_RAISED = 2;`
- ❑ `BORDER_BEVEL_LOWERED = 3;`
- ❑ `BORDER_LINE = 4;`
- ❑ `BORDER_LINE_DOUBLE = 5;`

Return: *void*;

See [Beveled Borders](#) on page 178 and [getBorderType\(\)](#) on page 724.

setBorderWidth()

This method sets the width of a border that may be drawn around an annotation, title box, subtitle box, or footnote box object. It is not necessary to call `setBorder()` before defining the width with this method. This is a convenience method that can also be used from the scripting interface.

Syntax:

```
void setBorderWidth(
    IdentObj anObject,
    int borderWidth )

void setBorderWidth(
    IdentObj anObject,
    Insets theBorderInsets )
```

where:

anObject

Is an object ID returned by a `getObject()` method.

borderWidth

Is a uniform border width for all sides of anObject.

theBorderInsets

Is the uniform inset for all sides of anObject.

Return: *void*;

Note: Currently supported objects include: an annotation object (for example, `getAnnotationBox(0)`), the title box object (that is, `getTitleBox()`), the subtitle box object (that is, `getSubtitleBox()`), or the footnote box object (that is, `getFootnoteBox()`).

See [getBorderWidth\(\)](#) on page 724, [setBorder\(\)](#) on page 859.

setCornerArc()

This method applies a corner arc of a specified width and height to an annotation title box, subtitle box, or footnote box object.

Syntax:

```
void setCornerArc(
    IdentObj anObject,
    int aWidth,
    int aHeight )
```

where:

anObject

Is an object ID returned by a `getObject()` method.

aWidth

Is the width of the corner arc.

aHeight

Is the height of the corner arc.

Return: *void*;

Note: Currently supported objects include: an annotation object (for example, `getAnnotationBox(0)`), the title box object (that is, `getTitleBox()`), the subtitle box object (that is, `getSubtitleBox()`), or the footnote box object (that is, `getFootnoteBox()`).

See [getCornerArcWidth/Height\(\)](#) on page 725.

setCornerArcWidth/Height()

These methods set the width and height of the corner arc of a border applied to an object. The default value is 5.

Syntax:

```
setCornerArcHeight (IdentObj idObj, int newValue);  
setCornerArcWidth (IdentObj idObj, int newValue);
```

where:

idObj

Is the object ID returned by `getAnnotationBox()`, `getBeveledLegend()`, `getBeveledLegendMarker()`, `getFootnoteBox()`, `getSubtitleBox()`, or `getTitleBox()`.

newValue

Is the height or width of corner arc in virtual coordinates.

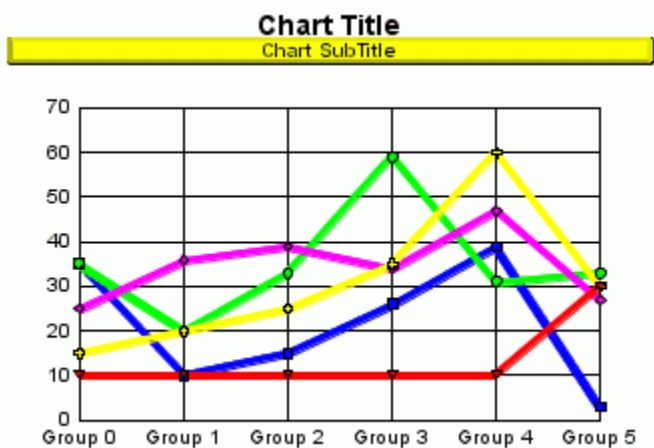
Return: void

Example:

```

setDepthRadius(0);
setGraphType(41);
setDisplay(getSubtitleLabel(), true);
setBorderType(getSubtitleLabel(), 3);
setTopInset(getSubtitleLabel(), 250);
setLeftInset(getSubtitleLabel(), 250);
setBottomInset(getSubtitleLabel(), 250);
setRightInset(getSubtitleLabel(), 250);
setFillColor(getSubtitleLabel(), new Color(255,255,0));
setCornerArcWidth(getSubtitleLabel(), 500);
setCornerArcHeight(getSubtitleLabel(), 500);

```

**setCubeRotationMatrix()**

This method sets the rotation matrix of a 3D graph.

Syntax:

```

void setCubeRotationMatrix (double rx, double ry,
                             double rq);
void setCubeRotationMatrix (tdg.model3d.Matrix3d m);
void setCubeRotationMatrix(double m00, double m01,
                             double m02, double m10, double m11, double m12,
                             double m20, double m21, double m22)

```

where:

rx

Is the rotation in x direction.

ry

Is the rotation in y direction.

rz

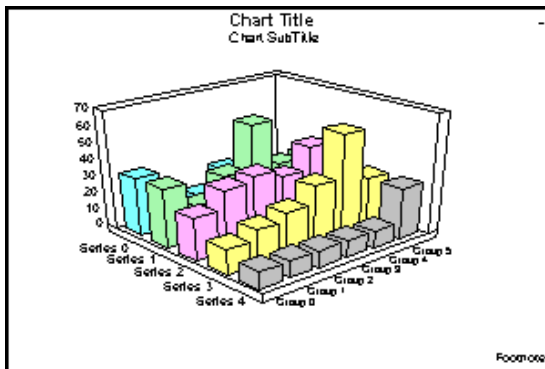
Is the rotation in z direction.

Return: *void*;**Example:**

```

setCubeFocusFactor(87.49999876279527);
setCubeRotationMatrix(0.9595622863061368,
    -0.4715354993437172, 1.031956469611238,
    -0.03648998497747465, 1.3380131050659487,
    0.6453131606305538, -1.1339966328478057,
    -.44205753827450966, 0.8524531853183448);
setCubeViewerZ(80.8979539352483);
setGraphType(0);

```

**Note:** The default setting is

```

setCubeRotationMatrix (0.7071067811865474, 0.0, 0.7071067811865477,
    -0.3535533905932738, 0.8660254037844387, 0.3535533905932736,
    -0.6123724356957947,
    -0.49999999999999994, 0.6123724356957945);

```

setCurveFitPolynomialOrder()

If the setCurveFit() method selects a polynomial fit line across a series of risers (that is, setCurveFit(FIT_POLYNOMIAL)), this method sets the order (the largest power to which x is raised) for the polynomial fit. The default value is 3.

Syntax:

```
void setCurveFitPolynomialOrder(IdentObj id, int newValue)
void setCurveFitPolynomialOrder(int seriesID, int newValue)
```

where:

newValue

Is the order for the polynomial fit (0...10).

id

Is the object ID returned by a getObjectID() method (for example, getSeries()).

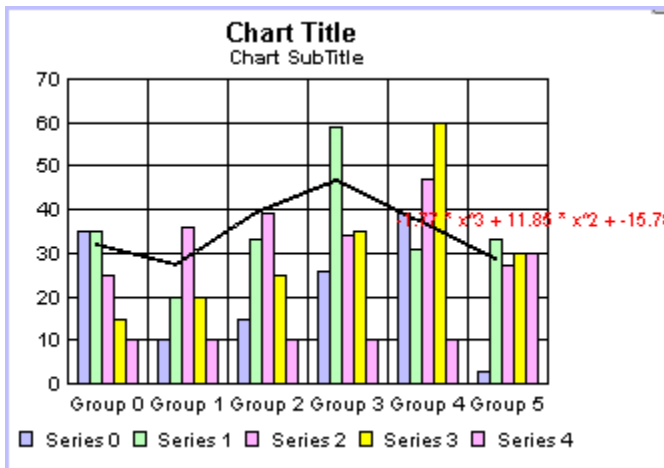
seriesID

Is the series number.

Return: void

Example:

```
setDepthRadius(0);
setCurveFitType(1,3); setCurveFitPolynomialOrder(1,3);
```



Note: If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [getCurveFitPolynomialOrder\(\)](#) on page 726, [getCurveFitType\(\)](#) on page 726, [setCurveFitType\(\)](#) on page 868.

setCurveFitType()

This method selects a curve fit line and draws the selected curve type across one or more series of risers in a graph.

Syntax:

```
void setCurveFitType(IdentObj id, int newValue)
void setCurveFitType(int seriesID, int newValue)
```

where:

newValue

Is the curve fit type (0...11) described in the following table.

CurveFitType Constant	Value	Description
FIT_NONE	0	None
FIT_LINEAR	1	Linear regression curve fit line
FIT_QUADRATIC	2	Quadratic curve fit line
FIT_POLYNOMIAL	3	Polynomial fit line
FIT_HYPERBOLIC	4	Hyperbolic curve fit line
FIT_LOGARITHMIC	5	Logarithmic curve fit line
FIT_MODHYPERBOLIC	6	Modified hyperbolic
FIT_RATIONAL	7	Rational curve fit line
FIT_EXPONENTIAL	8	Exponential curve fit line
FIT_MODEXPONENTIAL	9	Modified exponential
FIT_LOGQUADRATIC	10	Logarithmic/Quadratic curve fit line
FIT_GEOMETRIC	11	Geometric curve fit line

id

Is the object ID returned by a getObjectID() method (for example, getSeries()).

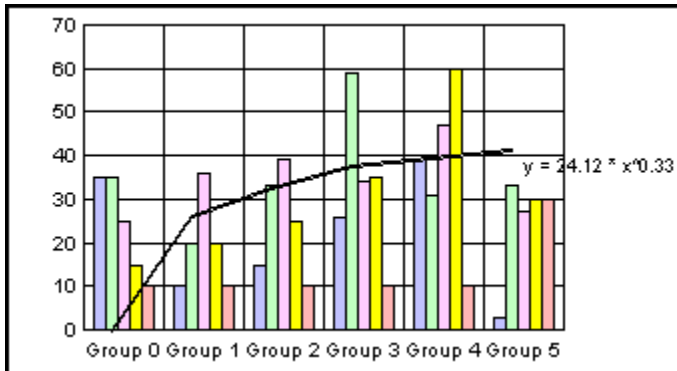
seriesID

Is the series number.

Return: *void*

Example:

```
setDepthRadius(0); setCurveFitType(1,11);
```



Note: If an object ID (id) or series ID (seriesID) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [getCurveFitType\(\)](#) on page 726, [CurveFitEquationDisplay](#) on page 303, [CurveFitHighOrderFirst](#) on page 305.

setCustomDataText()

When data text is displayed with `setDataTextDisplay(true)`, the custom data text methods allow you to assign a different format to each series in the graph. This method enables/disables custom data text for a specified series. When enabled, the `setCustomDataTextFormat()` and `setCustomDataTextFormatPattern()` methods can be used to assign a different format to each series.

Syntax:

```
void setCustomDataText(
    IdentObj idSeries,
    boolean bNewValue)
```

where:

idSeries

Is an object ID returned by a `getObject()` method (for example, `getSeries(0)`).

bNewValue

Is a boolean where a value of:

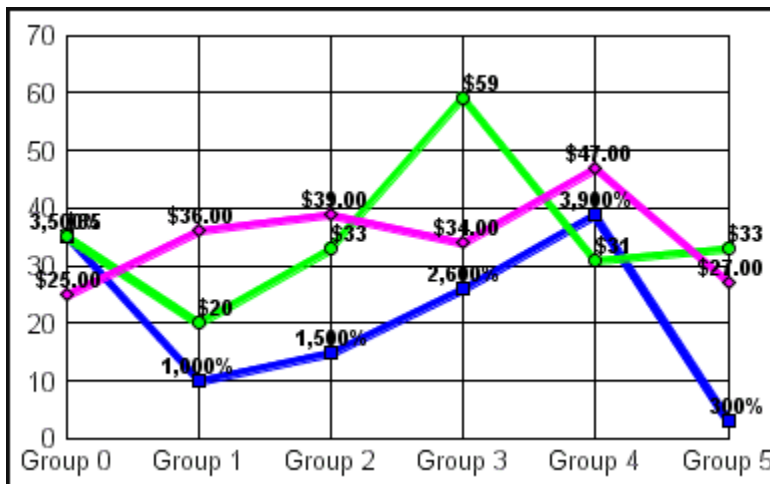
`true` enables custom data text for the specified series.

`false` disables custom data text for the series.

Return: *void*

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setCustomDataText(getSeries(2), true);
setCustomDataText(getSeries(1), true);
setCustomDataText(getSeries(0), true);
setCustomDataTextFormat(getSeries(2), 5);
setCustomDataTextFormat(getSeries(1), 6);
setCustomDataTextFormat(getSeries(0), 2);
setDataTextDisplay(true);
setViewableSeries(3);
```



See [setCustomDataTextFormat\(\)](#) on page 870, [setCustomDataTextFormatPattern\(\)](#) on page 872.

setCustomDataTextFormat()

This method assigns a data text format to a specified series. Custom data text must be enabled for the series with `setCustomDataText(getSeries(n), true);`.

Syntax:

```
void setCustomDataTextFormat(
    IdentObj idSeries,
    int newValue)
```

where:

idSeries

Is an object ID returned by a getObject() method (for example, getSeries(0)).

newValue

Is a value between -1 and 21. -1 tells the graphing engine that the format will be defined by setCustomDataTextFormatPattern().

0...21 selects one of the following formats:

0/1 = # (for example, 123 = 123)

2 = %# (for example, 123 = 12,300%)

3 = #.##% (for example, 123 = 12,300.0%)

4 = #.###% (for example, 123 = 12,300.00%)

5 = \$#.## (for example, 123 = \$123.00)

6 = \$# (for example, 123 = \$123)

7 = #K (Show K for values over 999) (for example, 1,234 = 1K)

8 = \$#K (Show K for values over 999) (for example, 1,234 = \$1K)

9 = #M (Show M for millions) (for example, 1,234,567 = 1M)

10 = \$#M (Show M for millions) (for example, 1,234,567 = \$1M)

11 = #B (Show B for billions) (for example, 1,234,567,891 = 1B)

12 = \$#B (Show B for billions) (for example, 1,234,567,891 = \$1B)

13 = #T (Show T for trillions) (for example, 1,234,567,891,234 = 1T)

14 = \$#T (Show T for trillions) (for example, 1,234,567,891,234 = \$1T)

15 = Number with thousands separators, no decimal places (for example, 1,234 = 1K)

16 = Number with thousands separators, two decimal places (for example, 1,234 = 1.23K)

17 = General currency format for current Locale

18 = Short Date Format: MM/DD/YY (for example, 10/01/03)

19 = Medium Date Format: Mon. DD, YYYY (for example, Oct. 01, 2003)

20 = Long Date Format: Month DD, YYYY (for example, October 01, 2003)

21 = Full Date Format: Day, Month DD, YYYY (for example, Wednesday, October 01, 2003)

Return: *void*;

See [setCustomDataText\(\)](#) on page 869.

setCustomDataTextFormatPattern()

When `setCustomDataTextFormat()` assigns -1 as the data text format for a particular series, this method defines a data text format pattern for the series.

Syntax:

```
void setCustomDataTextFormatPattern(  
    IdentObj idSeries,  
    String newValue)
```

where:

idSeries

Is an object ID returned by a `getObject()` method (for example, `getSeries(0)`).

newValue

Is a pattern string in the following format:

```
pattern:= subpattern{;subpattern}subpattern:=  
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFF' -  
specialCharacterssuffix:= '\\u0000'..'\\uFFFF' -  
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

Notation:

<i>X*</i>	0 or more instances of X
<i>(X Y)</i>	either X or Y.
<i>X..Y</i>	any character from X up to Y, inclusive.
<i>S - T</i>	characters in S, except those in T

The first subpattern is for positive numbers. The second (optional) subpattern is for negative numbers. In both cases, a comma (,) can occur inside the integer portion. Here are the special characters used in the parts of the subpattern, with notes on their usage.

0 = a digit

= a digit, zero shows as absent

`.` = a period (.) is a placeholder for decimal separator

`,` = a comma (,) is a placeholder for grouping separator

`;` = a semicolon (;) separates formats

`-` = a minus sign/dash (-) is the default negative prefix

`%` = divide by 100 and show as percentage

`x` = any other characters can be used in the prefix or suffix

`'` = a single quote (') is used to quote special characters in a prefix or suffix

If there is no explicit negative subpattern, a minus sign (-) is prefixed to the positive form (i.e., "0.00" alone is equivalent to "0.00;-0.00"). Illegal formats, such as "#.#.#" or mixing '_' and '*' in the same format, will cause a `ParseException`. The `ParseException` string will show where the error occurred. The grouping separator is commonly used for thousands, but in some countries for ten-thousands. The interval is a constant number of digits between the grouping characters, such as 100,000,000 or 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "#####,####" == "##,####,####". This class only handles localized digits where the 10 digits are contiguous in Unicode, from 0 to 9.

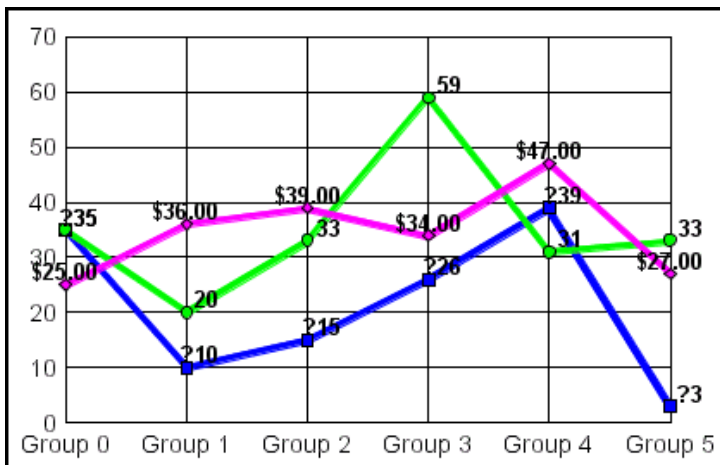
Return: `void`;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setCustomDataText(getSeries(2), true);
setCustomDataText(getSeries(1), true);
setCustomDataText(getSeries(0), true);
setCustomDataTextFormat(getSeries(2), 5);
setCustomDataTextFormat(getSeries(1), 1);
setCustomDataTextFormat(getSeries(0), -1);
setCustomDataTextFormatPattern(getSeries(0), "?##");
setDataTextDisplay(true);
setVisibleSeries(3);

```



Note: `setCustomDataTextFormat()` must assign -1 to the specified series in order for this custom format pattern to be used.

See [getCustomDataTextFormatPattern\(\)](#) on page 729, [setCustomDataText\(\)](#) on page 869, [setCustomDataTextFormat\(\)](#) on page 870.

setData()

This method sets the data value for a series/group intersection. This is the master method for setting data in graphs. You may also specify whether or not the graph should be repainted when the new value is assigned. You must use setDataRangeToExtent() or setDataRange() after the last setData() in order to include the data in the graph.

Syntax:

```
void setData (int row,int col,double fValue);
void setData(int row, int col, double fValue, boolean bRecalc);
setData(int row, int col, Object obj)
setData(int row, int col, Object obj, boolean bRecalc)
setData(int row, int col, int nMonth, int nDay, int nYear)
```

where:

row

Is the row number.

col

Is the column number.

obj

Is the object ID.

fValue

Is the data value to set at the row or column.

bRecalc

Can be one of the following:

true (recalculate)

false (do not recalculate)

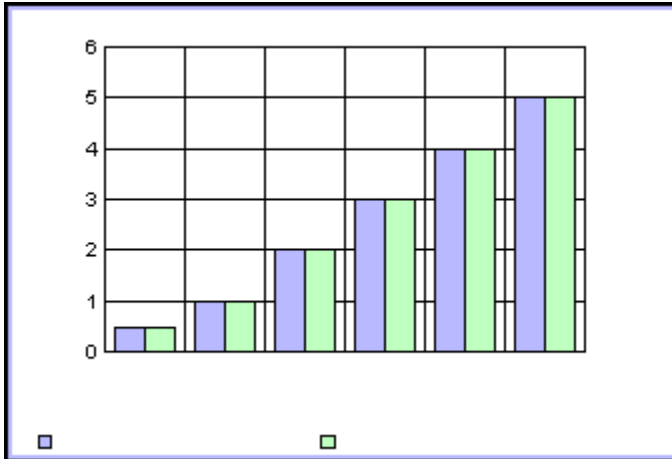
Return: *void*;

Example:

```

setUseSampleData(false); setData(0,0,.5); setData(0,1,1);
setData(0,2,2); setData(0,3,3); setData(0,4,4); setData(0,5,5);
setData(1,0,.5); setData(1,1,1); setData(1,2,2); setData(1,3,3);
setData(1,4,4); setData(1,5,5);
setDataRangeToExtent();

```



See [setDataRange\(\)](#) on page 877, [setDataRangeToExtent\(\)](#) on page 878.

setDataLabel()

This method sets a label string at a specified series (s) and group (g).

Syntax:

```
void setDataLabel (int s,int g, String newValue);
```

where:

s

Is a series number.

g

Is a group number.

newValue

Is a data label string.

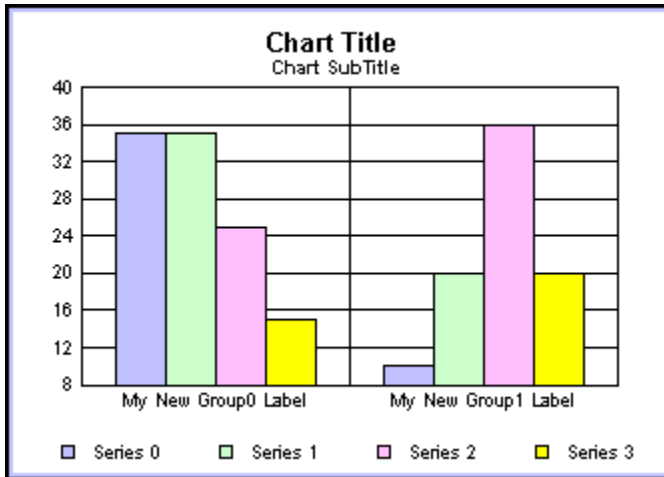
Return: void;

Example:

```

setDataLabel(0,0, "My New Group 0 Label"); setDataLabel(0,1, "My New Group
1 Label");
setGroupLabel(0, getDataLabel(0,0));
setGroupLabel(1, getDataLabel(0,1));
setDepthRadius(0);
setDataRange(0,0,3,1);

```



See [getDataLabel\(\)](#) on page 733, [getGroupLabel\(\)](#) on page 761, [setGroupLabel\(\)](#) on page 914, [getSeriesLabel\(\)](#) on page 809, [setSeriesLabel\(\)](#) on page 979.

setDataRange()

This method is used to establish the range of data that will be used in a graph.

Syntax:

```

void setDataRange(int rowStop,int colStop);
void setDataRange(int rowStart, int colStart,
    int rowStop, int colStop);

```

where:

rowStop

Is the last row number.

colStop

Is the last column number.

rowStart

Is the first row number (0 if not specified).

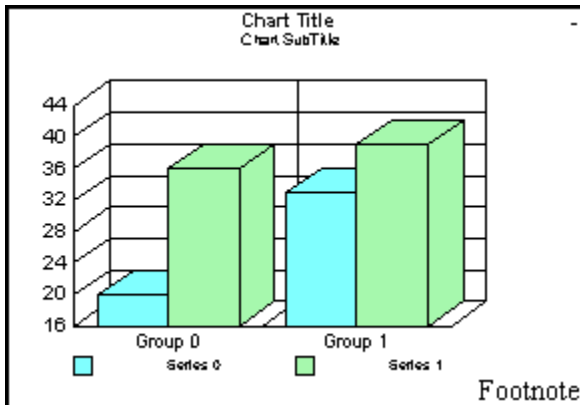
colStart

Is the first column number (0 if not specified).

Return: *void*;

Example:

```
setDataRange(1,1,2,2);
```



Note: When x-axis labels are defined, but no riser values exist for those groups, the labels will not display, by default, if the missing points are for the *last* groups on the chart. If you want to display those labels, use `setDataRange()` to specify the range of groups you want to display.

See [setData\(\)](#) on page 875, [setDataRangeToExtent\(\)](#) on page 878.

setDataRangeToExtent()

This method sets the range of data to be used in the graph according to the number of values identified by the `setData()` method. It will override values that may be set by `setDataRange()`.

Syntax:

```
void setDataRangeToExtent ();
```

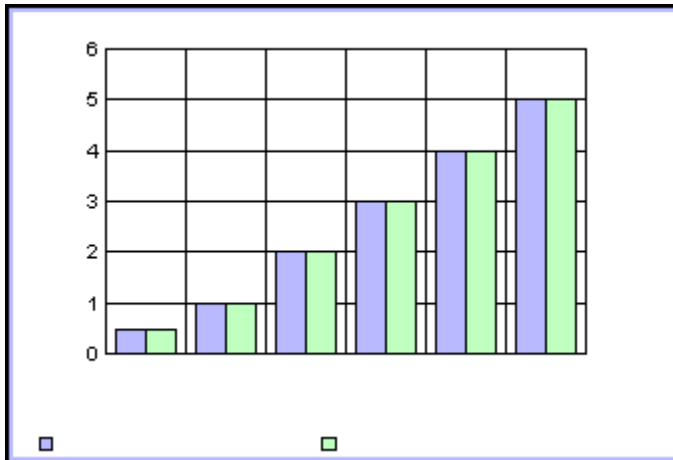
Return: *void*;

Example:

```

setUseSampleData(false);
setDepthRadius(0);
setData(0,0,.5); /* Row 0/Col 0 */
setData(0,1,1); /* Row 0/Col 1 */
setData(0,2,2); /* Row 0/Col 2 */
setData(0,3,3); /* Row 0/Col 3 */
setData(0,4,4); /* Row 0/Col 4 */
setData(0,5,5); /* Row 0/Col 5 */
setData(1,0,.5); /* Row 1/Col 0 */
setData(1,1,1); /* Row 1/Col 1 */
setData(1,2,2); /* Row 1/Col 2 */
setData(1,3,3); /* Row 1/Col 3 */
setData(1,4,4); /* Row 1/Col 4 */
setData(1,5,5); /* Row 1/Col 5 */  setDataRangeToExtent();

```



See [setData\(\)](#) on page 875, [setDataRange\(\)](#) on page 877.

setDataTextAngle()

Method sets the angle from center point that all/selected data text is drawn from in graph.

Syntax:

```

void setDataTextAngle(IdentObj id,int newValue);
void setDataTextAngle(int seriesID,int newValue);
void setDataTextAngle(int seriesID,int groupID,int newValue);

```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getDataText()`).

seriesID

Is a series number.

groupID

Is a group number.

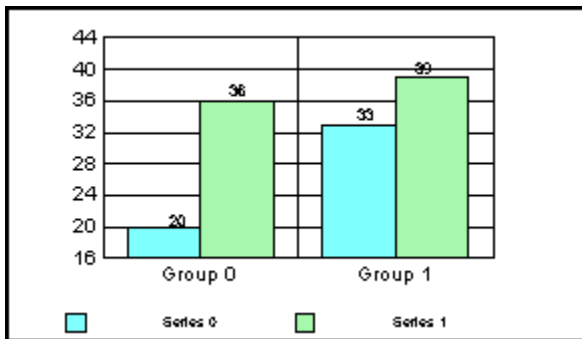
newValue

Is a data text angle (0...360).

Return: *void*;

Example:

```
setViewableGroups(2);
setViewableSeries(2);
setDataScrollerPresenceGroups(0);
setDataTextDisplay(true);
setDataTextPosition(0);  setDataTextAngle(0,0,30);
setDataTextAngle(0,1,60); setDataTextAngle(1,0,90);
setDataTextAngle(1,1,120);
setDepthRadius(0);
```



Note: `DataTextPosition` must be set to zero and `DataTextDisplay` must be set to true in order for data text to be drawn in the graph.

See [getDataTextAngle\(\)](#) on page 733, [getDataTextRadius\(\)](#) on page 734, [setDataTextRadius\(\)](#) on page 881, [DataTextAngleDefault](#) on page 314, [DataTextDisplay](#) on page 316, [DataTextPosition](#) on page 322.

setDataTextRadius()

These methods set the radius for the data text position.

Syntax:

```
void setDataTextRadius(IdentObj id,int newValue);  
void setDataTextRadius(int seriesID,int newValue);  
void setDataTextRadius(int seriesID,int groupID,  
                        int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getDataText()).

seriesID

Is a series number.

groupID

Is a group number.

newValue

Is a data text radius (0...100).

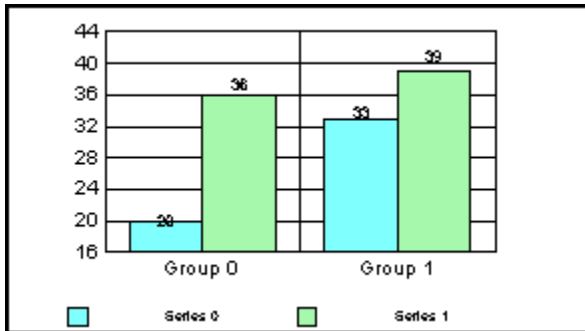
Return: *void*;

Example:

```

setViewableGroups(2);
setViewableSeries(2);
setDataScrollerPresenceGroups(0);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextRadius(0,0,0);
setDataTextRadius(0,1,10);
setDataTextRadius(1,0,20);
setDataTextRadius(1,1,30);
setDepthRadius(0);

```



Note: The `DataTextPosition` property must be set to zero. The `DataTextDisplay` property must be set to true in order for data text to be drawn in the graph.

See [getDataTextRadius\(\)](#) on page 734, [getDataTextAngle\(\)](#) on page 733, [setDataTextAngle\(\)](#) on page 879, [DataTextRadiusDefault](#) on page 324, [DataTextDisplay](#) on page 316, [DataTextPosition](#) on page 322.

setDataValue()

This method sets a data value for a specific data object (riser) in a graph.

Syntax:

```
void setDataValue(IdentObj id, double newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeriesGroup()`).

newValue

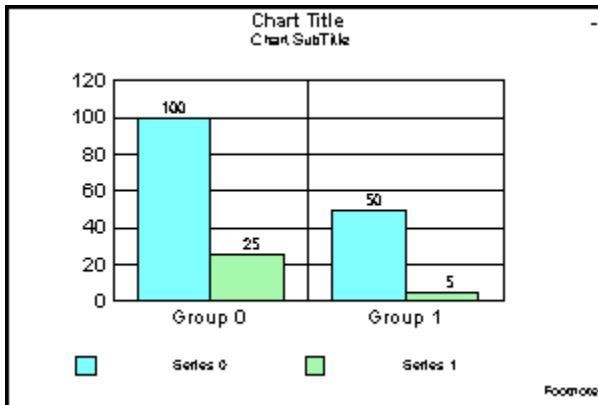
Is a data value to be assigned to the object.

Return: *void*;

Example:

```
setDataRange(1,1,2,2);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextRadius(0,0,20);
setDataTextRadius(0,1,20);
setDataTextRadius(1,0,20);
setDataTextRadius(1,1,20);  setDataValue(getSeriesGroup(0,0),100);
setDataValue(getSeriesGroup(0,1),50);
setDataValue(getSeriesGroup(1,0),25);  setDataValue(getSeriesGroup(1,1),5);

setDepthRadius(0);
```



See [getDataValue\(\)](#) on page 735.

setDisplay()

This method sets the display attribute of an object. If an object is not identified by id or objectID, this method will set the display attribute for the first item in the selection list.

Syntax:

```
void setDisplay(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Title()).

newValue

Can be true or false. If the value is:

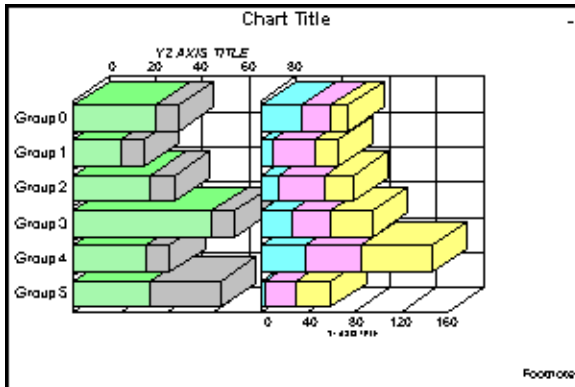
`true`, the object is imaged.

`false`, the object is not imaged.

Return: `void`;

Example:

```
setDisplay(getLegendArea(), false);
setY1TitleString("Y1 AXIS TITLE"); setDisplay(getY1Title(), true);
setDisplay(getSubtitle(), false);
setY2TitleString("Y2 AXIS TITLE"); setDisplay(getY2Title(), true);
```



See [getDisplay\(\)](#) on page 736, [DataTextDisplay](#) on page 316, [Display3DFloor](#) on page 336, [Display3DRightWall](#) on page 340, [Display3DLeftWall](#) on page 338, [DualAxisLineDisplay](#) on page 350, [FootnoteDisplay](#) on page 356, [FrameDisplay](#) on page 362, [getStackConnectLine\(\)](#) on page 1129, and many other object specific display properties.

setDisplayOffScale()

This method can be used to set whether or not off scale data points are imaged in a graph.

Syntax:

```
void setDisplayOffscale(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Axis()).

newValue

Is true or false. If the value is:

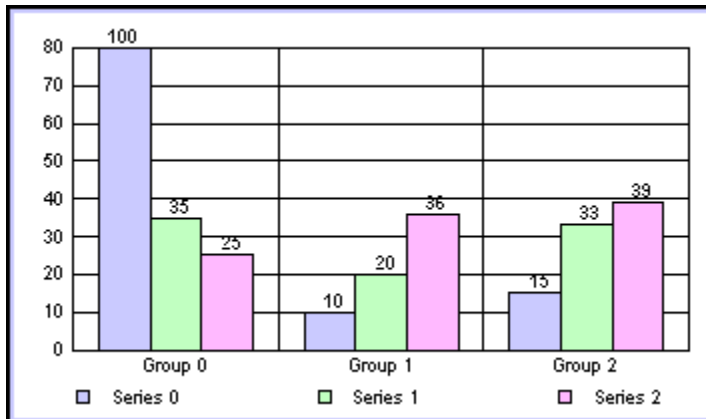
`true`, the off scale values are imaged.

`false`, the offscale values are not imaged.

Return: `void`;

Example:

```
setDepthRadius(0);
setDataValue(getSeriesGroup(0,0),100);
setDataRange(0,0,2,2);
setDataTextDisplay(true);  setDisplayOffScale(getY1Axis(),true);
```



Note: You may also use one of the object-specific properties to set the off-scale attribute of a specific object (for example, `X1OffScaleDisplay()`, `Y1OffScaleDisplay()`, and so on).

See [getDisplayOffScale\(\)](#) on page 737, [X1OffScaleDisplay](#) on page 606, [Y1OffScaleDisplay](#) on page 642, [Y2OffScaleDisplay](#) on page 680.

setDrawTicksPerpendicular()

When a 2.5D depth affect is applied to a 2D graph with the `DepthAngle` and `DepthRadius` properties and independent tick marks (not grid lines) are drawn, this method can be used to force the tick marks to be drawn perpendicular to the axis.

Syntax:

```
void setDrawTicksPerpendicular(
    IdentObj id,
    boolean newValue)
```

where:

id

Is an object ID returned by a getObjectID() method. It must be one of the tick mark objects (for example, getY1MajorTick()).

newValue

Is a boolean where a value of

true draws tick marks perpendicular to axis

false ticks may be drawn on an angle consistent with the frame

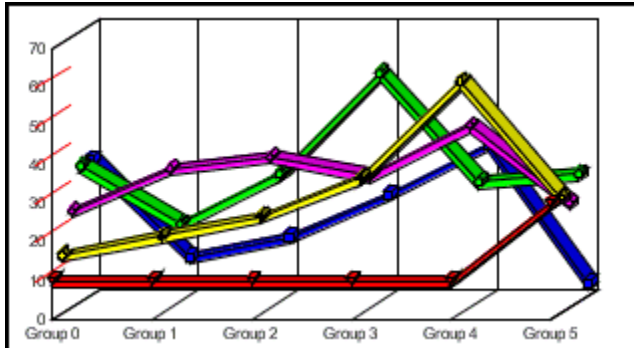
Return: *void*;

Example:

```

setGraphType(41);
setY1MajorGridDisplay(false);
setY1MajorTickDisplay(true);
setTickLength(getY1MajorTick(), 2000);
setDrawTicksPerpendicular(getY1MajorTick(), false);

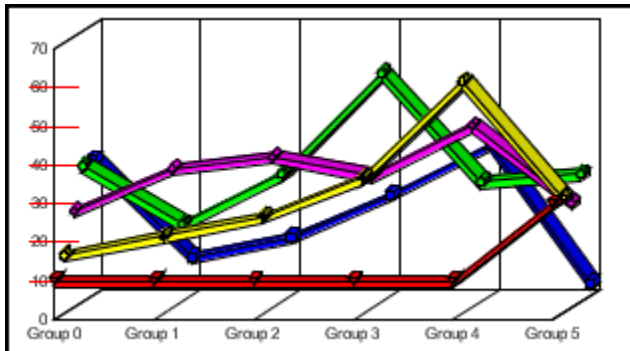
```



```

setDrawTicksPerpendicular(getY1MajorTick(), true);

```



See [getDrawTicksPerpendicular\(\)](#) on page 739.

setExceptionalRiser()

This property can be used to specify an exceptional riser. An exceptional riser can be manipulated separately (for example, colored) from other risers in the series.

Syntax:

```

void setExceptionalRiser();
void setExceptionalRiser(int s, int g);

```

where:

s

Is an optional series number.

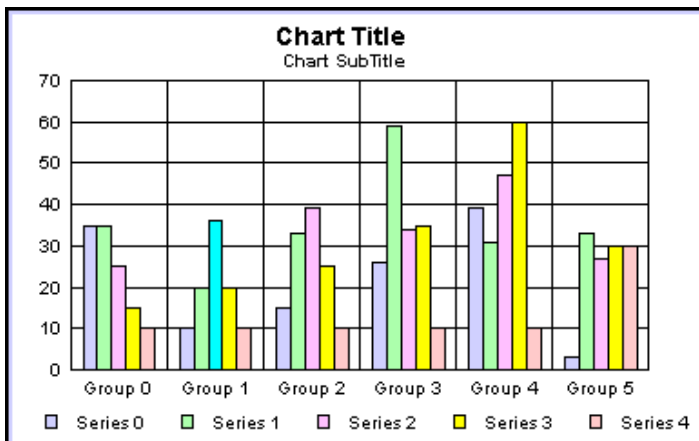
g

Is an optional group number.

Return: *void*;

Example:

```
setExceptionalRiser(2,1);
setFillType(getExceptionalRiser(2,1),1);
setFillColor(getExceptionalRiser(2,1),new Color(0,255,255));
setDepthRadius(0);
```



Note: If an object ID (id) is not specified, the exceptional riser attribute is assigned to the first item in the selection list (if applicable).

See [setNoExceptionalRiser\(\)](#) on page 944.

setExcludeMaxLabel()

This method can be used to exclude the maximum label from an ordinal or numeric axis. You may also use one of the axis-specific properties to exclude the maximum label.

Syntax:

```
void setExcludeMaxLabel(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Label()).

newValue

Is true or false. If the value is:

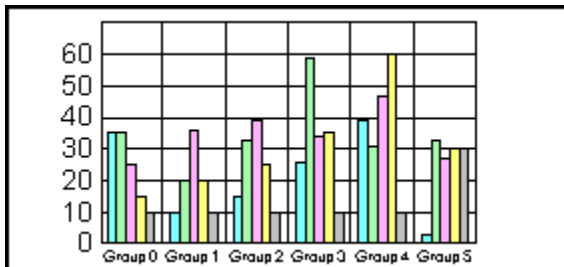
true, the maximum label is excluded.

false, the maximum label is included.

Return: void;

Example:

```
setDepthRadius(0);
setLegendDisplay(false); setExcludeMaxLabel(getY1Label(), true);
```



See [getExcludeMaxLabel\(\)](#) on page 744, [getExcludeMinLabel\(\)](#) on page 745, [setExcludeMinLabel\(\)](#) on page 889, [O1ExcludeMaxLabel](#) on page 428, [O2ExcludeMaxLabel](#) on page 454, [X1ExcludeMaxLabel](#) on page 579, [Y1ExcludeMaxLabel](#) on page 621, [Y2ExcludeMaxLabel](#) on page 657.

setExcludeMinLabel()

This method can be used to exclude the minimum label on an ordinal or numeric axis.

Syntax:

```
void setExcludeMinLabel(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1Label()).

newValue

Is true or false. If the value is:

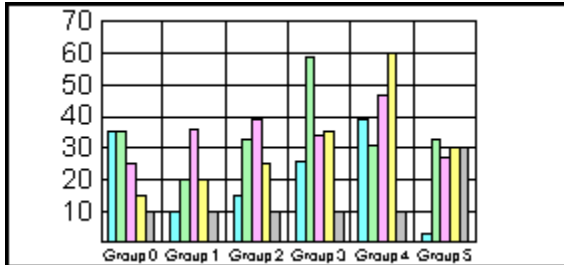
`true`, the minimum label is excluded.

`false`, the minimum label is included.

Return: `void`;

Example:

```
setDepthRadius(0);
setLegendDisplay(false); setExcludeMinLabel(getY1Label(), true);
```



Note: You may also use one of the axis-specific properties to exclude the minimum label.

See [getExcludeMinLabel\(\)](#) on page 745, [getExcludeMaxLabel\(\)](#) on page 744, [setExcludeMaxLabel\(\)](#) on page 888, [O1ExcludeMinLabel](#) on page 429, [O2ExcludeMinLabel](#) on page 456, [X1ExcludeMinLabel](#) on page 581, [Y1ExcludeMinLabel](#) on page 622, [Y2ExcludeMinLabel](#) on page 658.

setFillColor()

This method assigns a fill color to an area object.

Syntax:

```
void setFillColor(new Color (newValue));
void setFillColor(IdentObj id, new Color (newValue));
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getChartBackground()`).

newValue

Is the value of the fill color, in one of the following formats:

r, g, b [,t]

r, g, and b are the desired intensities of red, green, and blue, respectively. There can also be an optional transparency value. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense. For the transparency value, 0 means fully transparent, and 255 means fully opaque.

#hexcolor

Is the hexadecimal color code, preceded by a pound sign (#).

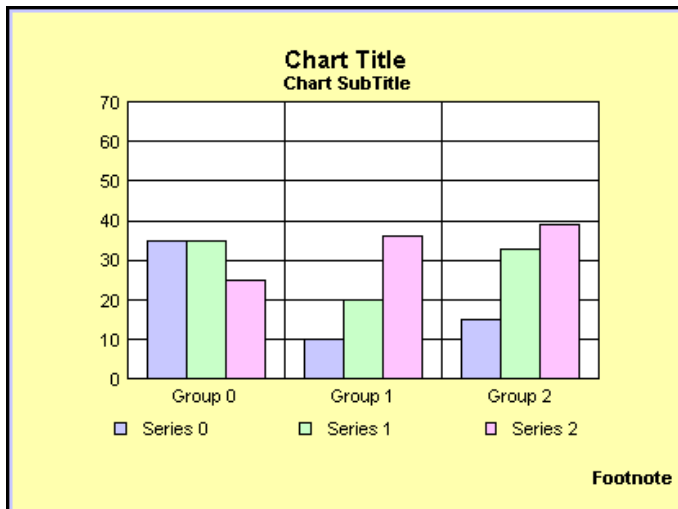
Return: *void*;

Example:

```
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);
  setFillColor(getChartBackground(), new Color(255,255,174));
setDepthRadius(0);
```

or, using a hexadecimal color code:

```
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);  setFillColor(getChartBackground(), new
  Color(#FFFAE));
setDepthRadius(0);
```



Note:

- ❑ If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).
- ❑ The fill type must be set to color with `setFillType(1);`

See [getFillColor\(\)](#) on page 746, [getBorderColor\(\)](#) on page 723, [setBorderColor\(\)](#) on page 860, [getFillType\(\)](#) on page 747, [setFillType\(\)](#) on page 892, [getTransparentFillColor\(\)](#) on page 825, [setTransparentFillColor\(\)](#) on page 1022, [setSeriesFillColor\(\)](#) on page 977.

setFillType()

This method assigns a fill type to an area object.

Syntax:

```
void setFillType(int newValue);  
void setFillType(IdentObj id, int newValue);
```

where:

id

Is the object ID returned by a `getObjectID()` method (for example, `getSeries()`).

newValue

Is one of the following fill types:

- 1 (color)
- 2 (gradient)
- 3 (texture)

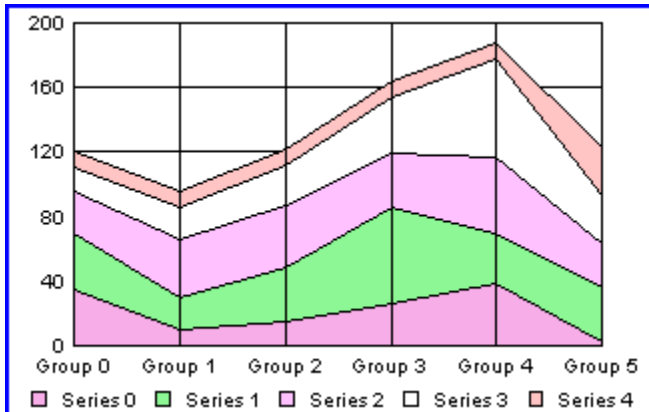
Return: *void*;

Example:

```

setDepthRadius(0);
setGraphType(32); setFillType(getSeries(0), 1);
setFillColor(getSeries(0), new Color(245,175,224));
setFillType(getSeries(1), 1);
setFillColor(getSeries(1), new Color(140,246,147));
setFillType(getSeries(3), 1);
setFillColor(getSeries(3), new Color(255,255,255));

```

**Note:**

- ❑ If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).
- ❑ If you select fill type 2 (gradient), use the setGradientDirection(), setGradientNumPins(), get/setGradientPinLeftColor(), GradientPinPosition(), setGradientPinRightColor(). If you select fill type 3 (textures), use setTextureURL() and setTextureDisplayMode() methods to define the texture.

See [setGradientDirection\(\)](#) on page 902, [setGradientNumPins\(\)](#) on page 903, [setGradientPinLeftColor\(\)](#) on page 904, [setGradientPinPosition\(\)](#) on page 906, [setGradientPinRightColor\(\)](#) on page 908, [setTextureURL\(\)](#) on page 1007, [getTextureDisplayMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004.

setFontName()

This method assigns a font name to a text object.

Syntax:

```
void setFontName(IdentObj id, String newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getFootnote()).

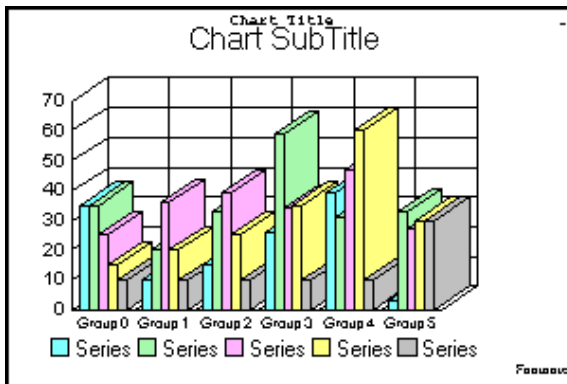
newValue

Is a font name string.

Return: void;

Example:

```
setFontName(getFootnote(),"TimesRoman");
setFontName(getLegendText(),"Dialog");
setFontName(getSubtitle(),"DialogInput");
setFontName(getTitle(),"Courier");
```



See [getFontName\(\)](#) on page 748, [getFontSizeVC\(\)](#) on page 751, [setFontSizeVC\(\)](#) on page 898, [getFontStyle\(\)](#) on page 752, [setFontStyle\(\)](#) on page 899, [setFontSize\(\)](#) on page 894.

setFontSize()

This method assigns a font size to a text object. If absolute font size is disabled for the object (that is, setFontSizeAbsolute(object, false), you must use setFontSizeVC() to set the object's font size in virtual coordinates. This method sets the font size in points.

Syntax:

```
void setFontSize (int newValue);
```

```
void setFontSize (IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getTitle()`).

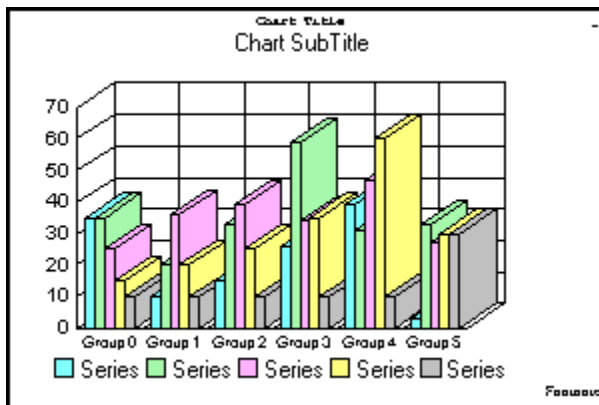
newValue

Is a font size value (in points).

Return: *void*;

Example:

```
setFontName(getFootnote(),"TimesRoman");
setFontName(getLegendText(),"Dialog");
setFontName(getSubtitle(),"DialogInput");
setFontName(getTitle(),"Courier"); setFontSize(10);
```



Note:

- ❑ If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).
- ❑ This method disables text autofitting.
- ❑ This method has been replaced by `setFontSizeInPoints()` when `setFontSizeAbsolute()` is set to true for a text object.

See [setFontSizeAbsolute\(\)](#) on page 896, [setFontSizeVC\(\)](#) on page 898.

setFontSizeAbsolute()

This method assigns absolute font sizing to a specific text object. If `newValue` is set to `true`, use `setFontSizeInPoints()` to set a text object's font size in points. If `newValue` is set to `false`, use `setFontSizeVC()` to set a text object font size in virtual coordinates.

Note: When PDE is enabled with `setPlace(true)`, add the `setPlaceResize();` command.

Syntax:

```
void setFontSizeAbsolute(IdentObj id, boolean newValue)
```

where:

newValue

Is true or false.

If the value is:

`true`, the font size is absolute. Use `setFontSizeInPoints()` to set the font size in points.

`false`, the font size is not absolute. Use `setFontSizeVC()` to set the font size in virtual coordinates.

id

Is the object ID returned by a `getObjectID()` method for a text object (for example, `getTitle()`).

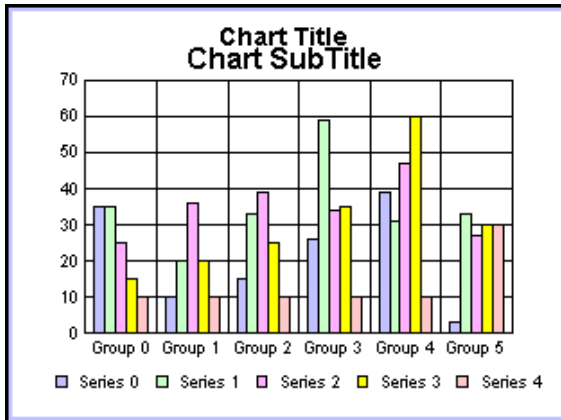
Return: `void`;

Example:

```

setFontSizeAbsolute(getTitle(), true);
setFontSizeInPoints(getTitle(), 14);  setFontSizeAbsolute(getSubtitle(),
false);
setFontSizeVC(getSubtitle(), 2000);
setDepthRadius(0);

```



Note: If an object ID (id) is not specified, `newValue` is assigned to the first item in the selection list (if applicable).

See [setFontSizeInPoints\(\)](#) on page 897, [setFontSizeVC\(\)](#) on page 898, [setPlaceResize\(\)](#) on page 954.

setFontSizeInPoints()

This method assigns a font size to a text object. If absolute font size is disabled for the object (that is, `setFontSizeAbsolute(object, false)`), you must use `setFontSizeVC()` to set the object's font size in virtual coordinates. This method sets the font size in points.

Syntax:

```
void setFontSizeInPoints(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getTitle()`)

newValue

Is a font size value in points (1...128). The default value is 12.

Return: `void`;

See [setFontSizeAbsolute\(\)](#) on page 896 for an example.

Note:

- ❑ This method disables text autofitting.
- ❑ This method replaces `setFontSize()` when `setFontSizeAbsolute()` is true for a specified object.

setFontSizeVC()

This method sets a text object's font size in virtual coordinates. If font size absolute is enabled with `setFontSizeAbsolute(object, true)`, you must use `setFontSize()` to set the font size in points. This method sets the font size in virtual coordinates.

Syntax:

```
void setFontSizeVC(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getTitle()`).

newValue

Is the font size in virtual coordinates.

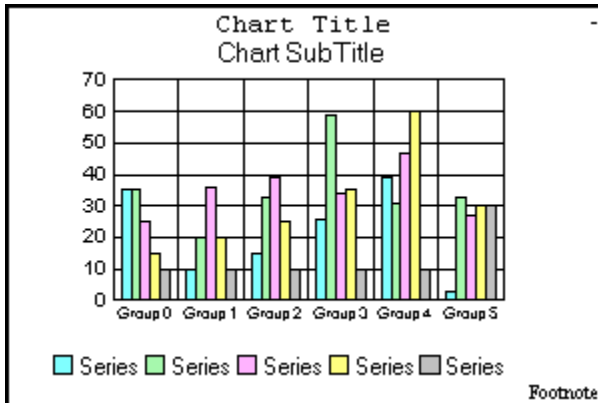
Return: *void*;

Example:

```

setAutofit(getFootnote(), false);
setAutofit(getLegendText(), false);
setAutofit(getSubtitle(), false);
setAutofit(getTitle(), false);  setFontSizeVC(getY1Label(), 1391);
setFontSizeVC(getLegendText(), 1159);  setFontSizeVC(getTitle(), 2086);
setFontSizeVC(getO1Label(), 927);  setFontSizeVC(getSubtitle(), 2086);
setFontSizeVC(getFootnote(), 1391);

```



Note: See [Virtual Coordinate System](#) on page 47 if you need additional information about the values that can be set by this method. If autofitting is enabled for the text object, the font size setting will be ignored.

See [setFontSizeAbsolute\(\)](#) on page 896, [getFontSizeVC\(\)](#) on page 751, [setFontSize\(\)](#) on page 894.

setFontStyle()

This method assigns a font style to a text object.

Syntax:

```
void setFontStyle(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getTitle()).

newValue

Is the font style (0...7). Values can be:

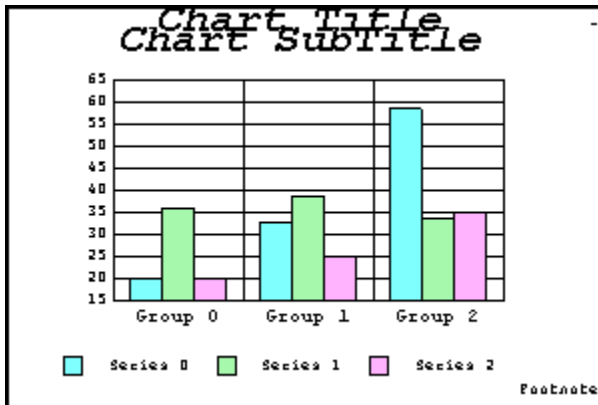
❑ 0 = Plain

- ☐ 1 = *Italic*
- ☐ 2 = **Bold**
- ☐ 3 = ***Italic/Bold***
- ☐ 4 = Underline
- ☐ 5 = *Italic/Underline*
- ☐ 6 = **Bold/Underline**
- ☐ 7 = ***Bold/Italic/Underline***

Return: *void*;

Example:

```
setViewableGroups(3);
setViewableSeries(3);
setDataScrollerPresenceGroups(0);  setFontStyle(getSubtitle(),2);
setFontStyle(getTitle(),7);
```



See [getFontStyle\(\)](#) on page 752, [getFontSizeVC\(\)](#) on page 751, [setFontSizeVC\(\)](#) on page 898, [getFontName\(\)](#) on page 748, [setFontName\(\)](#) on page 893, [setFontSize\(\)](#) on page 894.

setGradientAutoDirectionMode()

This method controls whether gradient direction is automatically rotated when a chart changes orientation. When enabled, the direction of gradients are mapped into corresponding or complimentary directions based on the orientation of the chart. For example, a down gradient on a vertical bar chart would automatically become a right gradient on a horizontal bar chart. When disabled, the original gradient direction remains the same when chart is rotated.

Syntax:

```
void setGradientAutoDirectionMode(boolean);
```

where:

boolean

Is true or false. If the value is:

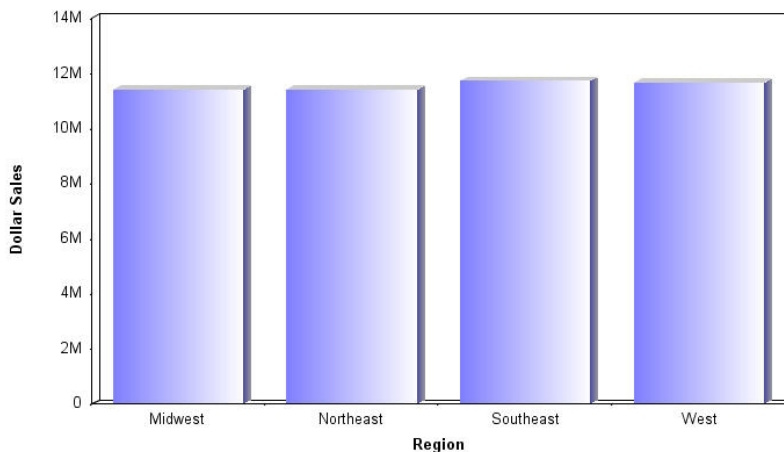
true, the gradient direction is automatically rotated.

false, the gradient direction is retained and not rotated with the chart. This is the default value.

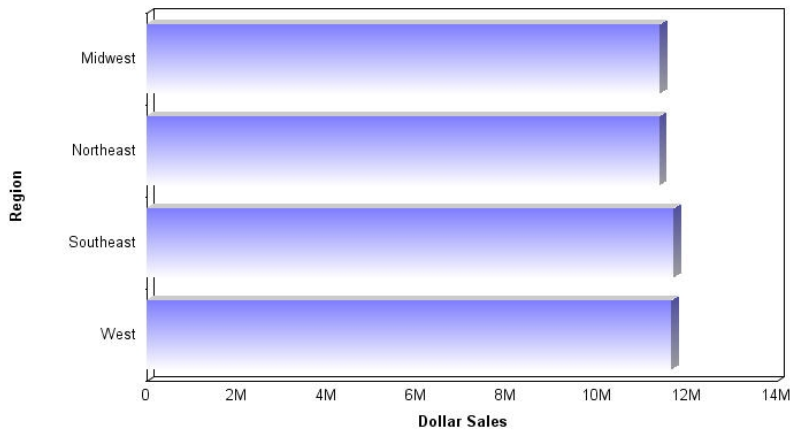
Return: *void*;

Example:

```
setColorMode(1);
setFillType(getSeries(0), 2);
setGradientNumPins(getSeries(0), 2);
setGradientPinPosition(getSeries(0), 0.0,0);
setGradientPinPosition(getSeries(0), 1.0,1);
setGradientPinLeftColor(getSeries(0),new Color(128,128,255), 0);
setGradientPinRightColor(getSeries(0),new Color(128,128,255), 0);
setGradientPinLeftColor(getSeries(0),new Color(128,128,255), 2);
setGradientPinRightColor(getSeries(0),new Color(128,128,255), 2);
setGradientAutoDirectionMode(true);
setGradientDirection(getSeries(0),1);
```



The chart in the following image shows the same gradient direction as the previous example, but rotated horizontally.



setGradientDirection()

This method sets the direction of a gradient that is applied to an object.

Syntax:

```
void setGradientDirection(int newValue);
void setGradientDirection(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries()).

newValue

Is a gradient direction (1...13, 16, and 17). Values can be:

1 = right	9 = radial
2 = left	10 = radial/top/left
3 = down	11 = radial/top/right
4 = up	12 = radial/bottom/left

5 = down/left	13 = radial/bottom/right
6 = up/left	16 = radial pie
7 = down/right	17 = inverted radial pie
8 = up/right	

Return: *void*;

Note:

- ❑ If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).
- ❑ The fill type for the selected object must be set to two with `setFillType(2)`.

See [getGradientDirection\(\)](#) on page 755, [getFillType\(\)](#) on page 747, [setFillType\(\)](#) on page 892, [setGradientNumPins\(\)](#) on page 903, [setGradientPinLeftColor\(\)](#) on page 904, [setGradientPinRightColor\(\)](#) on page 908, [setGradientPinPosition\(\)](#) on page 906.

setGradientNumPins()

This method sets the number of pins in a gradient that is applied to an object in a graph.

Syntax:

```
void setGradientNumPins(int newValue);
```

```
void setGradientNumPins(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries()`).

newValue

Is the number of pins in the gradient.

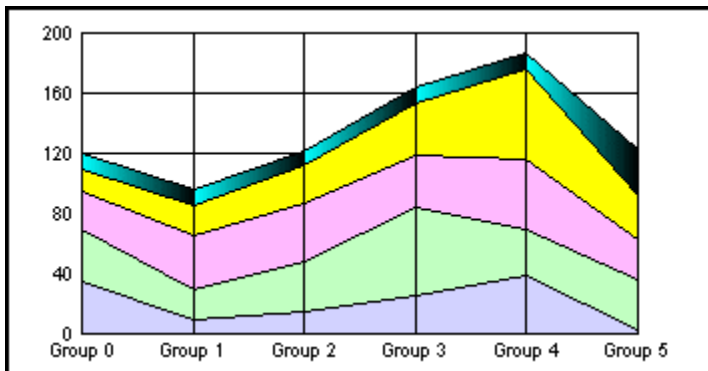
Return: *void*;

Example:

```

setGraphType(32);
setDepthRadius(0);
setFillType(getSeries(4),2);
setGradientDirection(getSeries(4),1); setGradientNumPins(getSeries(4),2);
setGradientPinLeftColor(getSeries(4), new Color(0,0,0),1);
setGradientPinLeftColor(getSeries(4), new Color(0,255,255),0);
setGradientPinPosition(getSeries(4),1.0,1);
setGradientPinPosition(getSeries(4),0.0,0);
setGradientPinRightColor(getSeries(4), new Color(0,0,0),1);
setGradientPinRightColor(getSeries(4), new Color(0,255,255),0);

```

**Note:**

- ❑ If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).
- ❑ The fill type for the selected object must be set to two with the method setFillType(2).

See [getGradientNumPins\(\)](#) on page 755, [setFillType\(\)](#) on page 892, [setGradientDirection\(\)](#) on page 902, [setGradientPinLeftColor\(\)](#) on page 904, [setGradientPinRightColor\(\)](#) on page 908, [setGradientPinPosition\(\)](#) on page 906.

setGradientPinLeftColor()

This method sets the color value of the left pin of a gradient that is applied to an object in a graph.

Syntax:

```
void setGradientPinLeftColor(new Color (newValue),int nIndex);
```

```
void setGradientPinLeftColor(
    IdentObj id, new Color (newValue), int nIndex);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries()).

nIndex

0...Number of pins in a gradient.

newValue

Is the color value of the left pin in a gradient in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

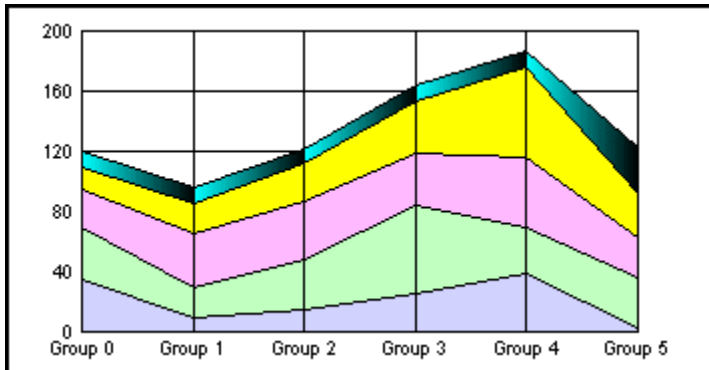
Return: *void*;

Example:

```
setGraphType(32);
setDepthRadius(0);
setFillType(getSeries(4),2);
setGradientDirection(getSeries(4),1);
setGradientNumPins(getSeries(4),2); setGradientPinLeftColor(getSeries(4),
    new Color(0,0,0),1); setGradientPinLeftColor(getSeries(4), new
Color(0,255,255),0);
setGradientPinPosition(getSeries(4),1.0,1);
setGradientPinPosition(getSeries(4),0.0,0);
setGradientPinRightColor(getSeries(4), new Color(0,0,0),1);
setGradientPinRightColor(getSeries(4), new Color(0,255,255),0);
```

or, using hexadecimal color codes:

```
setGraphType(32);
setDepthRadius(0);
setFillType(getSeries(4),2);
setGradientDirection(getSeries(4),1);
setGradientNumPins(getSeries(4),2); setGradientPinLeftColor(getSeries(4),
    new Color(#000000),1); setGradientPinLeftColor(getSeries(4), new
    Color(#00FFFF),0);
setGradientPinPosition(getSeries(4),1.0,1);
setGradientPinPosition(getSeries(4),0.0,0);
setGradientPinRightColor(getSeries(4), new Color(#000000),1);
setGradientPinRightColor(getSeries(4), new Color(#00FFFF),0);
```



Note:

- ❑ If an object ID (id) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).
- ❑ The fill type for the selected object must be set to two with `setFillType(2)`.

See [getGradientPinLeftColor\(\)](#) on page 756, [setFillType\(\)](#) on page 892, [setGradientDirection\(\)](#) on page 902, [setGradientNumPins\(\)](#) on page 903, [setGradientPinRightColor\(\)](#) on page 908, [setGradientPinPosition\(\)](#) on page 906.

setGradientPinPosition()

This method sets the pin position of a gradient that is applied to an object in a graph.

Syntax:

```
void setGradientPinPosition(double newValue,int nIndex);
void setGradientPinPosition(IdentObj id,double newValue, int nIndex);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1Label()).

nIndex

0...Number of pins in gradient.

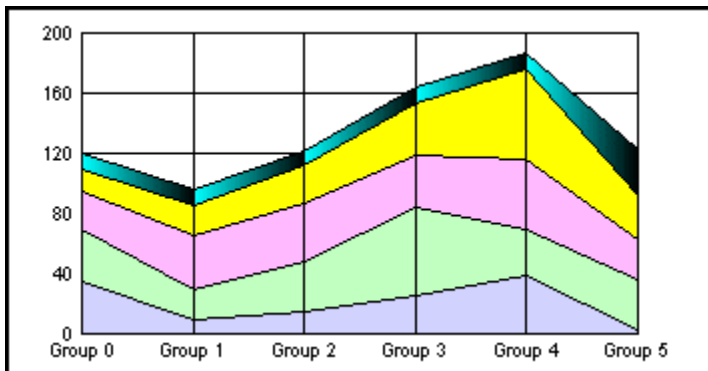
newValue

Is the position in the gradient (0.0 to 1.0).

Return: void;

Example:

```
setFillType(getSeries(4),2);
setGradientDirection(getSeries(4),1);
setGradientNumPins(getSeries(4),2);
setGradientPinLeftColor(getSeries(4), new Color(0,0,0),1);
setGradientPinLeftColor(GetSeries(4), new Color(0,0,255),0);
setGradientPinPosition(getSeries(4),1.0,1);
setGradientPinPosition(getSeries(4),0.0,0);
setGradientPinRightColor(getSeries(4), new Color(0,0,0),1);
setGradientPinRightColor(getSeries(4), new Color(0,0,255),0);
```



Note:

- ❑ If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).
- ❑ The fill type for the selected object must be set to two with setFillType(2).

See [getGradientPinPosition\(\)](#) on page 756, [setFillType\(\)](#) on page 892, [setGradientDirection\(\)](#) on page 902, [setGradientNumPins\(\)](#) on page 903, [setGradientPinRightColor\(\)](#) on page 908, [setGradientPinLeftColor\(\)](#) on page 904.

setGradientPinRightColor()

This method sets the color value of the right pin of a gradient.

Syntax:

```
void setGradientPinRightColor(new Color (newValue),int nIndex);  
void setGradientPinRightColor(IdentObj id, new Color (newValue), int  
nIndex);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries()).

nIndex

0...Number of pins in gradient.

newValue

Is the color value of the right pin in a gradient in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

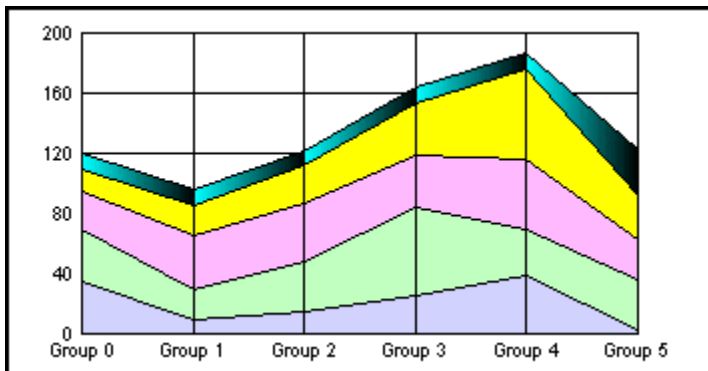
Return: *void*;

Example:

```
setFillType(getSeries(4),2);  
setGradientDirection(getSeries(4),1);  
setGradientNumPins(getSeries(4),2);  
setGradientPinLeftColor(getSeries(4), new Color(0,0,0),1);  
setGradientPinLeftColor(GetSeries(4), new Color(0,0,255),0);  
setGradientPinPosition(getSeries(4),1.0,1);  
setGradientPinPosition(getSeries(4),0.0,0);  
setGradientPinRightColor(getSeries(4), new Color(0,0,0),1);  
setGradientPinRightColor(getSeries(4), new Color(0,0,255),0);
```


or, using hexadecimal color codes:

```
setFillType(getSeries(4),2);
setGradientDirection(getSeries(4),1);
setGradientNumPins(getSeries(4),2);
setGradientPinLeftColor(getSeries(4), new Color(#000000),1);
setGradientPinLeftColor(GetSeries(4), new Color(#0000FF),0);
setGradientPinPosition(getSeries(4),1.0,1);
setGradientPinPosition(getSeries(4),0.0,0);
setGradientPinRightColor(getSeries(4), new Color(#000000),1);
setGradientPinRightColor(getSeries(4), new Color(#0000FF),0);
```



Note:

- ❑ If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).
- ❑ The fill type for the selected object must be set to two with setFillType(2).

See [getGradientPinRightColor\(\)](#) on page 756, [setFillType\(\)](#) on page 892, [setGradientDirection\(\)](#) on page 902, [setGradientNumPins\(\)](#) on page 903, [setGradientPinPosition\(\)](#) on page 906, [setGradientPinLeftColor\(\)](#) on page 904.

setGridCount()

This method can be used to set the number of minor grid lines that will appear between major grid lines.

Syntax:

```
void setGridCount(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1MinorGrid()).

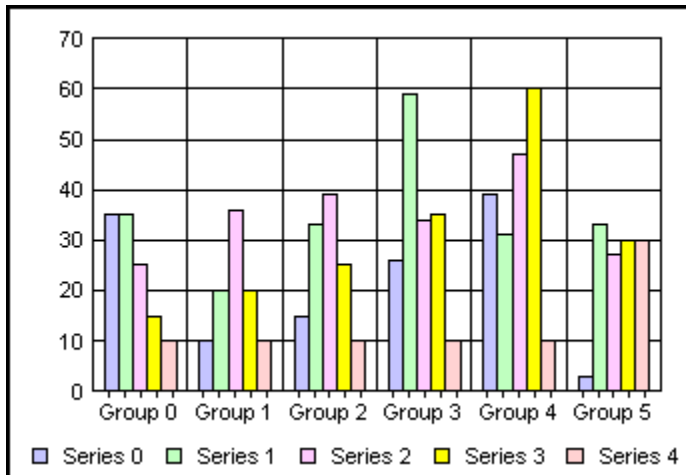
newValue

Is the number of grid lines for the axis object.

Return: void;

Example:

```
setDepthRadius(0);
setDisplay(getO1MinorGrid(), true);
setGridStyle(getO1MinorGrid(), 4); setGridCount(getO1MinorGrid(), 5);
```



Note: If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [getGridCount\(\)](#) on page 757, [O1MinorGridCount](#) on page 446.

setGridStep()

This method sets the number of grid steps to image on a numeric axis.

Syntax:

```
void setGridStep(IdentObj id, double newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getY1MajorGrid()`).

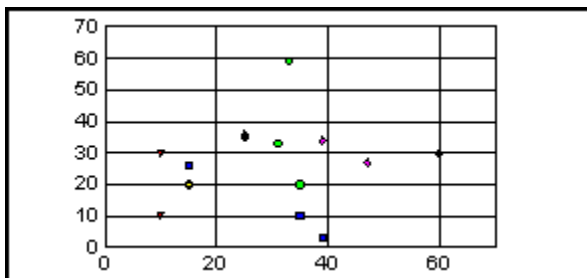
newValue

Is the number of grid steps on an axis.

Return: *void*;

Example:

```
setGraphType(61);  setGridStep(getX1MajorGrid(),20.0);
setGridStep(getY1MajorGrid(),10.0);
setX1MinorGridStepAuto(true);
setX1MajorGridStepAuto(false);
```



Note:

- ❑ You may also use one of the axis-specific properties to set the number of grid steps assigned on an axis (for example, `X1MajorGridStep`, `X1MinorGridStep`, `Y1MajorGridStep`, and so on).
- ❑ The value assigned by this method will be ignored if `setGridStepAuto()` enables (true) automatic calculation of grid steps on a particular axis.

See [getGridStep\(\)](#) on page 758, [getGridStepAuto\(\)](#) on page 759, [setGridStepAuto\(\)](#) on page 911, [X1MajorGridStep](#) on page 592, [X1MajorGridStepAuto](#) on page 594, [X1MinorGridStep](#) on page 599, [X1MinorGridStepAuto](#) on page 601, [Y2MajorGridStep](#) on page 668, [Y2MajorGridStepAuto](#) on page 670, [Y2MinorGridStep](#) on page 675, [Y2MinorGridStepAuto](#) on page 676, [Y1MajorGridStep](#) on page 632, [Y1MajorGridStepAuto](#) on page 633, [Y1MinorGridStepAuto](#) on page 638.

setGridStepAuto()

This method enables/disables automatic calculation of grid steps on a numeric axis.

Syntax:

```
void setGridStepAuto(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1MajorGrid()).

newValue

Is true or false. If the value is:

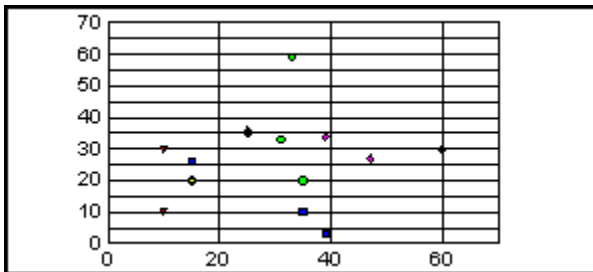
true, grid steps are automatically calculated.

false, grid steps are not automatically calculated.

Return: void;

Example:

```
setGraphType(61);
setX1MajorGridStep(20.0);
setY1MinorGridStep(5.0);
setY1MajorGridStep(10.0);  setGridStepAuto(getX1MinorGrid(), true);
setGridStepAuto(getY1MinorGrid(), false);
setGridStepAuto(getX1MajorGrid(), false);
setGridStepAuto(getY1MajorGrid(), true);
```

**Note:**

- ❑ You may also use one of the axis-specific properties to set whether or not grid steps are automatically calculated on an axis (for example, X1MajorGridStepAuto, X1MinorGridStepAuto, Y1MajorGridStepAuto, and so on).
- ❑ If this method sets a value of true, any value assigned by setGridStep() will be ignored.

See [getGridStepAuto\(\)](#) on page 759, [getGridStep\(\)](#) on page 758, [setGridStep\(\)](#) on page 910, [X1MajorGridStepAuto](#) on page 594, [X1MinorGridStepAuto](#) on page 601, [Y2MajorGridStepAuto](#) on page 670, [Y2MinorGridStepAuto](#) on page 676, [Y1MajorGridStepAuto](#) on page 633, [Y1MinorGridStepAuto](#) on page 638.

setGridStyle()

This method selects the style of grids on an axis.

Syntax:

```
void setGridStyle(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getX1MajorGrid()).

newValue

Is a grid style (0...5). Values can be:

- ☐ 0 = No Grid
- ☐ 1 = Normal Grid
- ☐ 2 = Grid and Ticks
- ☐ 3 = Inner Ticks
- ☐ 4 = Outer Ticks
- ☐ 5 = Spanned Ticks

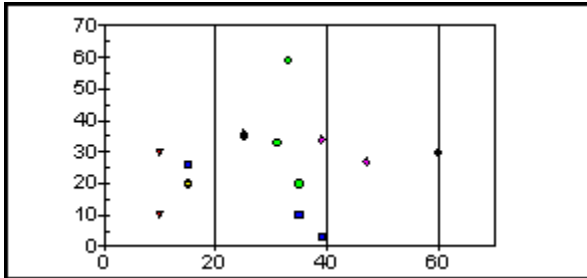
Return: *void*;

Example:

```

setY1MinorGridDisplay(true);
setGraphType(61);
setX1MajorGridStep(20.0);
setY1MinorGridStep(5.0);
setY1MajorGridStep(10.0);
setY1MinorGridStepAuto(false);
setX1MajorGridStepAuto(false);  setGridStyle(getX1MinorGrid(),0);
setGridStyle(getX1MajorGrid(),1);  setGridStyle(getY1MinorGrid(),2);
setGridStyle(getY1MajorGrid(),4);

```



See [getGridStyle\(\)](#) on page 760, [O1MajorGridStyle](#) on page 444, [O1MinorGridStyle](#) on page 449, [X1MajorGridStyle](#) on page 596, [X1MinorGridStyle](#) on page 603, [Y1MajorGridStyle](#) on page 634, [Y1MinorGridStyle](#) on page 639, [Y2MajorGridStyle](#) on page 671, [Y2MinorGridStyle](#) on page 677.

setGroupLabel()

This method can be used to define group labels.

Syntax:

```

void setGroupLabel(IdentObj id, String newValue);
void setGroupLabel(int g, String newValue);

```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getGroup(0)`).

g

Is a group number.

newValue

Is a group label string.

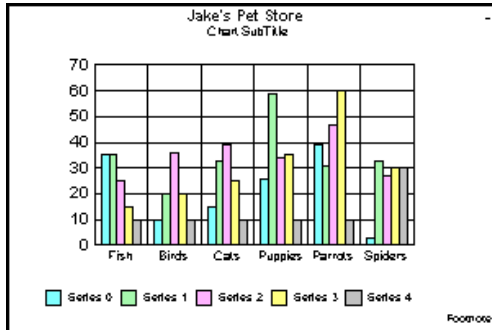
Return: *void*;

Example:

```

setDepthRadius(0);  setGroupLabel(0,"Fish");  setGroupLabel(1,"Birds");
setGroupLabel(2,"Cats");  setGroupLabel(3,"Puppies");
setGroupLabel(4,"Parrots");  setGroupLabel(5,"Spiders");
setTitleString("Jake's Pet Store");

```



See [getDataLabel\(\)](#) on page 733, [setDataLabel\(\)](#) on page 876, [getSeriesLabel\(\)](#) on page 809, [setSeriesLabel\(\)](#) on page 979.

setIgnoreSeries()

This method can be used to ignore (do not draw) a particular series in a graph.

Syntax:

```

void setIgnoreSeries(IdentObj id, boolean newValue);
void setIgnoreSeries(int seriesID, boolean newValue);

```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries(1)`).

seriesID

Is a series number.

newValue

Is true or false. If the value is:

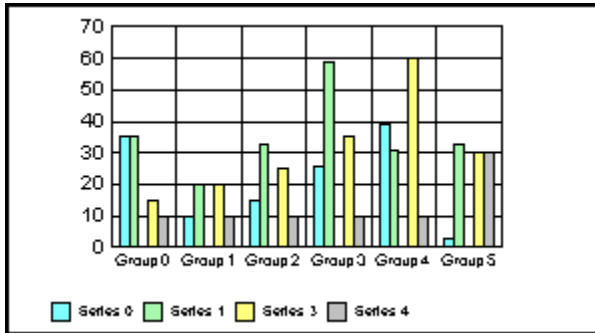
true, the identified series (s) is ignored (not drawn).

false, the identified series (s) is restored (drawn).

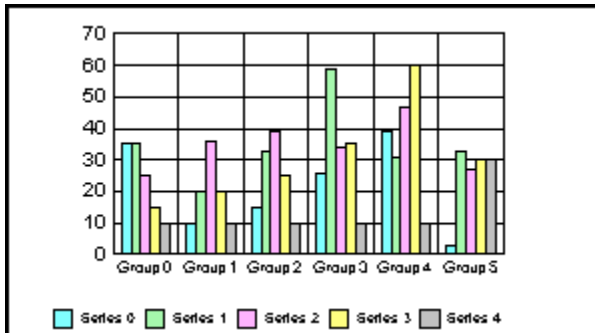
Return: `void`;

Example:

```
setDepthRadius(0);  setIgnoreSeries(2,true);
```



```
setIgnoreSeries(2,false);
```

**setInset()**

This method defines the inset of an annotation, title box, subtitle box, or footnote box object.

Syntax:

```
void setInset(IdentObj anObject, int theInsets)
```

```
void setInset(IdentObj anObject, Insets theInsets)
```

where:

anObject

Is an Object ID returned by a getObject() method.

theInset

Is a uniform inset for all sides of an object.

Return: void;

Note: Currently supported objects include: an annotation object (for example, `getAnnotationBox(0)`), the title box object (that is, `getTitleBox()`), the subtitle box object (that is, `getSubtitleBox()`), or the footnote box object (that is, `getFootnoteBox()`).

See [getInsets\(\)](#) on page 763.

set...Insets

These methods set the top/bottom and right/left insets of borders of a specified object. The default value is 1 (which is essentially off).

Syntax:

```
setBottomInset(IdentObj idObj, int newValue);
setLeftInset(IdentObj idObj, int newValue);
setRightInset(IdentObj idObj, int newValue);
setTopInset(IdentObj idObj, int newValue);
```

where:

idObj

Is the object ID returned by `getAnnotationBox()`, `getBeveledLegend()`, `getBeveledLegendMarker()`, `getBeveledRiser()`, `getFootnoteBox()`, `getSubtitleBox()`, or `getTitleBox()`.

newValue

Is the inset of the border in virtual coordinates.

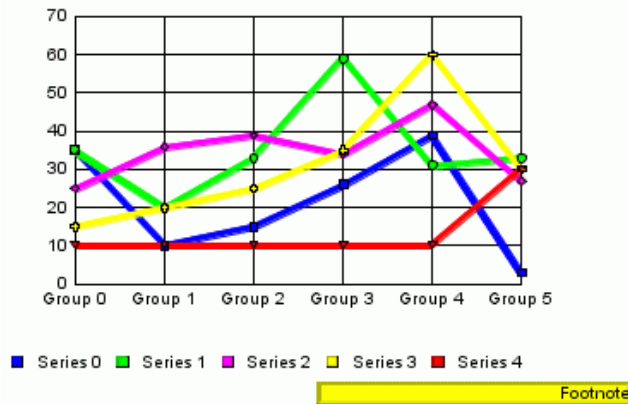
Return: *void*;

Example:

```

setDepthRadius(0);
setGraphType(41);
setDisplay(getFootnoteBox(), true);
setBorderType(getFootnoteBox(), 2);
setTopInset(getFootnoteBox(), 250);
setLeftInset(getFootnoteBox(), 250);
setBottomInset(getFootnoteBox(), 250);
setRightInset(getFootnoteBox(), 250);
setFillColor(getFootnoteBox(), new Color(255,255,0));

```



See [Beveled Borders](#) on page 178.

setLabelStagger()

This method enables/disables staggered labels on an axis.

Syntax:

```
void setLabelStagger(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1Label()).

newValue

Is true or false. If the value is:

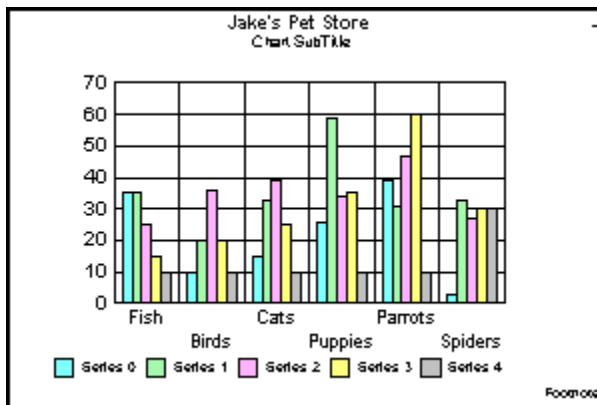
true, labels are staggered.

false, labels are not staggered.

Return: *void*;

Example:

```
setDepthRadius(0);
setGroupLabel(0, "Fish");
setGroupLabel(1, "Birds");
setGroupLabel(2, "Cats");
setGroupLabel(3, "Puppies");
setGroupLabel(4, "Parrots");
setGroupLabel(5, "Spiders"); setLabelStagger(getO1Label(), true);
setTitleString("Jake's Pet Store");
```



Note:

- ❑ If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).
- ❑ You may also use one of the axis-specific properties to set whether or not labels are staggered on a particular axis (for example, X1LabelStagger, Y1LabelStagger, Y2LabelStagger, and so on).

See [getLabelStagger\(\)](#) on page 764, [O1LabelStagger](#) on page 439, [X1LabelStagger](#) on page 588, [Y1LabelStagger](#) on page 628, [Y2LabelStagger](#) on page 664.

setLegendPosition()

You can select the position of the legend on your graph using setLegendPosition.

The code has been written so that additional presets can be added later if desired (for example place the legend inside the chart in the upper right corner). These positions are independent of Place Default Elements (PDE).

PDE should fix positioning issues. The user can define the legend rectangle by calling `setLegendAutomatic(false)` which instructs the legend engine to leave it alone. Similarly, calling `setLegendPosition(-1)` tells the legend engine to leave the legend's position alone.

Syntax:

`setLegendPosition (value);`

where:

value

Is one of the following:

- ☐ FREE FLOAT = -1;
- ☐ AUTO = 0;
- ☐ BOTTOM = 1;
- ☐ RIGHT = 2;
- ☐ LEFT = 3;
- ☐ TOP = 4;
- ☐ RIGHT TOP = 5;
- ☐ RIGHT BOTTOM = 6;
- ☐ LEFT TOP = 7;
- ☐ LEFT BOTTOM = 8;
- ☐ BOTTOM LEFT = 9;
- ☐ BOTTOM RIGHT = 10;
- ☐ TOP LEFT = 11;
- ☐ TOP RIGHT = 12;

See [Setting the Legend Position](#) on page 103 for examples of all the legend positions.

setLegendRect()

This method sets the size and location of the legend rectangle. This rectangle is the area in the graph where legend markers and labels are drawn.

Syntax:

```
void setLegendRect(Rectangle newValue);
```

```
void setLegendRect(Rectangle newValue,boolean bSetRecalc);
```

where:

newValue

Is the location/size of the legend area.

bSetRecalc

Is true or false. If the value is:

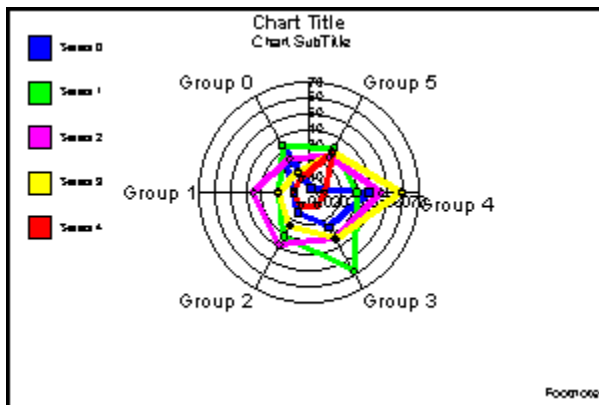
true, the entire graph is recalculated based on the new legend rectangle.

false, the graph is not recalculated.

Return: void;

Example:

```
setGraphType(69); setLegendRect(new Rectangle(-15340,-3478,4948,18550));
```



See [getLegendRect\(\)](#) on page 765, [LegendAutomatic](#) on page 397, [LegendDisplay](#) on page 398, [LegendMarkerPosition](#) on page 400, [LegendReverse](#) on page 404.

setLevelBorderColor()

When a Time Scale Axis is defined, this method applies a border color to a specified level.

Syntax:

```
void setLevelBorderColor(int level, Color color)
void setLevelBorderColor(IdentObj id, Color color)
```

where:

level

Is an integer between 0 and 2.

id

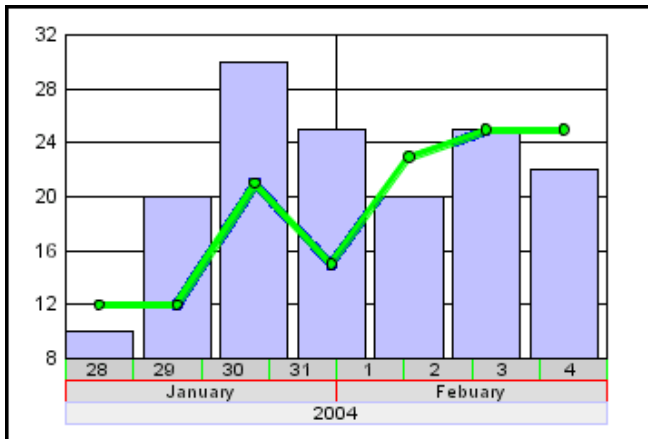
Is the object ID of a time scale axis level (that is, getTimeScaleLevel(n);)

color

Is the border color to be assigned to the specified level (for example, new Color (255,0,0)).

Return: void;

```
setLevelBorderColor(getTimeScaleLevel(1), new Color (255,0,0));
setLevelBorderColor(getTimeScaleLevel(0), new Color (0,255,0));
setLevelBorderColor(getTimeScaleLevel(2), new Color (0,0,255));
```

**Note:**

- ☐ A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.
- ☐ See UseTimeScaleAxis for the data that was used to create the example graph.

See [Time Scale Axis Graphs](#) on page 221.

setLevelDateFormat()

When a Time Scale Axis is defined and setLevelTextFormat() assigns FORMAT_RAW (3) to a level, this method defines a date format string to be used on a specified level.

Syntax:

```
void setLevelDateFormat(int level, String format)
```

```
void setLevelDateFormat(IdentObj id, String format)
```

where:

level

Is an integer between 0 and 3.

id

Is the object ID of a time scale axis level (that is, getTimeScaleLevel(n)).

format

Is a string of standard Java SimpleDateFormat codes which is described in the following table.

Format	Description
G =	Era Designation (for example, AD)
y =	Year (for example, 2004, 04)
M =	Month in year (e.g, March, Mar., 03)
w =	Week in year (for example, 1...52)
W =	Week in month (for example, 2)
D =	Day in year (for example, 1...365)
d =	Day in month (for example, 10)
F =	Day of week in month (for example, 2)
E =	Day in week (for example, Tuesday, Tue)
a =	AM/PM Marker (for example, PM)
H =	Zero-Based Hour in day (for example, 0...23)

Format	Description
k =	Hour in day (for example, 1...24)
K =	Zero-Based Hour in AM/PM (for example, 0...11)
h =	Hour in AM/PM (for example, 1...11)
m =	Minute in hour (for example, 30)
s =	Second in minute (for example, 55)
S =	Millisecond (for example, 978)
z =	General Time Zone (for example, Pacific Standard Time, PST, GMT-08:00)
Z =	RFC 822 Time Zone (for example, -0800)

Return: *void*;

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [getLevelDateFormat\(\)](#) on page 766 and [Time Scale Axis Graphs](#) on page 221.

setLevelFillColor()

When a Time Scale Axis is defined, this method assigns a fill color to a specified level.

Syntax:

`void setLevelFillColor(IdentObj id, Color color)`

`void setLevelFillColor(int level, Color color)`

where:

id

Is the object ID of a time scale axis level (that is, `getTimeScaleLevel(n)`;

level

Is an integer between 0 and 2.

color

Is the fill color to be assigned to the specified level (for example, `new Color (255,0,0)`).

Return: *void*;

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [getLevelFillColor\(\)](#) on page 767 and [Time Scale Axis Graphs](#) on page 221.

setLevelHeight()

When a Time Scale Axis is defined, this method assign a height to a specified level.

Syntax:

```
void setLevelHeight(int level, int height)
void setLevelHeight(IdentObj id, int height)
```

where:

level

Is an integer between 0 and 2.

id

Is the object ID of a time scale axis level (that is, `getTimeScaleLevel(n)`).

height

Is the level height in virtual coordinates.

Return: *void*;

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [getLevelHeight\(\)](#) on page 767 and [Time Scale Axis Graphs](#) on page 221.

setLevelInterval()

When a Time Scale Axis is defined, this method assigns an interval to a specified level. The level intervals must be set in decreasing order. The level zero interval must be less than level one, and level one must be less than level two.

Syntax:

```
void setLevelInterval(int level, int interval)
void setLevelInterval(IdentObj id, int interval)
```

where:

level

Is an integer between 0 and 2.

id

Is the object ID of a time scale axis level (that is, `getTimeScaleLevel(n)`).

interval

Is an integer between 0 and 8.

TIME_INTERVAL_UNDEFINED = 0

TIME_INTERVAL_SECONDS = 1

TIME_INTERVAL_MINUTES = 2

TIME_INTERVAL_HOURS = 3

TIME_INTERVAL_DAYS = 4

TIME_INTERVAL_WEEKS = 5

TIME_INTERVAL_MONTHS = 6

TIME_INTERVAL_QUARTERS = 7

TIME_INTERVAL_YEARS = 8

Return: *void*;

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [getLevelInterval\(\)](#) on page 767 and [Time Scale Axis Graphs](#) on page 221.

setLevelTextColor()

When a Time Scale Axis is defined, this method assigns a color to the text objects at a specified level.

Syntax:

```
void setLevelTextColor(int level, Color color)
```

```
void setLevelTextColor(IdentObj id, Color color)
```

where:

level

Is an integer between 0 and 2.

id

Is the object ID of a time scale axis level (that is, `getTimeScaleLevel(n)`).

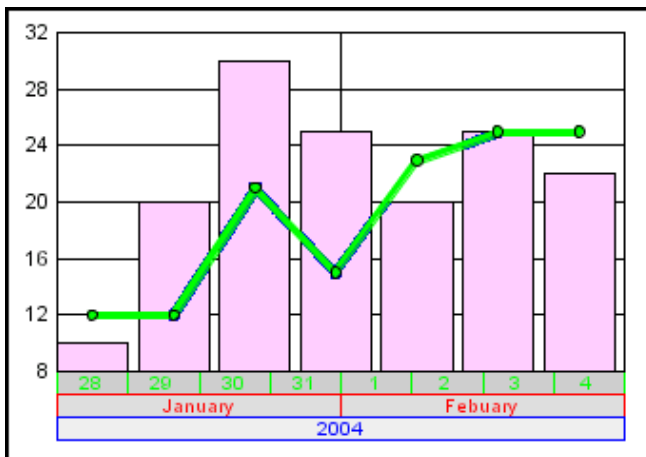
color

Is the text color to be assigned to the specified level (for example, new Color (255,0,0)).

Return: void;

Example:

```
setLevelBorderColor(getTimeScaleLevel(1), new Color (255,0,0));
setLevelBorderColor(getTimeScaleLevel(0), new Color(0,255,0));
setLevelBorderColor(getTimeScaleLevel(2), new Color(0,0,255));
setLevelTextColor(getTimeScaleLevel(1),new Color (255,0,0));
setLevelTextColor(getTimeScaleLevel(0),new Color(0,255,0));
setLevelTextColor(getTimeScaleLevel(2),new Color(0,0,255));
```

**Note:**

- ☐ A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.
- ☐ See UseTimeScaleAxis for the data that was used to create the example graph.

See [Time Scale Axis Graphs](#) on page 221.

setLevelTextFormat()

This method defines the format of text that is drawn to a specified level on a time scale axis.

Syntax:

```
void setLevelTextFormat(int level, int format)
void setLevelTextFormat(IdentObj id, int format)
```

where:

level

Is an integer between 0 and 2.

id

Is the object ID of a time scale axis level (that is, `getTimeScaleLevel(n)`).

format

Is an integer between 0 and 2.

`IntervalScale.FORMAT_SHORT = 0`
`IntervalScale.FORMAT_MEDIUM = 1`
`IntervalScale.FORMAT_LONG = 2`
`IntervalScale.FORMAT_RAW = 3`

The format in which text is drawn depends on the level interval set by `setLevelInterval()`. The following table shows how each text format is represented for each level:

	FORMAT_SHORT	FORMAT_MEDIUM	FORMAT_LONG
TIME_INTERVAL_SECONDS	1	Sec 1	Second 1
TIME_INTERVAL_MINUTES	1	Min 1	Minute 1
TIME_INTERVAL_HOURS	1	Hour 1	Hour 1
TIME_INTERVAL_DAYS	1	Day 1	Day 1
TIME_INTERVAL_WEEKS	1	Week 1	Week 1
TIME_INTERVAL_MONTHS	1	Mon 1	Month 1
TIME_INTERVAL_QUARTERS	1	Q 1	Quarter 1
TIME_INTERVAL_YEARS	1	01	2001

If `FORMAT_RAW` (`setLevelTextFormat(getTimeScaleLevel(1), 3)`) is used, you can specifically define how date and time text is formatted using `setLevelDateFormat()` and the standard Java `SimpleDateFormat` codes.

Return: `void`;

Note: A time scale axis can only be defined in clustered bar, stacked bar, line, and scatter graphs.

See [getLevelTextFormat\(\)](#) on page 768, [setLevelDateFormat\(\)](#) on page 923, and [Time Scale Axis Graphs](#) on page 221.

setLineBasicStrokeType()

This method defines which `BasicStroke` to create using Java2D `BasicStroke` constructors.

Syntax:

```
void setLineBasicStrokeType(IdentObj id, int newValue)
```

where:

id

Is the object ID of a line object returned by a getObject() method (that is, getY1MajorGrid()).

newValue

Can be one of the following values:

0 = Do not use Java2D. Draw line with regular JDK 1.1.8 code.

1 = Use Java2D BasicStroke() which constructs a new BasicStroke with defaults for all attributes.

2 = Use Java2D BasicStroke(float width) which constructs a solid BasicStroke with the specified line width and default values for the cap and join styles.

3 = Use Java2D BasicStroke(float width, int cap, int join) which constructs a solid BasicStroke with the specified attributes.

4 = Use Java2D BasicStroke(float width, int cap, int join, float miterlimit) which constructs a solid BasicStroke with the specified attributes.

5 = Use Java2D BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase) which constructs a new BasicStroke with the specified attributes.

See [setLineCap\(\)](#) on page 929 and [setLineJoin\(\)](#) on page 931 for examples. Also see [getLineBasicStrokeType\(\)](#) on page 768, [setLineDash\(\)](#) on page 930, [setLineDashPhase\(\)](#) on page 931, [setLineMiterLimit\(\)](#) on page 932.

setLineCap()

When setLineBasicStrokeType() is set to a value greater than zero to use the Java2D BasicStroke object, this method assigns an end cap style to a specified line object.

Syntax:

```
void setLineCap(IdentObj id, int newValue)
```

where:

id

Is the object ID of a line object returned by a getObject() method (that is, getY1MajorGrid()).

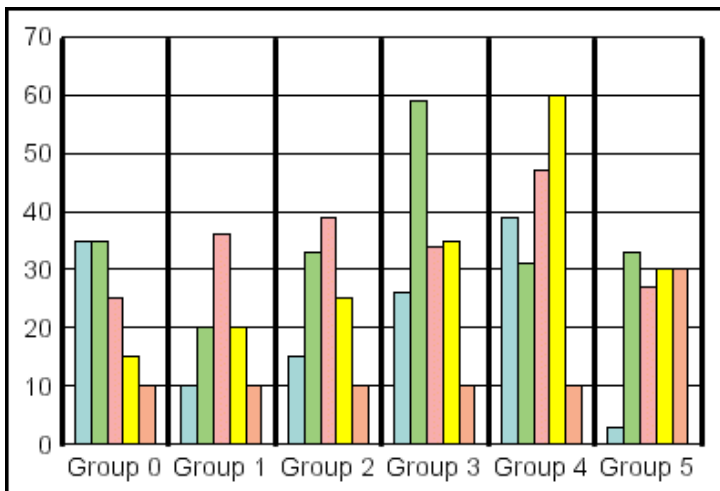
newValue

Defines the decoration to be applied to the ends of unclosed subpaths and dash segments of a specified line as defined by the Java 2D BasicStroke object (for example, CAP_BUTT, CAP_ROUND or CAP_SQUARE).

Return: *void*;

Example:

```
setLineBasicStrokeType(get01MajorGrid(), 5);
setLineWidth(get01MajorGrid(), 3); setLineCap(get01MajorGrid(),
BasicStroke.CAP_SQUARE);
```



Note: See the `java.awt.BasicStroke` constructors for constants that can be specified in *NewValue*.

See [getLineCap\(\)](#) on page 769, [setLineBasicStrokeType\(\)](#) on page 928.

setLineDash()

When `setLineBasicStrokeType()` is set to a value greater than zero to use the Java2D BasicStroke object, this method applies a dash pattern to a specified line object. Use `setLineDashPhase()` to specify the offset to start the dashing pattern.

Syntax:

```
void setLineDash(IdentObj id, Vector vectDashFloat)
```

where:

id

Is the object ID of a line object returned by a `getObject()` method (that is, `getY1MajorGrid()`).

Vector

Is a vector (`java.lang.Float`) of floating point numbers that defines the dash pattern.

Return: *void*;

See [getLineDash\(\)](#) on page 770, [setLineBasicStrokeType\(\)](#) on page 928, [setLineDashPhase\(\)](#) on page 931.

setLineDashPhase()

When `setLineBasicStrokeType()` is set to a value greater than zero to use the Java2D `BasicStroke` object and a line dashing pattern has been specified with `setLineDash()`, this method defines the offset to start the dashing pattern for a specified line object.

Syntax:

```
void setLineDashPhase(IdentObj id, float fNewValue)
```

where:

id

Is the object ID of a line object returned by a `getObject()` method (that is, `getY1MajorGrid()`).

fNewValue

Is the offset to start the dashing pattern.

Return: *void*;

See [getLineDashPhase\(\)](#) on page 770, [setLineBasicStrokeType\(\)](#) on page 928, [setLineDash\(\)](#) on page 930.

setLineJoin()

When `setLineBasicStrokeType()` is set to a value greater than zero to use the Java2D `BasicStroke` object, this method defines the decoration to be applied to the line object where line path segments meet.

Syntax:

```
void setLineJoin(IdentObj id, int newValue)
```

where:

id

Is the object ID of a line object returned by a `getObject()` method (that is, `getY1MajorGrid()`).

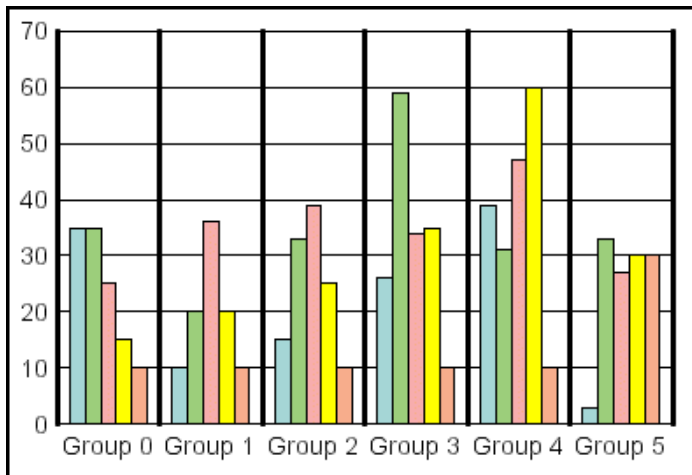
NewValue

Is the decoration applied where path segments meet (for example, `JOIN_ROUND`, `JOIN_BEVEL`, or `JOIN_MITER`).

Return: *void*;

Example:

```
setLineBasicStrokeType(getO1MajorGrid(), 3);
setLineWidth(getO1MajorGrid(), 3);
setLineCap(getO1MajorGrid(), BasicStroke.CAP_ROUND);
setLineJoin(getO1MajorGrid(), BasicStroke.JOIN_BEVEL);
```



Note: See the `java.awt.BasicStroke` constructors for constants that can be specified in *NewValue*.

See [getLineJoin\(\)](#) on page 771, [setLineBasicStrokeType\(\)](#) on page 928.

setLineMiterLimit()

When `setLineJoin()` applies `JOIN_MITER` to a line object, this method defines the limit to trim the miter join.

Syntax:

```
void setLineMiterLimit(IdentObj id, float fNewValue)
```

where:

id

Is the object ID of a line object returned by a getObject() method (that is, getY1MajorGrid()).

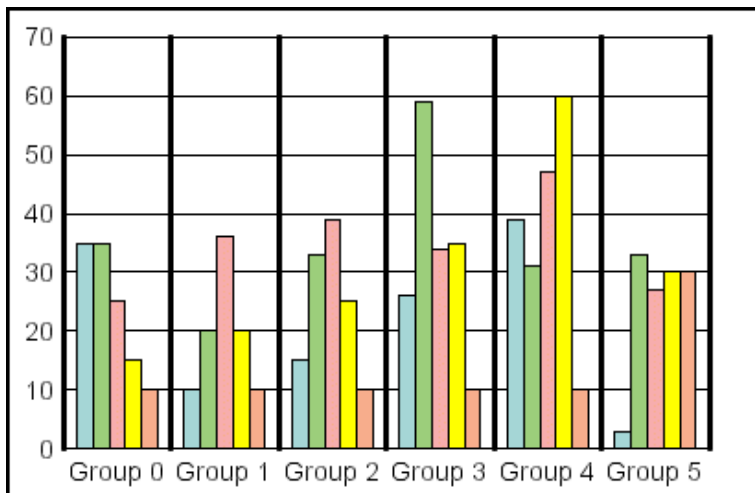
fNewValue

Is the limit to trim the miter join. The miterlimit must be greater than or equal to 1.0f.

Return: void;

Example:

```
setLineBasicStrokeType(getO1MajorGrid(), 4);
setLineBasicStrokeType(getO1MajorGrid(), 3);
setLineWidth(getO1MajorGrid(), 3);
setLineCap(getO1MajorGrid(), BasicStroke.CAP_ROUND);
setLineJoin(getO1MajorGrid(), BasicStroke.JOIN_MITER);
setLineMiterLimit(getO1MajorGrid(), 10.0f);
```



See [getLineMiterLimit\(\)](#) on page 771, [setLineJoin\(\)](#) on page 931.

setLineStyleLegendMarkers()

This method sets Line Style to the Legend Marker. When creating a line chart using the various marker shapes (square, circle, star, and so on), these marker shapes are now represented in corresponding legend markers. This is helpful in visually mapping the line segment to the legend entry.

Syntax:

```
setLineStyleLegendMarkers(value);
```

where:

value

Can be one of the following values:

0 = No Line Style Markers. This value is the default.

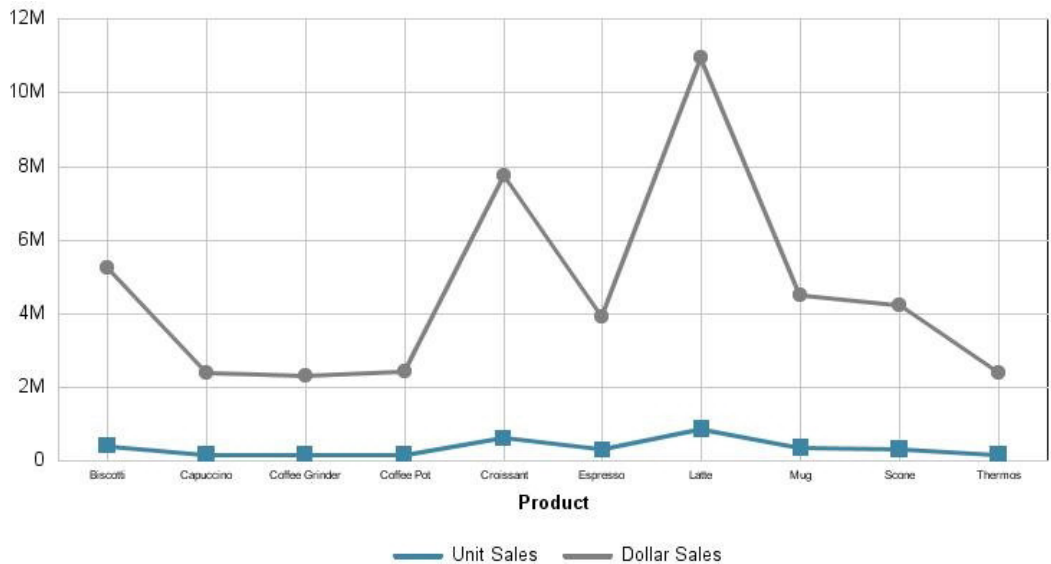
1 = Line Style Markers.

2 = Line Style and Markers.

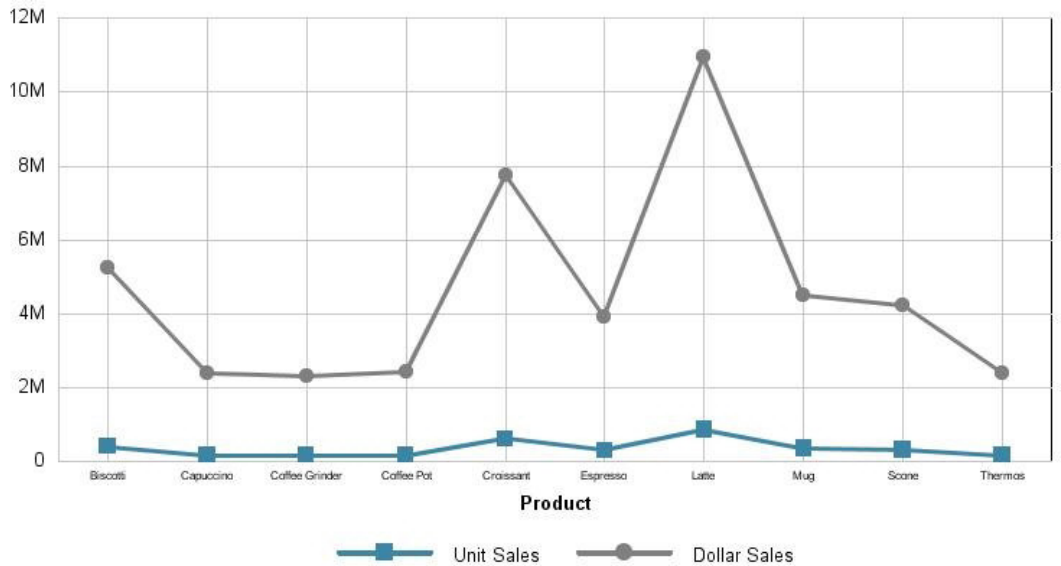
3 = Automatic Marker Style.

Examples:

```
setLineStyleLegendMarkers(1);  
setMarkerShape(getSeries(0),1);  
setMarkerShape(getSeries(1),2);
```



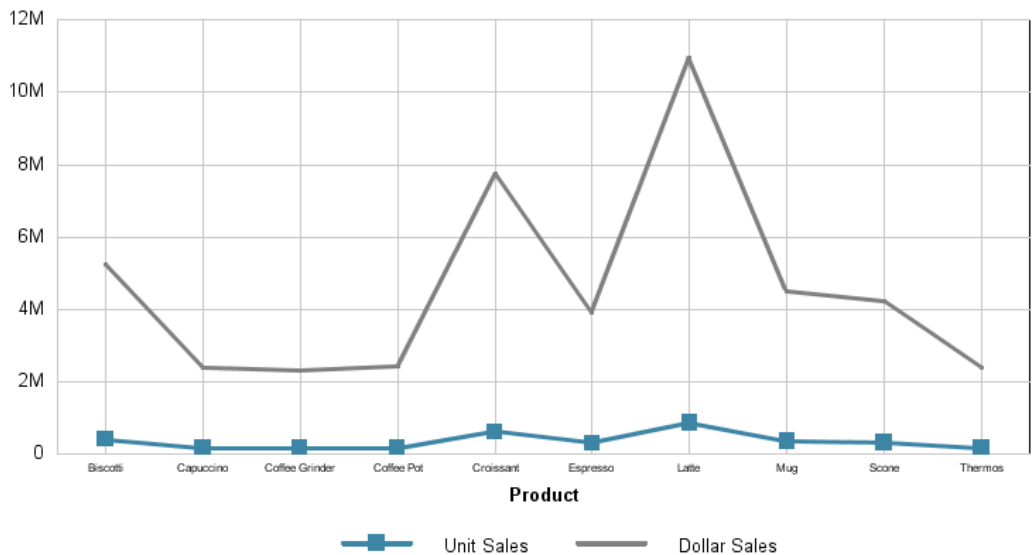
```
setLineStyleLegendMarkers(2);  
setMarkerShape(getSeries(0),1);  
setMarkerShape(getSeries(1),2);
```



```

setLineStyleLegendMarkers(3);
setNullLegendMarkerShapesAsSquares(false);
setMarkerShape(getSeries(0),1);
setMarkerShape(getSeries(1),0);

```



Note: `setNullLegendMarkerShapesAsSquares(false)`; must be used with this setting in order for it to work properly.

See [NullLegendMarkerShapesAsSquares](#) on page 416.

setLineWidth()

This method defines the thickness (in pixels) of a line object.

Syntax:

```
void setLineWidth(IdentObj id,int newValue);
void setLineWidth(int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries ()`).

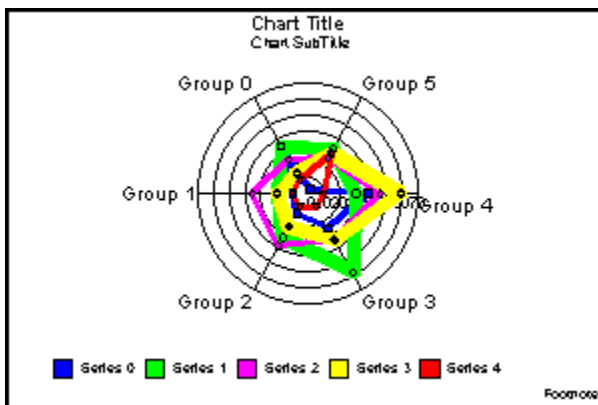
newValue

Is the line width in pixels.

Return: *void*;

Example:

```
setGraphType(67); setLineWidth(getSeries(3),10);
setLineWidth(getSeries(1),10);
```



Note: If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [getLineWidth\(\)](#) on page 771, [setLineWidthAllSeries\(\)](#) on page 938.

setLineWidthAllSeries()

This method assigns equal line widths for all series in a graph.

Syntax:

```
void setLineWidthAllSeries(int newValue)
```

where:

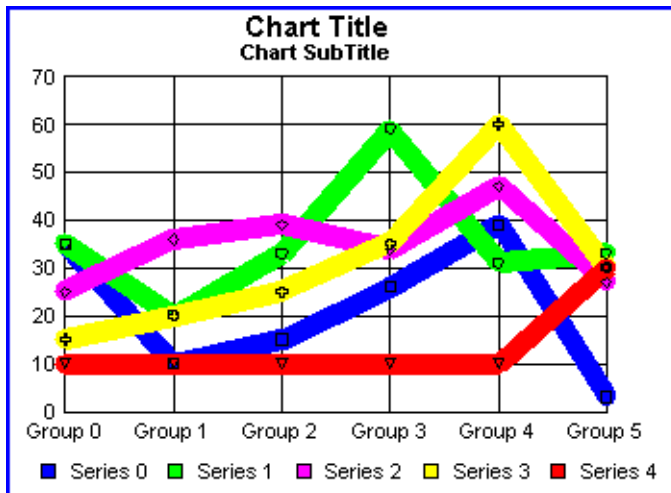
newValue

Is the line width in pixels.

Return: void;

Example:

```
setDepthAngle(0);  
setDepthRadius(0);  
setLineWidthAllSeries(10);  
setGraphType(41);
```



setLogScale()

This method selects logarithmic or linear scale on an axis.

Syntax:

```
void setLogScale(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getY1Axis()`).

newValue

Is true or false. If the value is:

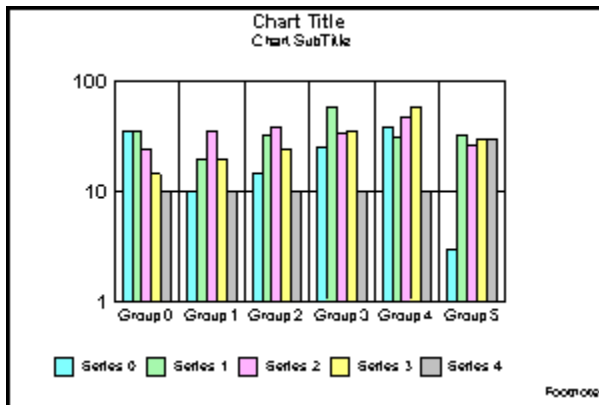
true, a logarithmic scale is used.

false, a linear scale is used.

Return: *void*;

Example:

```
setDepthRadius(0);
setY1MinorGridStep(0.5);
setY1MajorGridStep(1.0);
setY1MinorGridStepAuto(true);
setY1MajorGridStepAuto(true); setLogScale(getY1Axis(), true);
setY1ScaleMaxAuto(true);
setY1ScaleMinAuto(true);
setY1MustIncludeZero(false);
```



Note: You may also use one of the axis-specific properties to select a logarithmic or linear scale (for example, `X1LogScale`, `Y1LogScale`, `Y2LogScale`, and so on).

See [setLogScaleBase\(\)](#) on page 940, [X1LogScale](#) on page 589, [Y1LogScale](#) on page 629, [Y2LogScale](#) on page 666.

setLogScaleBase()

When logarithmic scaling is used on a numeric axis, this method sets the lowest/base value drawn on the axis.

Syntax:

```
void setLogScaleBase(IdentObj id, double newValue)
```

where:

newValue

Is a log scale base value.

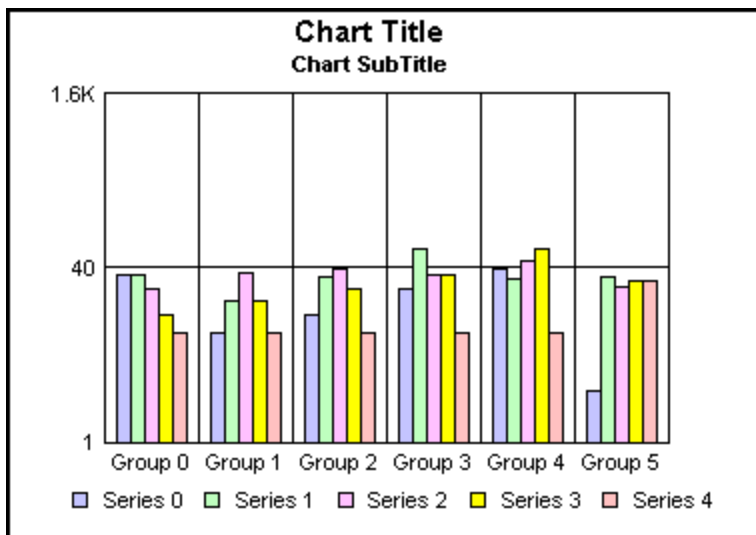
id

Is an object ID returned by a getObjectID() method (for example, getY1Axis()).

Return: void;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setY1LogScale(true); setLogScaleBase(getY1Axis(), 40);
```



See [getLogScale\(\)](#) on page 772, [getLogScaleBase\(\)](#) on page 773, [setLogScale\(\)](#) on page 938.

setMarkerShape()

This method sets the shape of a marker.

Syntax:

```
void setMarkerShape(int integerMarkerShape);
```

```
void setMarkerShape(IdentObj id,int integerMarkerShape);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries(1)).

integerMarkerShape

Is a value from 0...18, producing one of the following shapes:

1	2	3	4	5	6	7	8	9
								
10	11	12	13	14	15	16	17	18
								

Note: The value 0 is for no marker shape.

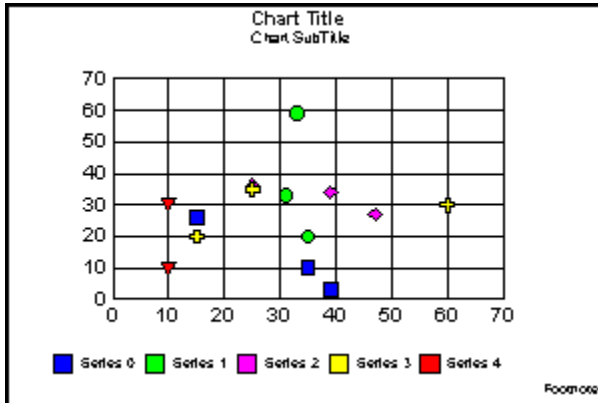
Return: *void*;

Example:

```

setDepthRadius(0);
setDepthAngle(0);
setGraphType(61);
setMarkerSizeDefault(60);  setMarkerShape(getSeries(0), 6);

```



Note: If an object ID (id) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [getMarkerShape\(\)](#) on page 774, [getMarkerSize\(\)](#) on page 776, [setMarkerSize\(\)](#) on page 942, [getMarkerTemplate\(\)](#) on page 777, [MarkerDisplay](#) on page 411, [MarkerSizeDefault](#) on page 412.

setMarkerSize()

This method sets the size of markers in a graph.

Syntax:

```

void setMarkerSize(int newValue);
void setMarkerSize(IdentObj id, int newValue);
void setMarkerSize(int seriesID,int newValue);
void setMarkerSize(int seriesID,int groupID,int newValue);

```

where:

seriesID

Is a series number.

groupID

Is a group number.

id

Is an object ID returned by a getObjectID() method (for example, getSeries(1)).

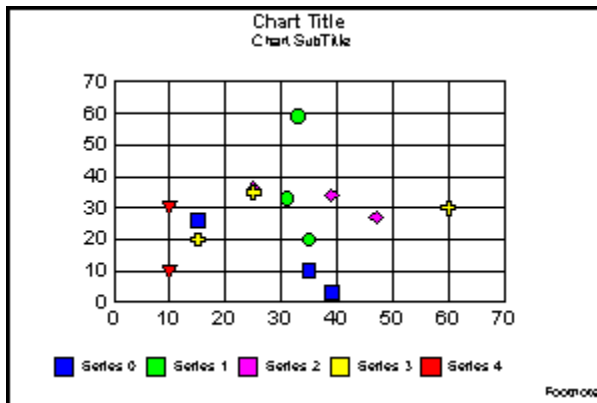
newValue

Is a value (0...100) that represents the size of a marker.

Return: void;

Example:

```
setGraphType(61); setMarkerSize(0,60);
setMarkerShape(0,6); setMarkerSize(1,60);
setMarkerShape(1,5); setMarkerSize(2,60);
setMarkerShape(2,4); setMarkerSize(3,60);
setMarkerShape(3,3); setMarkerSize(4,60);
setMarkerShape(4,2); setMarkerSize(5,60);
setMarkerShape(5,1);
```



Note: If an object ID (id), series number (seriesID), or group number (groupID) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [getMarkerSize\(\)](#) on page 776, [getMarkerShape\(\)](#) on page 774, [setMarkerShape\(\)](#) on page 940, [MarkerDisplay](#) on page 411, [MarkerSizeDefault](#) on page 412.

setMaxNumAnnotations()

The following method can be used to set the maximum number of annotations.

Syntax:

```
void setMaxNumAnnotations(int newValue);
```

where:

newValue

Is a value in the range 0...999.

For annotations in excess of the default (30), the `setTransparentFillColor()` and `setTransparentBorderColor()` methods should be used to set transparent fill and border colors for the associated annotation box as desired:

- ❑ `setTransparentFillColor(getAnnotationBox(nnn), true | false);`
- ❑ `setTransparentBorderColor(getAnnotationBox(nnn), true | false);`

setNoExceptionalRiser()

This method can be used to prohibit the use of exceptional risers in a graph.

Syntax:

```
void setNoExceptionalRiser();
```

Return: *void*;

See [setExceptionalRiser\(\)](#) on page 887.

setNonZeroBaseline()

This method enables/disables a non-zero baseline on a numeric axis in a graph. You must use `setNonZeroBaselineValue()` to define the location of the non-zero baseline. When this method enables (true) the non-zero baseline and a baseline value is defined, risers with values greater than the baseline value draw up from the baseline; risers with values less than the baseline value draw down from the baseline.

Syntax:

```
void setNonZeroBaseline(IdentObj id, boolean newValue)
```

where:

newValue

Is true or false. If the value is:

true, the baseline is non-zero.

false, the baseline is the default (0).

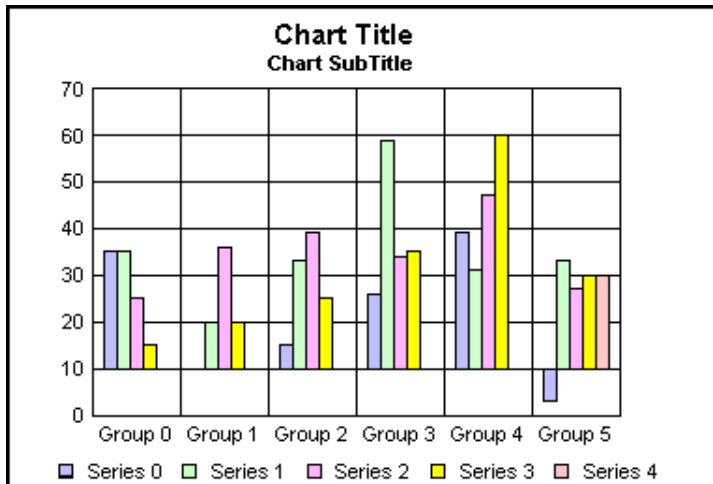
id

Is an object ID of a numeric axis returned by a `getObjectID()` method (for example, `getY1Axis()`).

Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0); setNonZeroBaseline(getY1Axis(),true);
setNonZeroBaselineValue(getY1Axis(),10);
```



See [getNonzeroBaselineValue\(\)](#) on page 782, [getNonZeroBaseline\(\)](#) on page 781, [setNonZeroBaselineValue\(\)](#) on page 945.

setNonZeroBaselineValue()

When `setNonZeroBaseline()` enables a non-zero baseline, this method defines the location of the non-zero baseline. When a non-zero baseline value is defined with this method, risers with values greater than the baseline value draw up from the baseline; risers with values less than the baseline value draw down from the baseline.

Syntax:

```
void setNonZeroBaselineValue(IdentObj id, double fValue)
```

where:

id

Is an object ID of a numeric axis returned by a `getObjectID()` method (for example, `getY1Axis()`).

fValue

Is a new non-zero value.

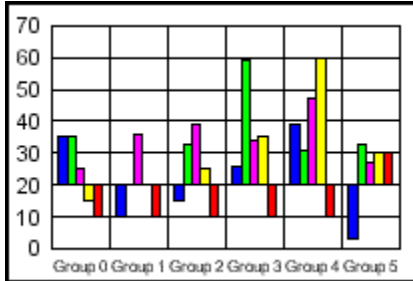
Return: *void*;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setNonZeroBaseline(getY1Axis(), true);
setNonZeroBaselineValue(getY1Axis(), 20);

```



See [getNonZeroBaseline\(\)](#) on page 781, [setNonZeroBaseline\(\)](#) on page 944, [getNonzeroBaselineValue\(\)](#) on page 782.

setNoSelection()

This method deselects all objects in the selection list.

Syntax:

```
void setNoSelection();
```

Return: *void*;

setPieBreakoutMode()

This method enables the pie breakout mode. Pie slices that are equal to or greater than a specific user assigned percentage are grouped together in one slice. The group is then displayed in a stacked bar associated with that slice. This mode is useful when the focus is on pie slices with larger values.

Syntax:

```
void setPieBreakoutMode(boolean);
```

where:

[true](#)

Enables the pie breakout mode.

[false](#)

Disables the pie breakout mode. This value is the default.

Return: *void*;

See [setPieBreakoutPercentage\(\)](#) on page 947.

setPieBreakoutPercentage()

This method is used to allow a certain percentage of a pie to be grouped and shown in detail using a bar.

Syntax:

```
void setPieBreakoutPercentage(n);
```

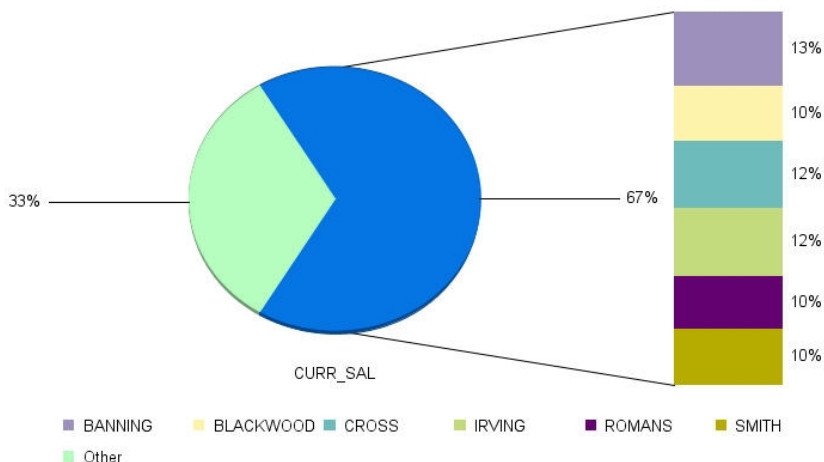
where:

n is a value between 0 to 100 that sets the percentage that is used as the threshold. The default value is 5 (5%).

Slices that represent a percentage that is below the set threshold get aggregated into one slice referred to as OTHER. Slices that represent a percentage that is equal to or above the set threshold get aggregated into a second slice, referred to as a breakout slice, and are also shown in detail in the accompanying legend.

Example:

```
setPieBreakoutMode(true); setPieBreakoutPercentage(9);  
setFillColor(getSeries(-2),new Color(0,119,227));  
setFillColor(getSeries(-1),new Color(179,255,191));
```



See [setPieBreakoutMode\(\)](#) on page 946 and [setFillColor\(\)](#) on page 890.

setPieSliceDelete()

This method can be used to delete a slice from a pie graph.

Syntax:

```
void setPieSliceDelete(IdentObj id, boolean newValue);  
void setPieSliceDelete(int seriesID, boolean newValue);  
void setPieSliceDelete(int seriesID,  
    int groupID,boolean newValue);
```

where:

id

Is object ID returned by a getObjectID() method (for example, getSlice ()).

seriesID

Is a series number.

groupID

Is a group number.

newValue

Is true or false. If the value is:

true, the slice is deleted.

false, the slice is not deleted.

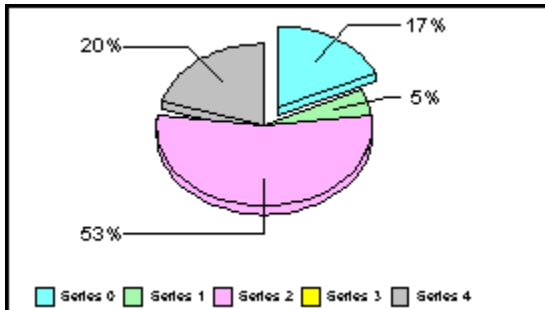
Return: *void*;

Example:

```

setPieSliceDelete(getSeries(3),true);
setPieSliceDelete(getSeriesGroup(3,0),true);
setPieSliceDelete(getSeries(0),false);
setPieSliceDelete(getSeriesGroup(0,0),false);
setPieSliceDetach(getSeries(0),100);
setPieSliceDetach(getSeriesGroup(0,0),100);

```



See [getPieSliceDelete\(\)](#) on page 789, [getPieSliceDetach\(\)](#) on page 790, [setPieSliceDetach\(\)](#) on page 949, [restoreAllSlices\(\)](#) on page 838.

setPieSliceDetach()

This method detaches a slice from a pie graph by a specified distance.

Syntax:

```

void setPieSliceDetach(IdentObj id, boolean newValue);
void setPieSliceDetach(int seriesID, boolean newValue);
void setPieSliceDetach(int seriesID, int groupID, int newValue);

```

where:

seriesID

Is a series number.

groupID

Is a group number.

newValue

Is a distance (0...100) to detach slice.

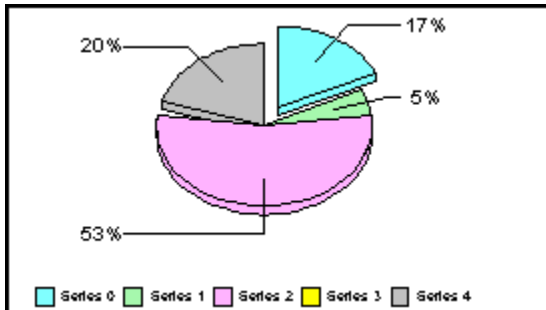
Return: void;

Example:

```

setPieSliceDelete(getSeries(3), true);
setPieSliceDelete(getSeriesGroup(3, 0), true);
setPieSliceDelete(getSeries(0), false);
setPieSliceDelete(getSeriesGroup(0, 0), false);
setPieSliceDetach(getSeries(0), 100);
setPieSliceDetach(getSeriesGroup(0, 0), 100);

```

**setPlaceAlign()**

When the automatic graph layout manager is enabled with `setPlace(true)`, this method defines whether or not horizontal alignment of a specified graph object is allowed. The default value is 1 (allow horizontal alignment of all graph objects).

Syntax:

```
void setPlaceAlign(IdentObj id, int newValue)
```

where:

id

Is an object ID of a text object returned by a `getObjectID()` method (for example, `getTitle()`, `getO1Label()`, etc.).

newValue

Is one of the following values:

0, do not allow horizontal alignment of the specified graph object.

1, allow horizontal alignment of the specified graph object.

Return: *void*;

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that realignment is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceAlign\(\)](#) on page 791.

setPlaceMargin()

When the automatic graph layout manager is enabled with `setPlace(true)`, this function will add a margin around the entire graph as defined by the rectangle parameter. The default is new Rectangle (-15500, -15500, 24000, 31000).

Syntax:

```
void setPlaceMargin(Rectangle newValue)
```

where:

newValue

Is 4 values specifying x, y, width, & height of rectangle in virtual coordinates (e.g. new Rectangle (-15500, -15500, 24000, 31000)).

Return: *void*;

Note: All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.

See [Place](#) on page 496, [getPlaceMargin\(\)](#) on page 791.

setPlaceMaxNumWordWrapLines()

When the automatic graph layout manager is enabled with `setPlace(true)` and word wrapping is enabled with `setPlaceWordWrap(1)`, this method defines the maximum number of lines that can be used when a specified label object is word wrapped.

Syntax:

```
void setPlaceMaxNumWordWrapLines(IdentObj id, int newValue)
```

where:

id

Is the object ID of a text object returned by a getObjectID() method like get O1Label().

newValue

Is the maximum number of lines to use for word wrapping.

Return: *void*;

Note:

- ❑ All getPlace...() and setPlace...() methods are only valid when the automatic graph layout manager has been enabled with setPlace(true).
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that word wrapping is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceMaxNumWordWrapLines\(\)](#) on page 792, [setPlaceWordWrap\(\)](#) on page 959.

setPlaceMinFontSize()

When the automatic graph layout manager is enabled with setPlace(true) and resizing of an object is enabled with setPlaceResize(1), this method defines the minimum font size (in points) that can be used to resize a specified text object.

Syntax:

```
void setPlaceMinFontSize(IdentObj id, int newValue)
```

where:

id

Is an object ID returned by a getObjectID() method for a text object (for example, getFootnote(), getO1Label(), etc.).

newValue

Is the minimum font size in points (4...96). The default value is 4.

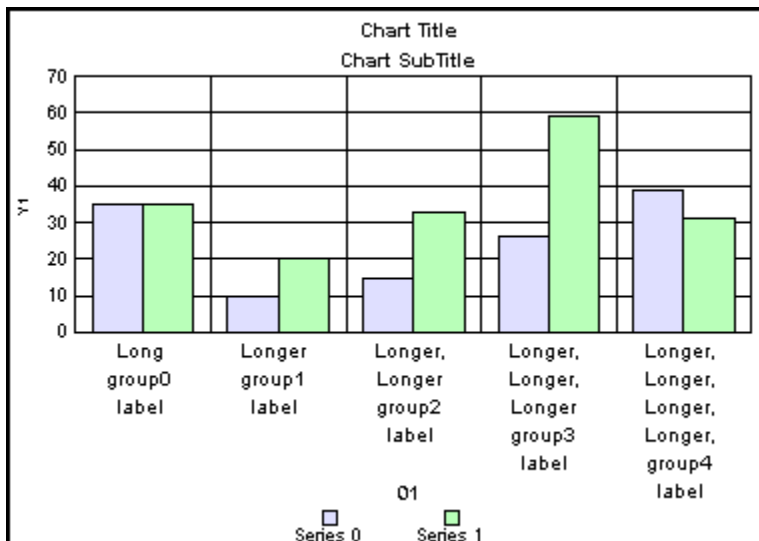
Return: *void*;

Example:

```

restoreDefaults();
setGroupLabel(getGroup(0), "Long group0 label");
setGroupLabel(getGroup(1), "Longer group1 label");
setGroupLabel(getGroup(2), "Longer, Longer group2 label");
setGroupLabel(getGroup(3), "Longer, Longer,
    Longer group3 label");
setGroupLabel(getGroup(4), "Longer, Longer,
    Longer, Longer, group4 label");
setDisplay(getY1Title(),true);
setDisplay(getO1Title(),true);
setTextString(getY1Title(),"Y1");
setTextString(getO1Title(),"O1");
setPlaceSkip(getO1Label(),0);
setFontSizeAbsolute(getO1Label(), true);
setFontSize(getO1Label(),10);
setPlaceMinFontSize(getO1Label(), 10);
setPlaceRect(getPlaceVertO1LabelsTitle(),
    new Rectangle(-14000, -15500, 29000, 24000));
setTextJustVert(getO1Label(),0);
setDataRange(0,0,1,4);
setPlace(true);
setDepthRadius(0);
setFootnoteDisplay(false);

```



Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that font resizing is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceMinFontSize\(\)](#) on page 792.

setPlaceRect()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method sets up an initial rectangle to be used for initial placement of objects in a graph. In the current implementation, only object IDs return by `getPlaceVertO1LabelsTitle()` or `getPlaceHorzO1LabelsTitle()` can be used.

Syntax:

```
void setPlaceRect(IdentObj id, Rectangle newValue)
```

where:

id

Is an object ID returned by `getPlaceVertO1LabelsTitle()` or `getPlaceHorzO1LabelsTitle()`.

newVa lue

Are the values that specify the X-axis, Y-axis, width, and height of the rectangle in virtual coordinates. 4 values specifying x, y, width, & height of rectangle in virtual coordinates (for example, new Rectangle (-15500, -15500, 24000, 31000)).

Return: *void;*

Note: All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.

See [setPlaceMinFontSize\(\)](#) on page 952 for an example. Also see [Place](#) on page 496, [getPlaceRect\(\)](#) on page 793.

setPlaceResize()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method defines whether or not an object can be moved or resized. The default value is 1 (allow resizing of all graph objects).

Syntax:

```
void setPlaceResize(IdentObj id, int newValue)
```

where:

id

Is the object ID of a graph object returned by a getObjectID() method (for example, getFrame()).

newValue

Is one of the following values:

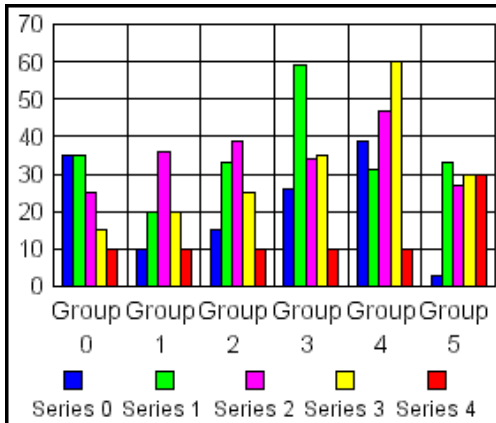
0 does not allow the specified object to be moved or resized.

1 allows the specified object to be moved or resized.

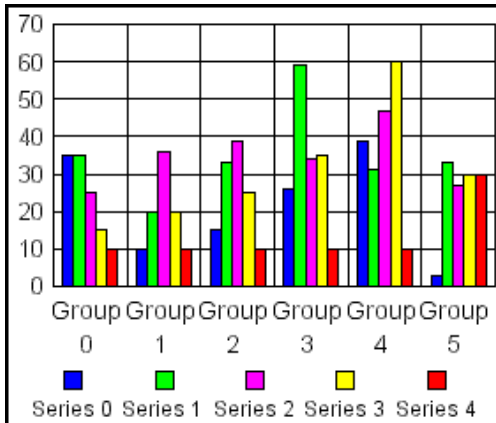
Return: *void*;

Example:

```
setDepthRadius(0);  setPlaceResize(getFrame(),1);
setPlace(true);
```



```
setDepthRadius(0);  setPlaceResize(getFrame(),0);
setPlace(true);
```

**Note:**

- ☐ All getPlace...() and setPlace...() methods are only valid when the automatic graph layout manager has been enabled with setPlace(true).
- ☐ The value assigned to the specified object will only be used if the automatic graph layout manager determines that resizing is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceResize\(\)](#) on page 793.

setPlaceSkip()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method disables (0) / enables (1) label skipping on the primary ordinal axis. The default value is 1 (allow skipping for all axis labels).

Syntax:

```
void setPlaceSkip(IdentObj id, int newValue)
```

where:

id

Is the object ID of primary ordinal axis labels (for example, `getO1Label()`).

newValue

Is one of the following values:

0 disables label skipping.

1 enables label skipping

Return: *void*;

Note:

- ❑ All `getPlace...()` and `setPlace...()` methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that label skipping is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceSkip\(\)](#) on page 794.

setPlaceTruncateCount()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method can be used to specify the maximum number of characters allowed for a text object. When a text object is longer than the specified number of characters ellipses (...) will be appended to the truncated text to indicate the text has been truncated. The ellipses count for 3 characters. In the example below, group labels "Group 0", "Group 1", etc., are truncated to "Gro..." because the maximum label character count is 6.

Syntax:

```
void setPlaceTruncateCount(IdentObj id, int newValue)
```

where:

id

Is the object ID of a text object returned by a getObjectID() method (for example, getO1Label()).

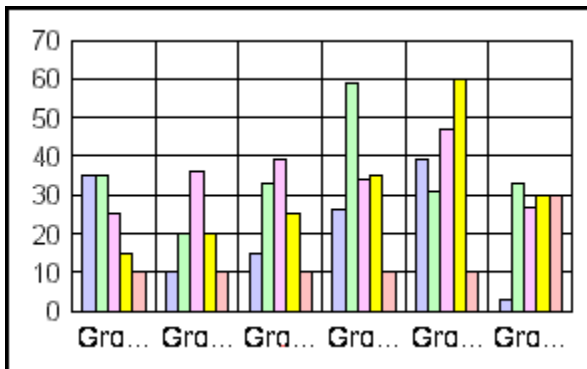
newValue

Is the maximum number of characters allowed.

Return: void;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFillColor(getO1Label(), new Color(255,0,0));
setPlace(true); setPlaceTruncateCount(getO1Label(), 6);
```

**Note:**

- ❑ All getPlace...() and setPlace...() methods are only valid when the automatic graph layout manager has been enabled with setPlace(true).
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that truncation is needed to produce a better graph layout.
- ❑ Data is truncated permanently in the Perspective for Java data manager when this method sets a truncate count.

See [Place](#) on page 496, [getPlaceTruncateCount\(\)](#) on page 794.

setPlaceWordWrap()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method defines whether or not long text objects can be word wrapped. The default value is 1 (allow word wrap for all text objects).

Syntax:

```
void setPlaceWordWrap(IdentObj id, int newValue)
```

where:

id

Is an object ID of a text object returned by a `getObjectID()` method (for example, `getO1Label()`).

newValue

Is one of the following values:

0 disables word wrap.

1 enables word wrap.

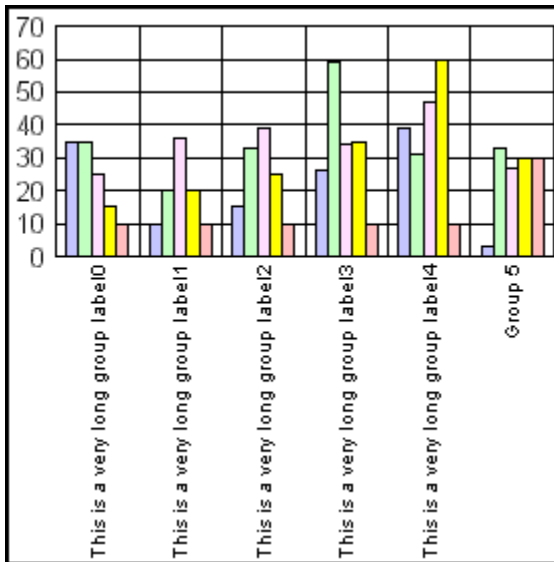
Return: *void*;

Example:

```

setDepthRadius(0);
setGroupLabel(getGroup(0), "This is a very long group label0");
setGroupLabel(getGroup(1), "This is a very long group label1");
setGroupLabel(getGroup(2), "This is a very long group label2");
setGroupLabel(getGroup(3), "This is a very long group label3");
setGroupLabel(getGroup(4), "This is a very long group label4");
setPlaceWordWrap(get01Label(), 0);
setPlace(true);

```

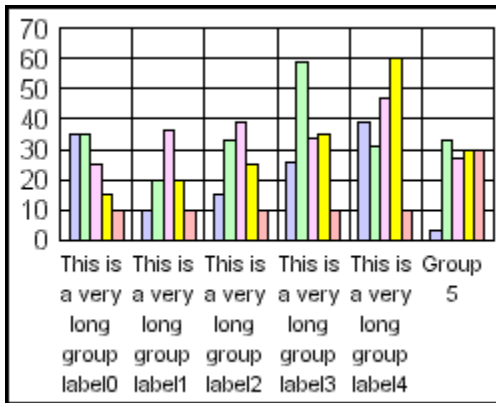


```

setDepthRadius(0);
setGroupLabel(getGroup(0), "This is a very long group label0");
setGroupLabel(getGroup(1), "This is a very long group label1");
setGroupLabel(getGroup(2), "This is a very long group label2");
setGroupLabel(getGroup(3), "This is a very long group label3");
setGroupLabel(getGroup(4), "This is a very long group label4");

```

```
setPlaceWordWrap(get01Label(), 1);
setPlace(true);
```



Note:

- ❑ All `getPlace...`() and `setPlace...`() methods are only valid when the automatic graph layout manager has been enabled with `setPlace(true)`.
- ❑ The value assigned to the specified object will only be used if the automatic graph layout manager determines that word wrapping is needed to produce a better graph layout.

See [Place](#) on page 496, [getPlaceWordWrap\(\)](#) on page 795.

setQuadrantLineValueX()

This method sets the scale value at which to place the vertical quadrant line number `nIndex`.

Syntax:

```
void setQuadrantLineValueX(int nIndex,
double newValue);
```

where:

nIndex

Is the quadrant line number (0...number of quadrant lines in the graph).

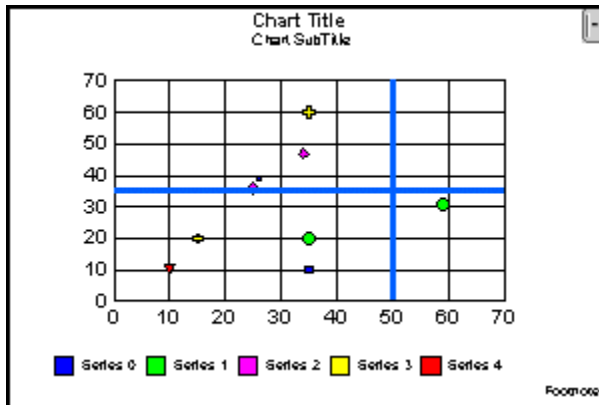
newValue

Is the scale value. `NO_LINEVALUE` enables automatic scaling.

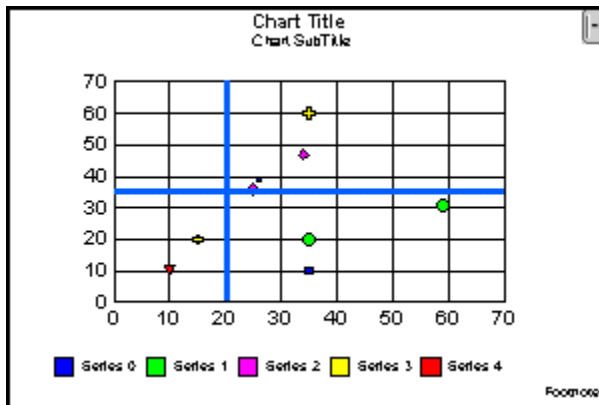
Return: *void*;

Example:

```
setGraphType(89);  setQuadrantLineValueX(0,50);
```



```
setQuadrantLineValueX(0,20);
```

**setQuadrantLineValueY()**

This method sets the scale value at which to place the horizontal quadrant line number *nIndex*.

Syntax:

```
void setQuadrantLineValueY(int nIndex, double newValue);
```

where:

nIndex

Is the quadrant line number (0...number of quadrant lines in the graph).

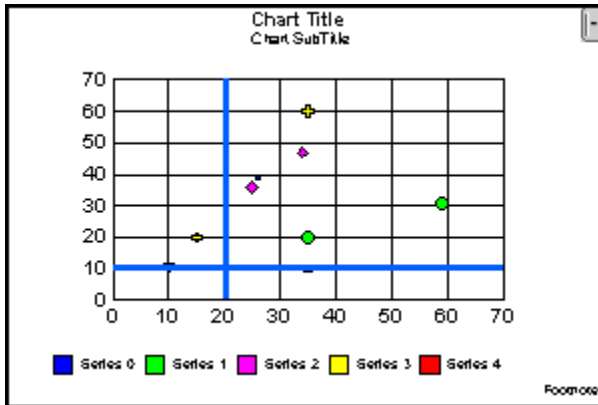
newValue

Is the scale value. NO_LINEVALUE enables automatic scaling.

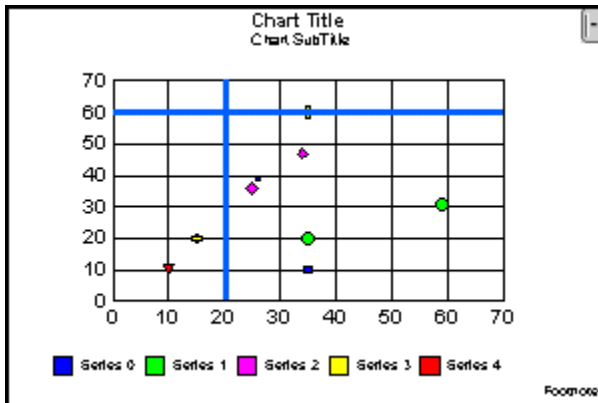
Return: *void*;

Example:

```
setGraphType(89);  setQuadrantLineValueY(10);
```



```
setQuadrantLineValueY(60);
```



setRect()

This method sets the bounding rectangle for a user-movable object in virtual coordinates (x, y, width, height). See [Virtual Coordinate System](#) on page 47.

Syntax:

```
void setRect(IdentObj id, Rectangle newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getFrame()`).

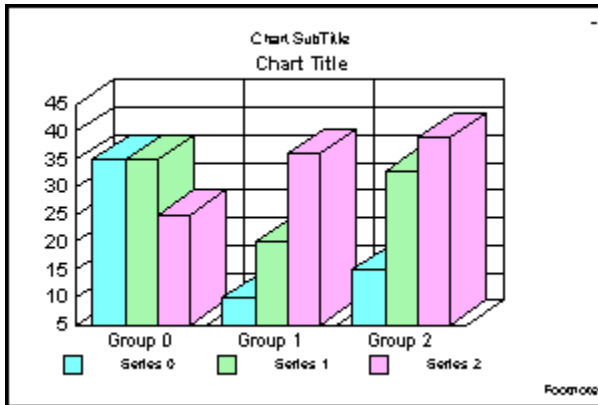
newValue

Is the bounding rectangle for a user-movable object in virtual coordinates (x, y, width, height).

Return: *void*;

Example:

```
setRect(getTitle(),
        new Rectangle(-15175,10550,29850,1800));
```



See [getRect\(\)](#) on page 797, [getFrameRect\(\)](#) on page 754, [getLegendRect\(\)](#) on page 765, [setLegendRect\(\)](#) on page 921.

setReferenceLine()

This method defines a reference line to be drawn in the graph.

Syntax:

```
void setReferenceLine(IdentObj axisObj, int nIndex, double nPosition)
```

where:

axisObj

Is an object ID of an axis object returned by a `getObjectID()` method (for example, `getY1Axis()`).

nIndex

Is the reference line index number (0...2).

nPosition

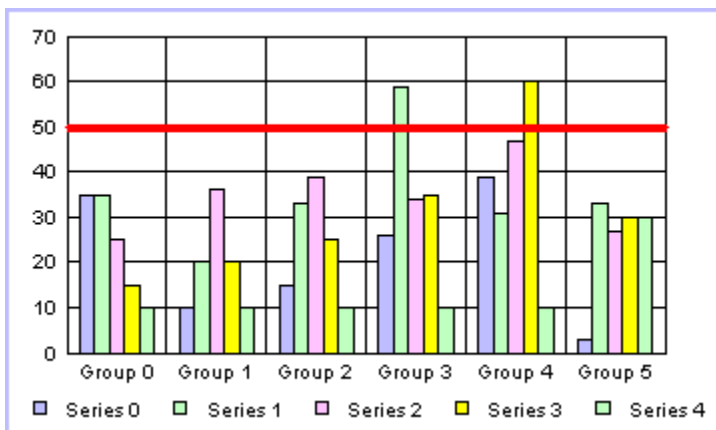
Is the reference line position.

Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0);

setReferenceLine(getY1Axis(),0,50);
setLineWidth(getReferenceLine(getY1Axis(),0),3);
setFillColor(getReferenceLine(getY1Axis(),0), new Color(255,0,0));
```



See [getReferenceLineValue\(\)](#) on page 798, and [Reference Lines](#) on page 183.

setReferenceLineValue()

This method sets the value at which the reference line displays.

Syntax:

```
void setReferenceLineValue(IdentOBJ id), value)
```

where:

id

Is the object ID of a reference line returned by a `getReferenceLineaxis()` method (for example, `getReferenceLineY1()`).

value

Is the value at which the reference line is drawn.

Return: *void*;

See [getReferenceLineY1\(\)](#) on page 1118.

setReferenceLineXYRatio()

This method sets a XY reference line on a graph based on a ratio. To use this method, the `setUseReferenceLineXYRatio()` method must be set to true.

Syntax:

```
void setReferenceLineXYRatio(IdentObj IDObj, value);
```

where:

IDObj

Can only be `getReferenceLineXY()`.

value

Can be one of the following:

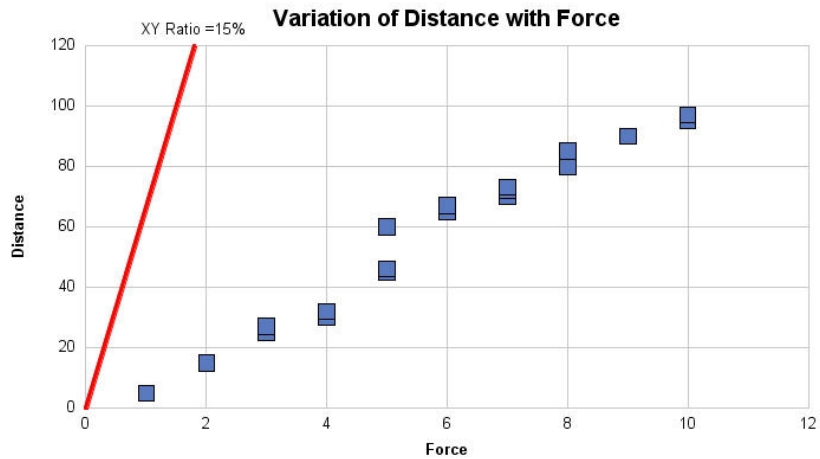
[True](#). When `setReferencePercentageMode` is set to True it will use the percent value of the top frame (0...100).

[False](#). When `setReferencePercentageMode` is set to False it will use the real value.

Example:

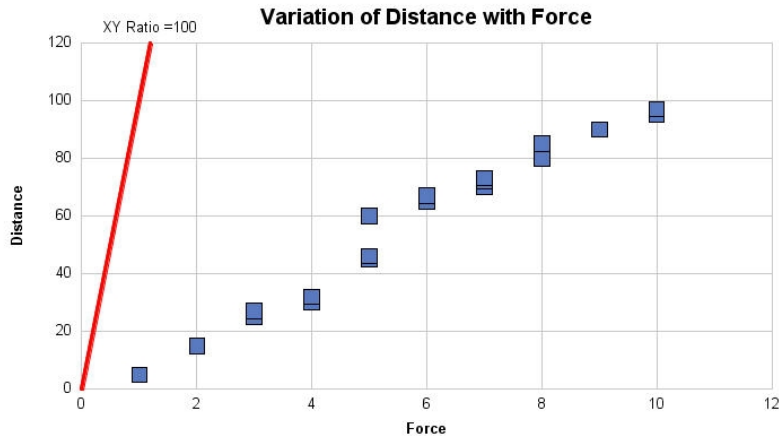
```
setBorderColor(getReferenceLineXY(0), new Color(255,0,0));  
setDisplay(getReferenceLineXY(0), true);  
setFillColor(getReferenceLineXY(0), new Color(255,0,0));  
setLineWidth(getReferenceLineXY(0), 3);  
setReferenceLineXYRatio(getReferenceLineXY(0), 15);  
setUseReferenceLineXYRatio(getReferenceLineXY(0), true);  
setReferencePercentageMode(getReferenceLineXY(0), true);
```

```
setDisplay(getReferenceLineXYText(0), true);
setReferenceTextPosition(getReferenceLineXYText(0), 0);
setTextString(getReferenceLineXYText(0), "XY Ratio =15%");
```



```
setBorderColor(getReferenceLineXY(0), new Color(255,0,0));
setDisplay(getReferenceLineXY(0), true);
setFillColor(getReferenceLineXY(0), new Color(255,0,0));
setLineWidth(getReferenceLineXY(0), 3);
setReferenceLineXYRatio(getReferenceLineXY(0), 100);
setUseReferenceLineXYRatio(getReferenceLineXY(0), true);
setReferencePercentageMode(getReferenceLineXY(0), false);
```

```
setDisplay(getReferenceLineXYText(0), true);
setReferenceTextPosition(getReferenceLineXYText(0), 0);
setTextString(getReferenceLineXYText(0), "XY Ratio =100");
```



See [setBorderColor\(\)](#) on page 860, [getReferenceLineXY\(\)](#) on page 799, [setDisplay\(\)](#) on page 883, [setFillColor\(\)](#) on page 890, [setLineWidth\(\)](#) on page 937, [ReferencePercentageMode](#) on page 501, [ReferenceTextPosition](#) on page 502, [setTextString\(\)](#) on page 1002, [setUseReferenceLineXYRatio\(\)](#) on page 1023

setScaleMax()

This method sets the maximum scale value assigned to a numeric axis.

Syntax:

```
void setScaleMax(IdentObj id, double newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getX1Axis()).

newValue

Is a value identifying the maximum scale value on an axis.

Return: void;

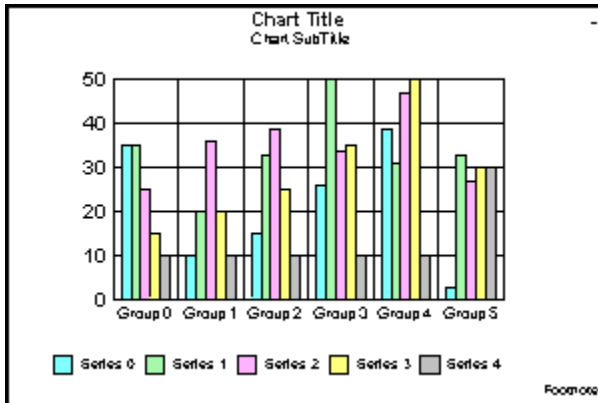
Example:

```
setDepthRadius(0);
```

```

setY1MajorGridStep(10.0);
setY1LogScale(false);  setScaleMax(getY1Axis(),50.0);
setY1ScaleMaxAuto(false);
setY1ScaleMinAuto(true);
setY1MustIncludeZero(false);

```

**Note:**

- ❑ You may also use one of the axis-specific properties to set the maximum scale on a particular axis (for example, `X1ScaleMax()`, `Y1ScaleMax()`, `Y2ScaleMax()`, and so on).
- ❑ If `setScaleMaxAuto()` sets autoscaling to true, the value set by `setScaleMax()` is ignored.

See [setScaleMax\(\)](#) on page 803, [setScaleMaxAuto\(\)](#) on page 969, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971.

setScaleMaxAuto()

This method sets automatic calculation of the maximum scale value for a numeric axis.

Syntax:

```
void setScaleMaxAuto(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getX1Axis()`).

newValue

Is true or false. If the value is:

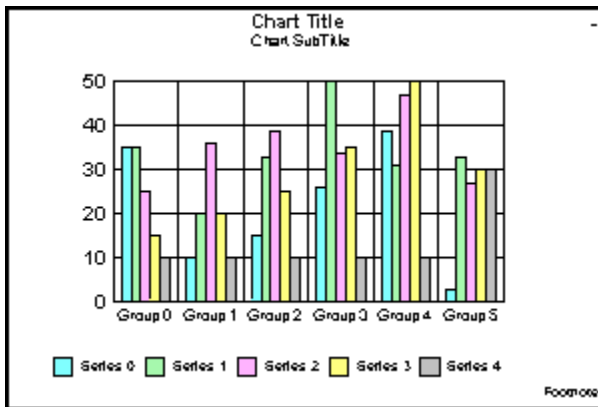
`true`, the maximum value is automatically calculated on the axis.

`false`, the value set by `setScaleMax()` is used.

Return: `void`;

Example:

```
setDepthRadius(0);
setY1MajorGridStep(10.0);
setY1LogScale(false);
setY1ScaleMax(50.0);  setScaleMaxAuto(getY1Axis(), false);
setY1ScaleMinAuto(true);
setY1MustIncludeZero(false);
```



Note:

- ❑ You may also use one of the axis-specific properties to set automatic scaling on a particular axis (for example, `X1ScaleMaxAuto()`, `Y1ScaleMaxAuto()`, `Y2ScaleMaxAuto()`, and so on).
- ❑ If this method sets a value of `true`, any value set by `setScaleMax()` is ignored.

See [setScaleMaxAuto\(\)](#) on page 804, [setScaleMax\(\)](#) on page 968, [setScaleMin\(\)](#) on page 970, [setScaleMinAuto\(\)](#) on page 971.

setScaleMin()

This method sets the minimum scale value assigned to a numeric axis in a graph.

Syntax:

```
void setScaleMin(IdentObj id, double newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getX1Axis()).

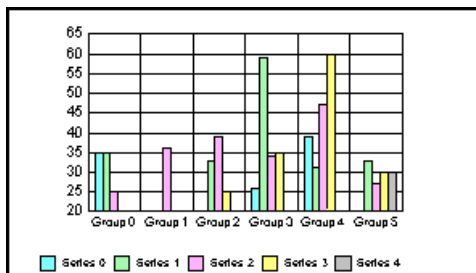
newValue

Is the minimum scaling value that can be used on an axis.

Return: void;

Example:

```
setDepthRadius(0);
setY1MajorGridStep(5.0);
setY1LogScale(false);
setY1ScaleMax(65.0);
setY1ScaleMaxAuto(true);  setScaleMin(getY1Axis(),20.0);
setY1ScaleMinAuto(false);
setY1MustIncludeZero(false);
```



Note:

- ❑ You may also use one of the axis-specific properties to set the minimum scale on a particular axis (for example, X1ScaleMin(), Y1ScaleMin(), Y2ScaleMin(), and so on).
- ❑ If setScaleMinAuto() sets autoscaling to true, the value set by setScaleMin() is ignored.

See [getScaleMin\(\)](#) on page 805, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969, [setScaleMinAuto\(\)](#) on page 971.

setScaleMinAuto()

This method sets automatic calculation of the maximum scale value for a numeric axis.

Syntax:

```
void setScaleMinAuto(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSlice ()`).

newValue

Is true or false. If the value is:

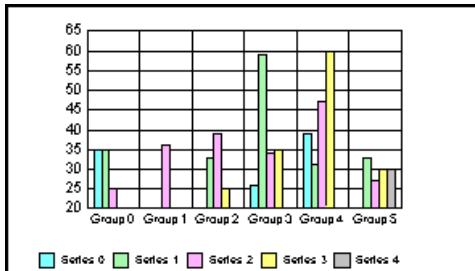
true, the minimum value is automatically calculated.

false, the value set by `setScaleMin()` is used.

Return: *void*;

Example:

```
setDepthRadius(0);
setY1MajorGridStep(5.0);
setY1LogScale(false);
setY1ScaleMax(65.0);
setY1ScaleMaxAuto(true);
setY1ScaleMin(20.0);  setScaleMinAuto(getY1Axis(), false);
setY1MustIncludeZero(false);
```



Note:

- ❑ You may also use one of the axis-specific properties to set automatic scaling on a particular axis (for example, `X1ScaleMinAuto()`, `Y1ScaleMinAuto()`, `Y2ScaleMinAuto()`, and so on).
- ❑ If this method sets a value of true, any value set by `setScaleMin()` is ignored.

See [setScaleMinAuto\(\)](#) on page 806, [setScaleMin\(\)](#) on page 970, [setScaleMax\(\)](#) on page 968, [setScaleMaxAuto\(\)](#) on page 969.

setScaleMustIncludeZero()

This method defines whether or not a given axis must include a zero value.

Syntax:

```
void setScaleMustIncludeZero(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getX1Axis()).

newValue

Is true or false. If the value is:

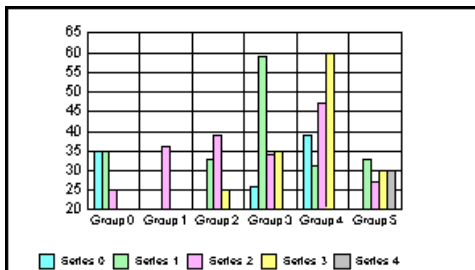
true, the axis must include zero.

false, zero is not required on the axis.

Return: *void*;

Example:

```
setDepthRadius(0);
setY1MajorGridStep(5.0);
setY1LogScale(false);
setY1ScaleMax(65.0);
setY1ScaleMaxAuto(true);
setY1ScaleMin(20.0);
setY1ScaleMinAuto(false); setScaleMustIncludeZero(getY1Axis(), false);
```

**Note:**

- ❑ You may also use one of the axis-specific properties to set a mandatory zero on a particular axis (for example, X1MustIncludeZero(), Y1MustIncludeZero(), Y2MustIncludeZero (), and so on).

See [setScaleMustIncludeZero\(\)](#) on page 807.

setSeriesAreRows()

This method can be used to store series values and rows or columns and to swap the order of series and group labels. In the normal default configuration (setSeriesAreRows(false, false)), each row/column value defines a series/group. When setSeriesAreRows(true, true|false) is used, each row/column value defines a group/series data point.

Syntax:

```
void setSeriesAreRows(  
    boolean newValue,  
    boolean swapLabels)
```

where:

newValue

Is true or false. If the value is:

true, series are stored as rows. This is the default value.

false, series are stored as columns (the reverse order).

swapLabels

Is true or false. If the value is:

true, series and group labels are swapped.

false, series and group labels are not swapped.

Note: The swapLabels value is only applicable when newValue is set to false.

Return: *void*;

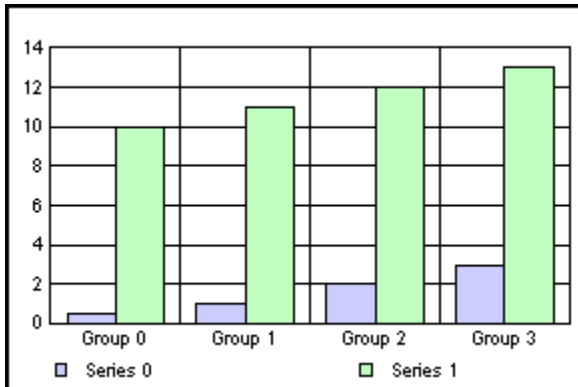
Example:

```
setUseSampleData(false);  
setDepthRadius(0);  
setData(0,0,.5); // Row 0/Column 0  
setData(0,1,1);  // Row 0/Column 1  
setData(0,2,2);  // Row 0/Column 2  
setData(0,3,3);  // Row 0/Column 3  
setData(1,0,10); // Row 1/Column 0  
setData(1,1,11); // Row 1/Column 1  
setData(1,2,12); // Row 1/Column 2  
setData(1,3,13); // Row 1/Column 3
```

```

setDataRangeToExtent();
setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");  setSeriesAreRows(true,true);

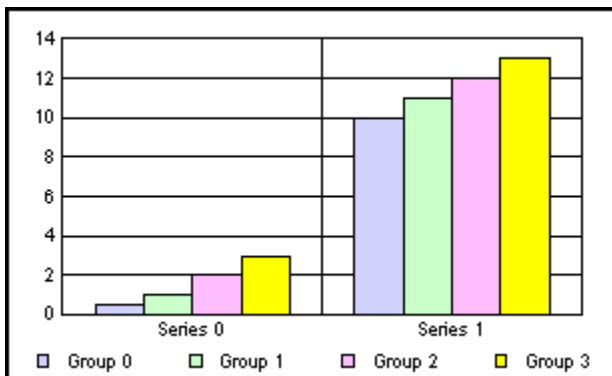
```



```

setSeriesAreRows(false,true);

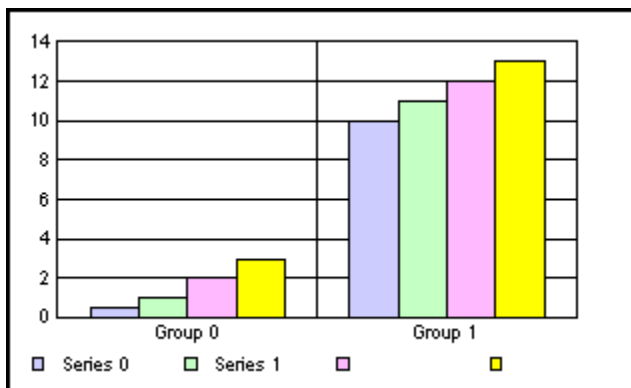
```



```

setSeriesAreRows(false,false);

```



setSeriesBorderColor()

This method can be used to set the border color of risers for a specified series in a graph.

Syntax:

```
void setSeriesBorderColor(int s, Color newValue);
```

where:

s

Is a series number.

newValue

Is an RGB color value.

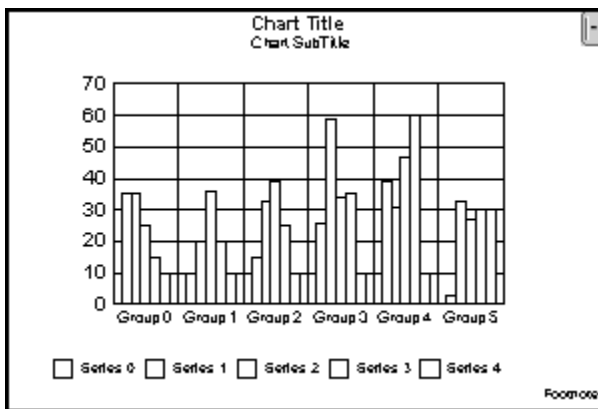
Return: void;

Example:

```

setSeriesBorderColor(4,new Color(0,0,0));
setDepthRadius(0);
setSeriesFillColor(4,new Color(255,255,255)); setSeriesBorderColor(3,new
    Color(0,0,0));
setSeriesFillColor(3,new Color(255,255,255)); setSeriesBorderColor(2,new
    Color(0,0,0));
setSeriesFillColor(2,new Color(255,255,255)); setSeriesBorderColor(1,new
    Color(0,0,0));
setSeriesFillColor(1,new Color(255,255,255)); setSeriesBorderColor(0,new
    Color(0,0,0));
setSeriesFillColor(0,new Color(255,255,255));

```



See [getBorderColor\(\)](#) on page 723, [setBorderColor\(\)](#) on page 860, [getTransparentBorderColor\(\)](#) on page 824, [setTransparentBorderColor\(\)](#) on page 1022, [setSeriesFillColor\(\)](#) on page 977.

setSeriesFillColor()

This method sets the fill color of risers for a specified series in a graph.

Syntax:

```
void setSeriesFillColor(int s, new Color (newValue));
```

where:

s

Is a series number.

newValue

Is a color value in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

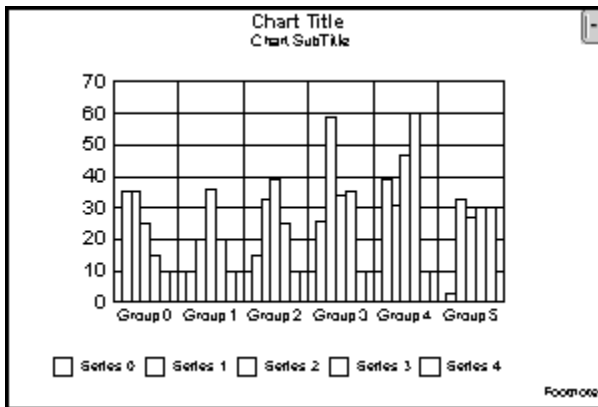
Return: *void*;

Example:

```
setSeriesBorderColor(4,new Color(0,0,0));  
setDepthRadius(0);  setSeriesFillColor(4,new Color(255,255,255));  
setSeriesBorderColor(3,new Color(0,0,0));  setSeriesFillColor(3,new  
Color(255,255,255));  
setSeriesBorderColor(2,new Color(0,0,0));  setSeriesFillColor(2,new  
Color(255,255,255));  
setSeriesBorderColor(1,new Color(0,0,0));  setSeriesFillColor(1,new  
Color(255,255,255));  
setSeriesBorderColor(0,new Color(0,0,0));  setSeriesFillColor(0,new  
Color(255,255,255));
```

or, using hexadecimal color codes:

```
setSeriesBorderColor(4,new Color(#000000));
setDepthRadius(0); setSeriesFillColor(4,new Color(#FFFFFF));
setSeriesBorderColor(3,new Color(#000000)); setSeriesFillColor(3,new
Color(#FFFFFF));
setSeriesBorderColor(2,new Color(#000000)); setSeriesFillColor(2,new
Color(#FFFFFF));
setSeriesBorderColor(1,new Color(#000000)); setSeriesFillColor(1,new
Color(#FFFFFF));
setSeriesBorderColor(0,new Color(#000000)); setSeriesFillColor(0,new
Color(#FFFFFF));
```



See [getFillColor\(\)](#) on page 746, [setFillColor\(\)](#) on page 890, [getTransparentFillColor\(\)](#) on page 825, [setTransparentFillColor\(\)](#) on page 1022, [setSeriesBorderColor\(\)](#) on page 976.

setSeriesLabel()

This method defines a label string associated with a series in a graph.

Syntax:

```
void setSeriesLabel(IdentObj id, String newValue);
void setSeriesLabel(int seriesID, String newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries()`).

seriesID

Is a series number.

newValue

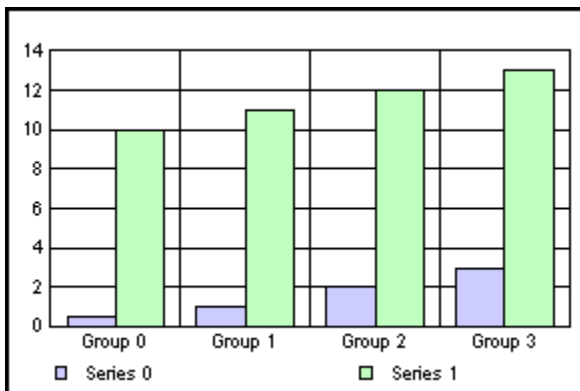
Is a series label string.

Return: *void*;

Example:

```
setUseSampleData(false);

setDepthRadius(0);
setData(0,0,.5); // Row 0/Column 0
setData(0,1,1);  // Row 0/Column 1
setData(0,2,2);  // Row 0/Column 2
setData(0,3,3);  // Row 0/Column 3
setData(1,0,10); // Row 1/Column 0
setData(1,1,11); // Row 1/Column 1
setData(1,2,12); // Row 1/Column 2
setData(1,3,13); // Row 1/Column 3
setDataRangeToExtent(); setSeriesLabel(0,"Series 0");
setSeriesLabel(1,"Series 1");
setGroupLabel(0,"Group 0");
setGroupLabel(1,"Group 1");
setGroupLabel(2,"Group 2");
setGroupLabel(3,"Group 3");
setSeriesAreRows(true,true);
```



See [getSeriesLabel\(\)](#) on page 809.

setSeriesLooping()

This method sets an index for the number of risers in a group. You can automatically shade successive groups of risers by lightening or darkening the colors used for the risers in the previous group. This type of shading is implemented using color mode 4, a looping index, and a factor for lightening or darkening the colors. For more information, see [Risers and Markers](#) on page 92 .

Syntax:

```
void setSeriesLooping(int newValue);
```

where:

newValue

Identifies the number of risers in each group on the graph.

Return: *void*;

See [Risers and Markers](#) on page 92, [ColorMode](#) on page 260, [setAutoColorLerpFactor\(\)](#) on page 848.

setSeriesType()

This method sets the series type for a specific object, series number, or the first item in the selection list.

Syntax:

```
void setSeriesType(IdentObj id, int newValue);
```

```
void setSeriesType(int seriesID, int newValue);
```

For a multi-axis merged graph:

```
void setSeriesType(int [$]seriesID, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries()).

seriesID

Is an object number.

\$seriesID

Automatically assigns a type to each series by using the number of measures as an increment that ensures that all series related to the same measure use the same series type. This automatic series increment is also available for [setAxisAssignment\(\)](#) on page 850.

newValue

Is a series type. Values can be:

0, riser unspecified

1, riser bar

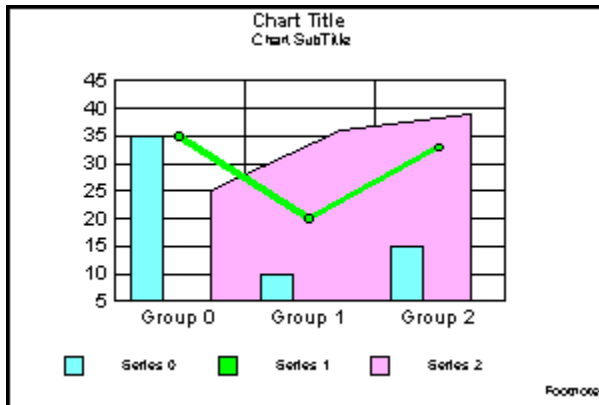
2, riser marker

3, riser area

Return: *void*;

Example:

```
setDataRange(0,0,2,2);
setDepthRadius(0); setSeriesType(0,1); setSeriesType(1,2);
setSeriesType(2,3);
```



See [getSeriesType\(\)](#) on page 810.

Example for a Multi-Axis Merged Graph

The following request uses a bi-polar bar chart with two measures (CURR_SAL and SALARY) and two sort fields (BY DEPARTMENT and ACROSS LAST_NAME). This makes it a multi-axis merged graph request:

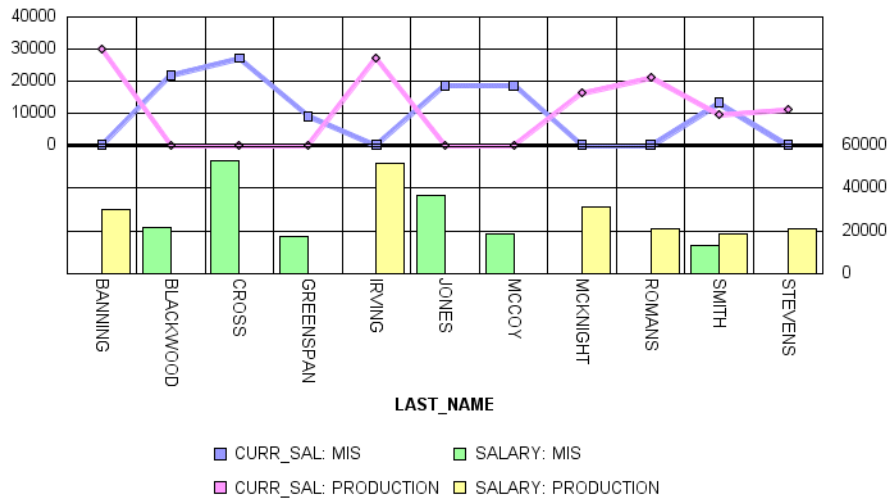
```
GRAPH FILE EMPLOYEE
SUM EMPLOYEE.EMPINFO.CURR_SAL
EMPLOYEE.PAYINFO.SALARY
BY EMPLOYEE.EMPINFO.DEPARTMENT
ACROSS EMPLOYEE.EMPINFO.LAST_NAME
ON GRAPH SET LOOKGRAPH VBAR2AXS
ON GRAPH SET 3D OFF
ON GRAPH SET BARNUMB OFF
ON GRAPH SET GRID ON
ON GRAPH SET GRAPHEDIT OFF
ON GRAPH SET GRMERGE ON
ON GRAPH SET VZERO ON
ON GRAPH SET HTMLENCODE ON
ON GRAPH SET STYLE *
*GRAPH_SCRIPT
- * ***Series Assignments***
  setSeriesType($0,2);
setSeriesType($1,1);
setAxisAssignment($0,0);
setAxisAssignment($1,1);
*END
```

Since DEPARTMENT has two values and there are two measures, there are four series in the graph request: CURR_SAL for the MIS department, CURR_SAL for the PRODUCTION department, SALARY for the MIS department, and SALARY for the PRODUCTION department

The series type assignments are setSeriesType(\$0,2); and setSeriesType(\$1,1). These are translated to the following API calls that make sure that series 0 and 2 (based on CURR_SAL) are represented by lines and that series 1 and 3 (based on SALARY) are represented by bars:

```
setSeriesType(0,2);
setSeriesType(2,2);
setSeriesType(1,1);
setSeriesType(3,1);
```

The output is:



setShadowColor()

This method sets the color value of a drop shadow.

Syntax:

```
void setShadowColor(IdentObj id, new Color (newValue));
```

where:

id

Is an object ID returned by a getObjectID() method.

newValue

Is the value of the color to be applied to the drop shadow ID in one of the following formats:

r, *g*, *b* is the desired intensity of red, green, and blue, respectively. The values are on a scale of 0 to 255, where 0 is the least intense and 255 is the most intense.

#hexcolor is the hexadecimal color code, preceded by a pound sign (#).

Return: void;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setShadowDisplay(getFrame(), true);
setShadowXOffset(getFrame(), 10);
setShadowYOffset(getFrame(), 10); setShadowColor(getFrame(), new Color
(0,255,255));

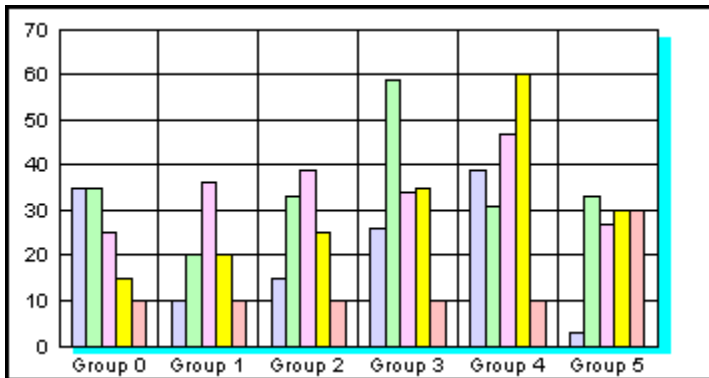
```

or, using a hexadecimal color code:

```

setDepthAngle(0);
setDepthRadius(0);
setShadowDisplay(getFrame(), true);
setShadowXOffset(getFrame(), 10);
setShadowYOffset(getFrame(), 10); setShadowColor(getFrame(), new Color
(#0000FF));

```



See [getShadowColor\(\)](#) on page 811, [getShadowDisplay\(\)](#) on page 812, [setShadowDisplay\(\)](#) on page 985, [getShadowXOffset\(\)](#) on page 814, [setShadowXOffset\(\)](#) on page 989, [getShadowYOffset\(\)](#) on page 814, [setShadowYOffset\(\)](#) on page 990.

setShadowDisplay()

This method applies a drop shadow to an object.

Syntax:

```
void setShadowDisplay(IdentObj id, boolean newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getTitle()).

newValue

Is true or false. If the value is:

true, the drop shadow appears.

false, the drop shadow does not appear.

Return: *void*;

See [setShadowColor\(\)](#) on page 984 for an example. Also see [getShadowColor\(\)](#) on page 811, [getShadowXOffset\(\)](#) on page 814, [setShadowXOffset\(\)](#) on page 989, [getShadowYOffset\(\)](#) on page 814, [setShadowYOffset\(\)](#) on page 990, [ShadowXOffsetDefault](#) on page 530, [ShadowYOffsetDefault](#) on page 530

setShadowFillType()

When shadow is enabled for a specified text object (for example, `setShadowDisplay(getTitle(), true)`), this method assigns a shadow fill type to the object. The default value is 1 (FILL_COLOR). If FILL_GRADIENT_LEFTTORIGHT (2) is used, use `setShadowGradientStartColor()` and `setShadowGradientEndColor()` to specified the starting and ending colors of the gradient. If FILL_TEXTURE (3) is used, use `setShadowTextureURL()` to define the texture file.

Syntax:

```
void setShadowFillType(IdentObj id, int newValue)
```

where:

id

object ID returned by a `getObject()` method (for example, `getTitle()`)

newValue

1...3. FILL_COLOR (1), FILL_GRADIENT_LEFTTORIGHT (2), FILL_TEXTURE (3)

Return: *void*;

See [setShadowGradientStartColor\(\)](#) on page 987 and [setShadowTextureURL\(\)](#) on page 988 for examples. Also see [getShadowFillType\(\)](#) on page 812, [setShadowDisplay\(\)](#) on page 985, [setShadowGradientEndColor\(\)](#) on page 986.

setShadowGradientEndColor()

When `setShadowFillType()` assigns a gradient (FILL_GRADIENT_LEFTTORIGHT (2)) to a text object, this method defines the ending color of a gradient for a specified object. The default value is green.

Syntax:

```
void setShadowGradientEndColor(IdentObj idObj, Color col)
```

where:

id

Is the object ID of a Java2D text object returned by a *getObject()* method (for example, *getTitle()*).

col

Is the color specification (for example, `new Color (255,0,0) // red`).

Return: *void*;

See [setShadowGradientStartColor\(\)](#) on page 987 for an example. Also see [getShadowGradientEndColor\(\)](#) on page 812, [setShadowFillType\(\)](#) on page 986.

setShadowGradientStartColor()

When `setShadowFillType()` assigns a gradient (`FILL_GRADIENT_LEFTTORIGHT (2)`) to a text object, this method defines the starting color of a gradient for a specified object. The default value is blue.

Syntax:

```
void setShadowGradientStartColor(
    IdentObj idObj,
    Color col)
```

where:

idObj

Is the object ID of a Java2D text object returned by a *getObject()* method (for example, *getTitle()*).

col

Is the color specification (for example, `new Color (255,0,0) // red`).

Return: *void*;

Example:

```
// Gradient on the title
setFillType(getTitle(),2);
setGradientDirection(getTitle(),1);
setGradientNumPins(getTitle(),2);
setGradientPinLeftColor(getTitle(),new Color(0,255,0),1);
setGradientPinLeftColor(getTitle(),new Color(0,0,255),0);
setGradientPinPosition(getTitle(),1.0,1);
setGradientPinPosition(getTitle(),0.0,0);
setGradientPinRightColor(getTitle(),new Color(0,255,0),1);
setGradientPinRightColor(getTitle(),new Color(0,0,255),0);
// Gradient on the drop shadow
setShadowDisplay(getTitle(),true);
setShadowFillType(getTitle(),
Java2DTextLowLevel.FILL_GRADIENT_LEFTRIGHT);
setShadowTextureURL(getTitle(),"file:/C:/JavaChart/pfjFull/tiles/Azul_Gra
nite.gif");    setShadowGradientStartColor(getTitle(),new Color(0,255,0));

setShadowGradientEndColor(getTitle(),new Color(255,0,0));
```

Chart Title
Chart SubTitle

See [getShadowGradientStartColor\(\)](#) on page 813, [setShadowFillType\(\)](#) on page 986, [setShadowGradientEndColor\(\)](#) on page 986.

setShadowTextureURL()

When `setShadowFillType()` assigns a texture (`FILL_TEXTURE (3)`) to a text object, this method defines the texture file to be applied to the object. This method is identical to `setTextureURL()` except it only applies to Java 2D Text.

Syntax:

```
void setShadowTextureURL(IdentObj id, String szURL)
```

where:

id

Is the object ID of a text object returned by a `getObject()` method.

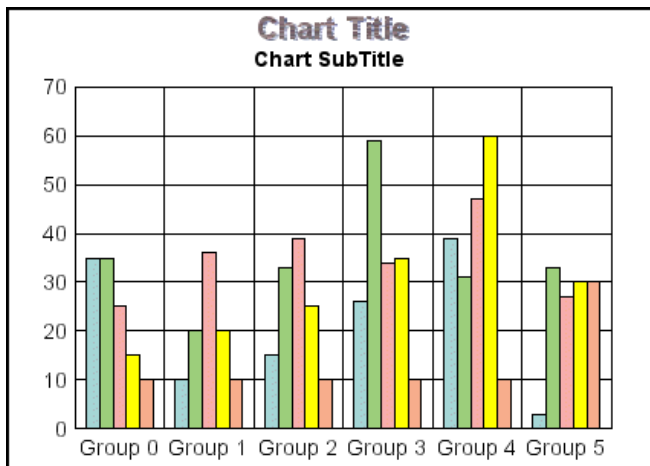
szURL

Is the URL specification string that identifies a texture file.

Return: *void*;

Example:

```
// Texture on the front
setFillType(getTitle(),3);
setTextureDisplayMode(getTitle(),0);
setTextureURL(getTitle(),"/images/tdg/Purple_Pink_Granite.gif");
// Texture on the drop shadow
setShadowDisplay(getTitle(),true);
setShadowFillType(getTitle(),3);
setShadowTextureURL(getTitle(),"/images/tdg/Azul_Granite.gif");
```



See [setShadowFillType\(\)](#) on page 986.

setShadowXOffset()

This method sets the offset in the X-direction of a drop shadow.

Syntax:

```
void setShadowXOffset(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getTitle()`).

newValue

Is the X-direction offset of the drop shadow applied to *id*. A value of null means that no drop shadow is applied.

Return: *void*;

See [setShadowColor\(\)](#) on page 984 for an example. Also see [getShadowXOffset\(\)](#) on page 814, [setShadowDisplay\(\)](#) on page 985, [setShadowYOffset\(\)](#) on page 990, [ShadowXOffsetDefault](#) on page 530, [ShadowYOffsetDefault](#) on page 530.

setShadowYOffset()

These methods get/set the offset in the Y-direction of a drop shadow.

Syntax:

```
void setShadowYOffset(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getTitle()).

newValue

Is the Y-direction offset of the drop shadow applied to id. If the value is null, then no drop shadow is applied.

Return: *void*;

See [setShadowColor\(\)](#) on page 984 for an example. Also see [getShadowYOffset\(\)](#) on page 814, [setShadowDisplay\(\)](#) on page 985, [setShadowXOffset\(\)](#) on page 989, [ShadowXOffsetDefault](#) on page 530, [ShadowYOffsetDefault](#) on page 530.

setSize()

This method resizes the graph to the specified height and width in pixels.

Syntax:

```
void setSize(Dimension d)
```

```
void setSize(int width, int height)
```

where:

d

Is the dimension of the object.

width

Is the width, in pixels.

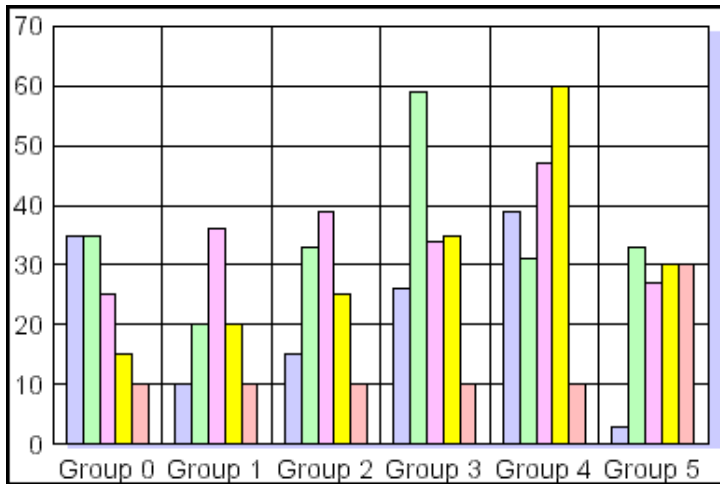
height

Is the height, in pixels.

Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setShadowDisplay(getFrame(), true);
setShadowXOffset(getFrame(), 10);
setShadowYOffset(getFrame(), 5);
setShadowColor(getFrame(), new Color (0,0,255)); setSize(200,200);
```



setSkipBegin()

When setAutoSkip() selects manual skip mode, this method selects the first label to skip on the O1 axis.

Syntax:

```
void setSkipBegin(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getO1Label ()).

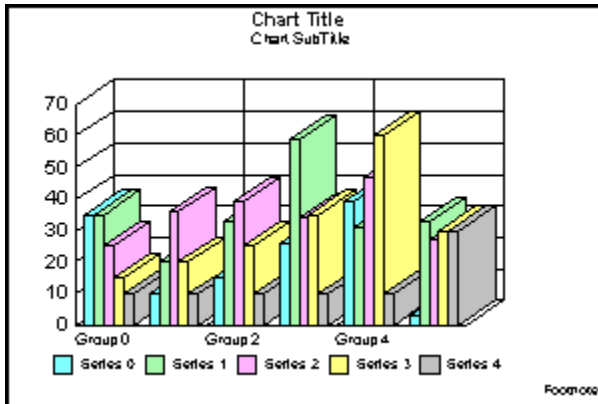
newValue

0...Number of groups in the graph.

Return: *void*;

Example:

```
setAutoSkip(getO1Label(),2);  setSkipBegin(getO1Label(),1);
setSkipCount(getO1Label(),1);
```



Note: If an object ID (id) is not specified, `newValue` is assigned to the first item in the selection list (if applicable).

See [setAutoSkip\(\)](#) on page 849, [setSkipCount\(\)](#) on page 992.

setSkipCount()

When `setAutoSkip()` selects manual skip mode, this method selects the interval of labels to skip after `setSkipBegin()`.

Syntax:

```
void setSkipCount(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getO1Label()`).

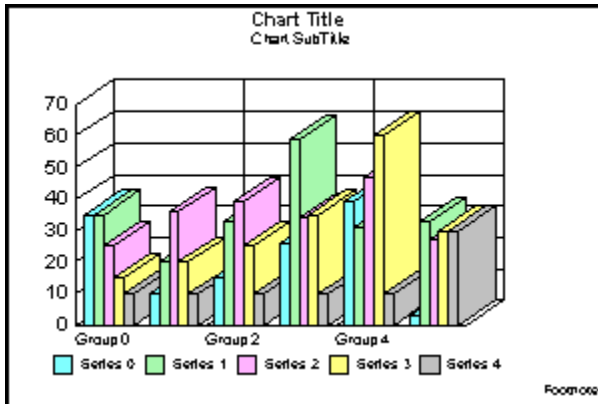
newValue

Is the number (0...Number) of groups in the graph.

Return: `void`;

Example:

```
setAutoSkip(get01Label(),2);
setSkipBegin(get01Label(),1);  setSkipCount(get01Label(),1);
```



Note: If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [setAutoSkip\(\)](#) on page 849, [setSkipBegin\(\)](#) on page 991.

setTextBackgroundColor()

When `setUseTextBackgroundColor()` enables (true) drawing of a background color behind a text object, this method assigns the background color.

Syntax:

```
void setTextBackgroundColor(IdentObj idObj, Color col)
```

where:

idObj

Is the object ID returned by a *getObjectID()* method (for example, `getY1Label()`).

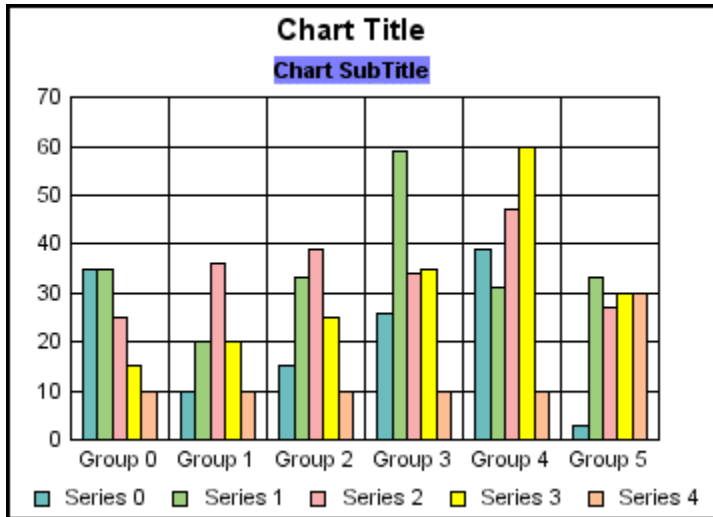
col

Is a color specification.

Return: void;

Example:

```
setUseTextBackgroundColor(getSubtitle(), true);
setTextBackgroundColor(getSubtitle(), new Color(128,128,255));
```



See [setUseTextBackgroundColor\(\)](#) on page 1025.

setTextFormatPattern()

This method assigns a standard Java number format pattern to numeric data text or a numeric label object.

Syntax:

```
void setTextFormatPattern(IdentObj id, String newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Label()).

newValue

Is a pattern string in the following format:

```
pattern:= subpattern{;subpattern}subpattern:=
{prefix}integer{.fraction}{suffix}prefix:= '\\u0000'..'\\uFFFF' -
specialCharacterssuffix:= '\\u0000'..'\\uFFFF' -
specialCharactersinteger:= '#'* '0'* '0'fraction:= '0'* '#'*
```

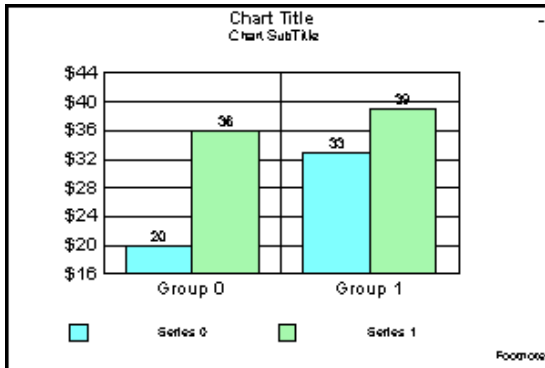
Return: void;

Example:

```

setDepthRadius(0);
setDataRange(1,1,2,2);
setDataTextDisplay(true);
setTextFormatPreset(getY1Label(), -1);
setTextFormatPattern(getY1Label(), "$##");

```



See [getTextFormatPattern\(\)](#) on page 815, [getTextFormatPreset\(\)](#) on page 816, [setTextFormatPreset\(\)](#) on page 995, [DataTextFormat](#) on page 318, [PieFeelerTextFormat](#) on page 478, [PieRingTotalFormat](#) on page 486, [X1LabelFormat](#) on page 584, [Y1LabelFormat](#) on page 625, [Y2LabelFormat](#) on page 661.

setTextFormatPreset()

This method assigns a number format to a numeric text object.

Syntax:

```
void setTextFormatPreset(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Label()).

newValue

Is one of the following values:

-2 uses the format defined by setTextFormatPattern(), divided by 100.

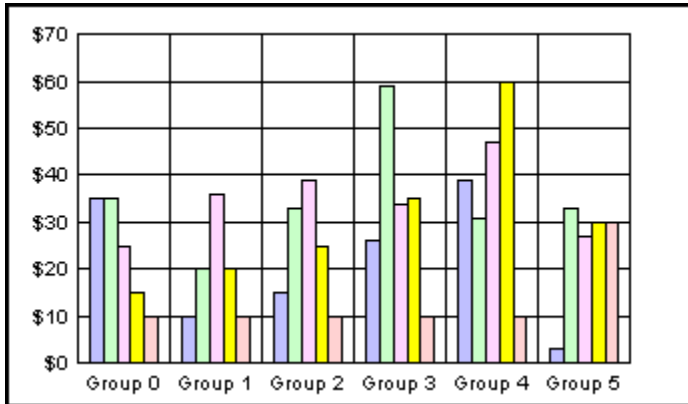
-1 uses the format defined by setTextFormatPattern().

0...21, 24...30 selects one of the preset formats. See [Preset Number Formats](#) on page 130.

Return: *void*;

Example:

```
setDepthRadius(0); setDepthAngle(0); setTextFormatPreset(getY1Label(),
6);
```



See [getTextFormatPreset\(\)](#) on page 816, [getTextFormatPattern\(\)](#) on page 815, [setTextFormatPattern\(\)](#) on page 994, [DataTextFormat](#) on page 318, [PieFeelerTextFormat](#) on page 478, [PieRingTotalFormat](#) on page 486, [X1LabelFormat](#) on page 584.

setTextJustHoriz()

This method selects the horizontal alignment of a text object within its bounding box.

Syntax:

```
void setTextJustHoriz(IdentObj id, int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getFootnote()).

newValue

Is the text justification. Values can be:

0 (left)

1 (center)

2 (tight)

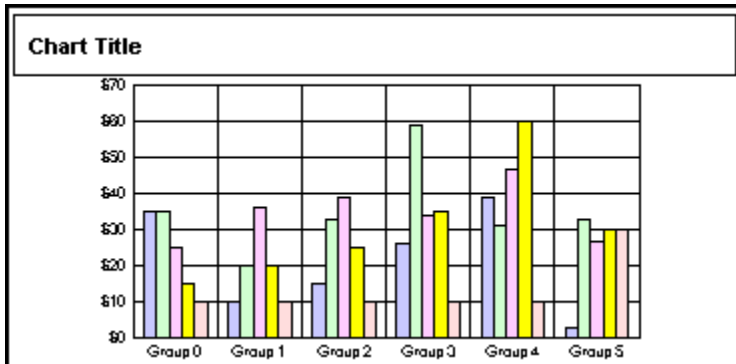
Return: *void*;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getTitleBox(), true);
setFillColor(getTitleBox(), new Color(255,255,255));
setSubtitleString("");
setTextJustVert(getTitle(), 1);  setTextJustHoriz(getTitle(), 0);

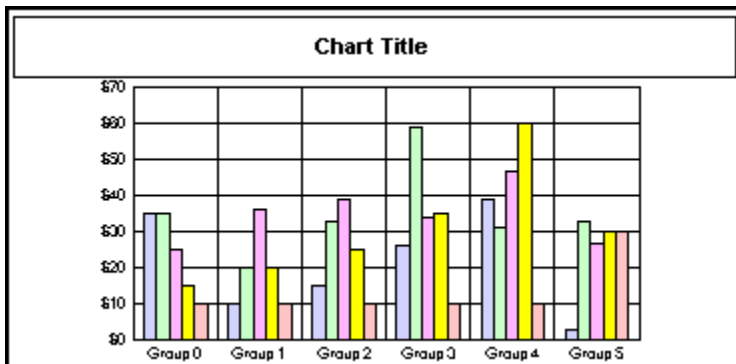
```



```

setTextJustHoriz(getTitle(), 1);

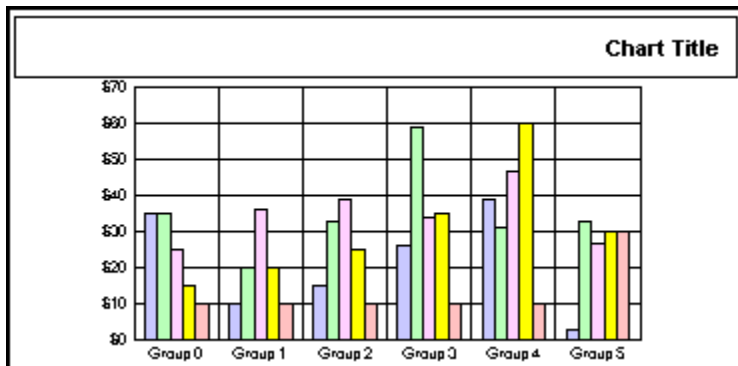
```



```

setTextJustHoriz(getTitle(), 2);

```



See [getTextJustHoriz\(\)](#) on page 817, [getTextJustVert\(\)](#) on page 818, [setTextJustVert\(\)](#) on page 998.

setTextJustVert()

This method selects the vertical alignment of a text object within its bounding box.

Syntax:

```
void setTextJustVert(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getTitle()).

newValue

Is the vertical justification. Values can be:

0 (top)

1 (center)

2 (bottom)

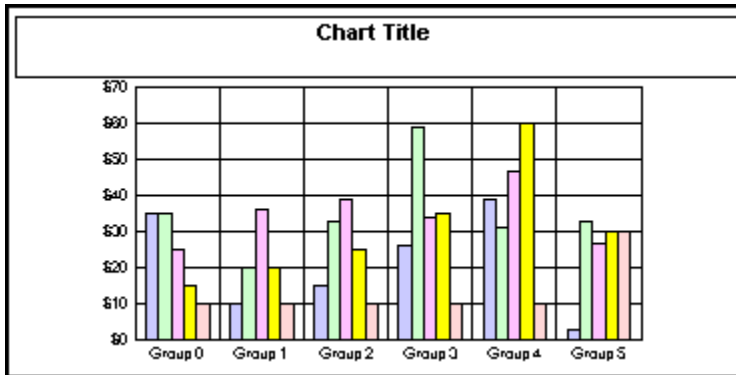
Return: void;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getTitleBox(), true);
setFillColor(getTitleBox(), new Color(255,255,255));
setSubtitleString("");  setTextJustVert(getTitle(), 0);

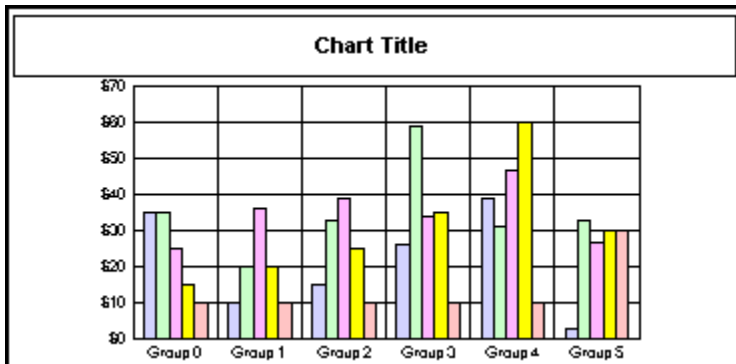
```



```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getTitleBox(), true);
setFillColor(getTitleBox(), new Color(255,255,255));
setSubtitleString("");  setTextJustVert(getTitle(), 1);

```

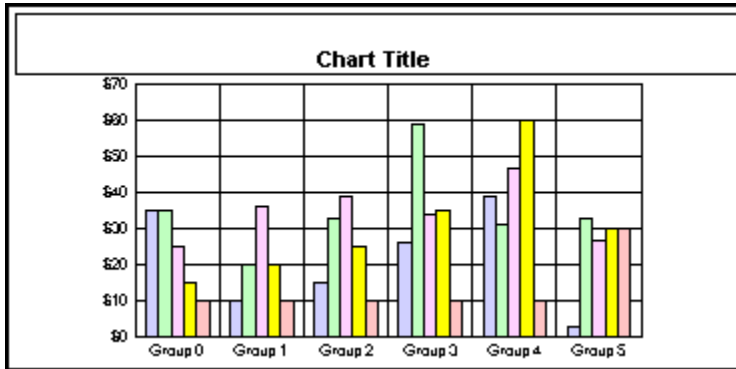


```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getTitleBox(), true);
setFillColor(getTitleBox(), new Color(255,255,255));
setSubtitleString("");

```

```
setTextJustVert(getTitle(), 2);
```



See [getTextJustVert\(\)](#) on page 818, [getTextJustHoriz\(\)](#) on page 817, [setTextJustHoriz\(\)](#) on page 996.

setTextRotation()

This method selects the rotation angle of a text object.

Syntax:

```
void setTextRotation(IdentObj id,int newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getY1Label()).

newValue

Is the text rotation. Values can be:

0 (none)

1 (90 degrees)

2 (270 degrees)

3 (45 degrees)

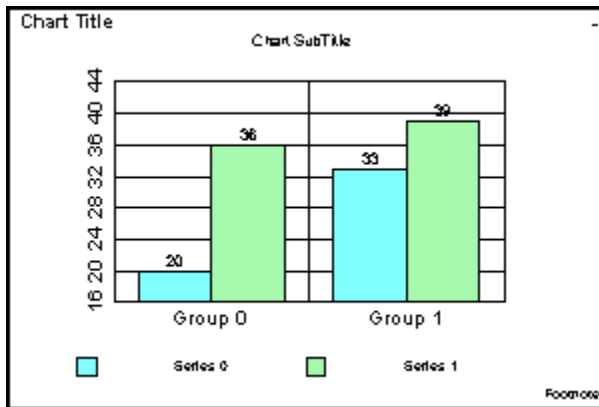
Return: void;

Example:

```

setDepthRadius(0);
setUseSampleData(false);
setData(0,0,20);
setData(0,1,36);
setData(1,0,33);
setData(1,1,39);
setDataRangeToExtent();
setTextJustHoriz(getTitle(),0);
setTextJustHoriz(getSubtitle(),1);
setTextJustHoriz(getFootnote(),2); setTextRotation(getY1Label(),1);

```



See [getTextRotation\(\)](#) on page 818, [O1LabelRotate](#) on page 435, [O2LabelRotate](#) on page 459, [X1LabelRotate](#) on page 586, [Y1LabelRotate](#) on page 627, [Y2LabelRotate](#) on page 663.

setTextStrikethrough()

This method enables (true) / disables (false) strike-through mode on a text object.

Syntax:

```

void setTextStrikethrough(
    IdentObj idObj,
    boolean newValue
)

```

where:

idObj

Is the object ID returned by a *getObjectID()* method (for example, *getTitle()*).

newValue

Can be one of the following values:

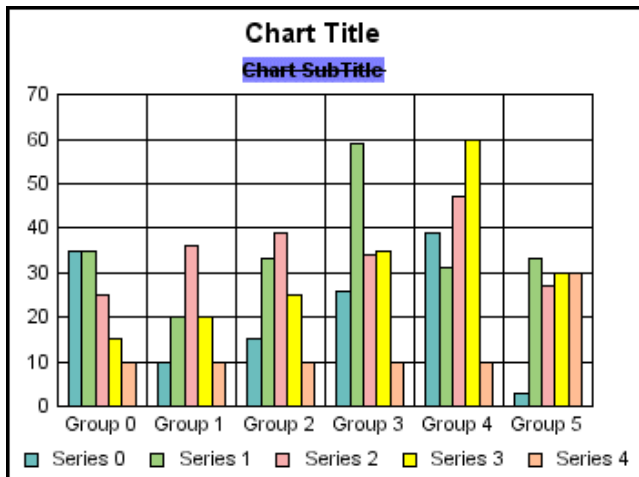
`true` enables strike-through.

`false` disables strike-through.

Return: `void`;

Example:

```
setUseTextBackgroundColor(getSubtitle(), true);
setTextBackgroundColor(getSubtitle(), new Color(128,128,255));
setTextStrikethrough(getSubtitle(), true);
```



setTextString()

This method defines a text string to be assigned to a text object.

Syntax:

```
void setTextString(IdentObj id,String newValue);
```

where:

id

Is an object ID returned by a getObjectID() method (for example, getTitle()).

newValue

Is the string assigned to a text object.

Return: `void`;

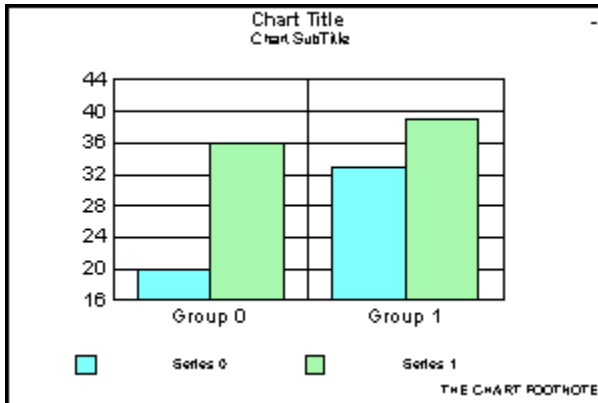
Example:

```
setDepthRadius(0);
```

```

setUseSampleData(false);
setData(0,0,20);
setData(0,1,36);
setData(1,0,33);
setData(1,1,39);
setDataRangeToExtent(); setTextString(getFootnote(), "THE CHART FOOTNOTE");

```



See [getTextString\(\)](#) on page 819, [FootnoteString](#) on page 357, [O1TitleString](#) on page 454, [O2TitleString](#) on page 466, [SubtitleString](#) on page 550, [TitleString](#) on page 555, [X1TitleString](#) on page 615, [Y1TitleString](#) on page 650, [Y2TitleString](#) on page 689.

setTextureBGColor()

This method assigns a background color to a texture for a specified object.

Syntax:

```

void setTextureBGColor(IdentObj id, Color newValue)
void setTextureBGColor(int seriesID, Color newValue)
void setTextureBGColor(
    int seriesID,
    int groupID,
    Color newValue)

```

where:

newValue

Is a color (for example, new Color (255,0,0)) to be applied to the background of a texture for a specified object.

id

Is the object ID returned by a [getObjectID\(\)](#) method.

seriesID

Is a series number.

groupID

Is a group number.

Return: *void*;

See [*getTextureBGColor\(\)*](#) on page 820.

setTextureDisplayMode()

This method sets the texture display mode.

Syntax:

```
void setTextureDisplayMode(int newValue);
```

```
void setTextureDisplayMode(IdentObj id, int newValue);
```

where:

id

Is the object ID returned by a getObjectID() method (for example, getSeries()).

newValue

Is a texture display mode:

0 is stretched.

1 is tiles.

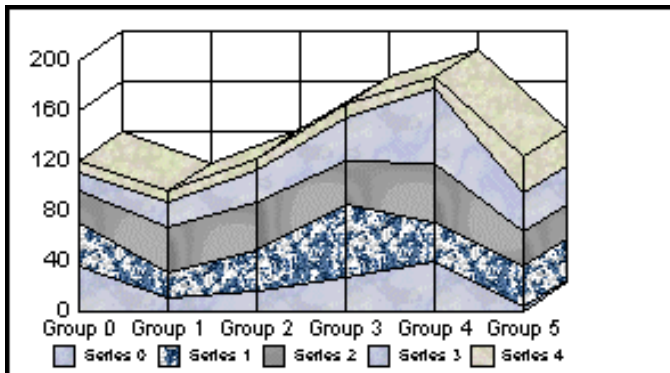
Return: *void*;

Example:

```

setFillType(getSeries(3),3);
setFillType(getSeries(0),3);
setFillType(getSeries(2),3);
setFillType(getSeries(4),3);
setFillType(getSeries(1),3);
setGraphType(32);  setTextureDisplayMode(getSeries(3),0);
setTextureDisplayMode(getSeries(0),0);
setTextureDisplayMode(getSeries(2),0);
setTextureDisplayMode(getSeries(4),0);
setTextureDisplayMode(getSeries(1),0);
setTextureURL(getSeries(3),"/images/tdg/Soft_Blue_Marble.gif");
setTextureURL(getSeries(0),"/images/tdg/Soft_Blue_Marble.gif");
setTextureURL(getSeries(2),"/images/tdg/Oil.gif");
setTextureURL(getSeries(4),"/images/tdg/Yellow.gif");
setTextureURL(getSeries(1),"/images/tdg/Azul_Granite.gif");

```



Note: If an object ID (id) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [setFillType\(\)](#) on page 892, [setTextureURL\(\)](#) on page 1007.

setTextureFlipMode()

This method can be used to change the direction in which a texture is drawn for a selected object.

Syntax:

```
void setTextureFlipMode(int newValue)
```

```
void setTextureFlipMode(IdentObj id, int newValue)
```

where:

id

Is an object ID returned by a getObject() method (for example, getSeries()).

newValue

Is the flip mode. Values can be:

0 (no flipping)

1 (flip vertical)

2 (flip horizontal)

3 (flip vertical and horizontal)

Return: void;

Note: If an object ID (id) is not specified, newValue is assigned to the first item in the selection list (if applicable).

See [getTexture\(\)](#) on page 819, [getTextureDisplayMode\(\)](#) on page 821, [getTextureFlipMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004, [setTextureResource\(\)](#) on page 1006, [setTextureURL\(\)](#) on page 1007.

setTextureResource()

This method assigns a texture from a file to a specified graph object.

Syntax:

`void setTextureResource(IdentObj id, String szResource)`

where:

id

Is an object ID returned by a getObjectID() method (for example, getSeries()).

szResource

Is a texture file name.

Return: void;

See [getTexture\(\)](#) on page 819, [getTextureDisplayMode\(\)](#) on page 821, [getTextureFlipMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004, [setTextureFlipMode\(\)](#) on page 1005, [setTextureURL\(\)](#) on page 1007.

setTextureURL()

This method can be used to load a texture image from a URL/file and assign it to object(s) in the graph. Note that the `setFillType()` method be used to select a texture fill type.

Syntax:

```
void setTextureURL(String szURL)
void setTextureURL(IdentObj id, String szURL)
void setTextureURL(String szURL, Image theImage)
void setTextureURL(IdentObj id, String szURL, Image theImage)
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getSeries()`).

theImage

Is a URL/file name where the texture image resides.

szURL

Is the size of the URL.

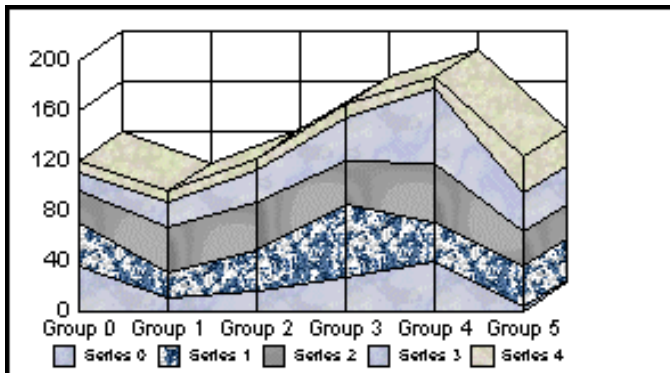
Return: *void*;

Example:

```

setFillType(getSeries(3),3);
setFillType(getSeries(0),3);
setFillType(getSeries(2),3);
setFillType(getSeries(4),3);
setFillType(getSeries(1),3);
setGraphType(32);
setTextureDisplayMode(getSeries(3),0);
setTextureDisplayMode(getSeries(0),0);
setTextureDisplayMode(getSeries(2),0);
setTextureDisplayMode(getSeries(4),0);
setTextureDisplayMode(getSeries(1),0);
setTextureURL(getSeries(3),"/images/tdg/Soft_Blue_Marble.gif");
setTextureURL(getSeries(0),"/images/tdg/Soft_Blue_Marble.gif");
setTextureURL(getSeries(2),"/images/tdg/Oil.gif");
setTextureURL(getSeries(4),"/images/tdg/Yellow.gif");
setTextureURL(getSeries(1),"/images/tdg/Azul_Granite.gif");

```



See [setFillType\(\)](#) on page 892, [getTextureDisplayMode\(\)](#) on page 821, [setTextureDisplayMode\(\)](#) on page 1004.

setTextWrap()

This method enables/disables line wrapping for a text object. When text wrapping is enabled, the graphing engine will look for space characters in the text string and draw the text on multiple lines at space character boundaries. When text wrapping is disabled, the text string will always be drawn on a single line.

Syntax:

```
void setTextWrap(IdentObj id,boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method (for example, `getO1Label()`).

boolean

Can be one of the following values:

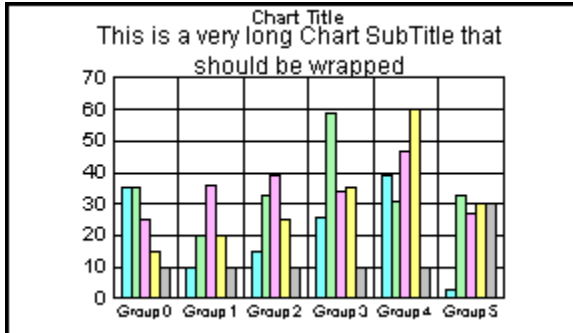
`true` enables text wrapping.

`false` disables text wrapping.

Return: `void`;

Example:

```
setAutofit(getSubtitle(), false);
setFontSizeVC(getSubtitle(), 1855);
setRect(getSubtitle(),
    new Rectangle(-13278, 10471, 25806, 3710));
setTextString(getSubtitle(),
    "This is a very long Chart SubTitle that
    should be wrapped"); setTextWrap(getSubtitle(), true);
setDepthRadius(0);
```



Note If a "\n" is encountered in the label string, the string will be wrapped even if this method is disabled (for example, `setTextWrap(getO2Label(), false)`).

See [getTextWrap\(\)](#) on page 822, [O1LabelWrap](#) on page 441, [O2LabelWrap](#) on page 463.

setTickCount()

This method can be used to set the number of minor tick marks that will appear between major tick marks on an ordinal axis.

Syntax:

```
void setTickCount(IdentObj id, int newValue)
```

where:

newValue

Is the number of minor tick marks between major tick marks.

id

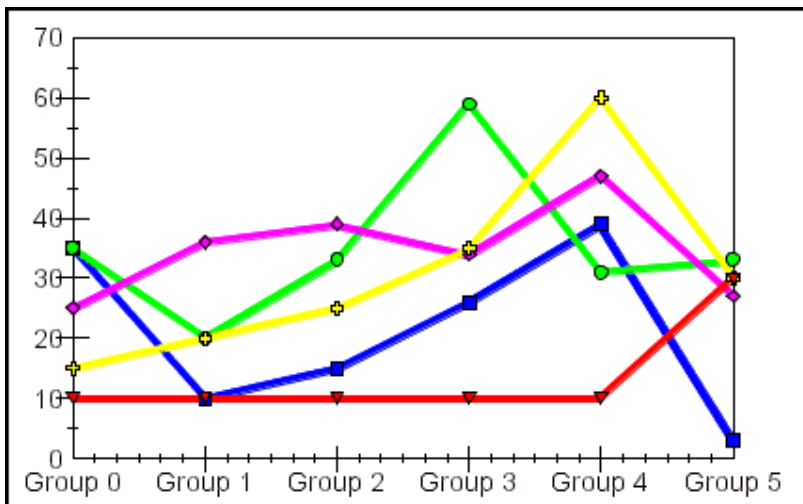
Is the object ID returned by a *getObject()* method for ordinal axis tick marks (for example, *getO1MinorTick()*).

Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setO1MajorGridDisplay(false);
setO1MajorTickDisplay(true);
setTickLength(getO1MajorTick(), 1000);
setO1MinorTickDisplay(true);
setO1MinorTickStepAuto(false);

setTickCount(getO1MinorTick(), 5);
setY1MajorGridDisplay(false);
setY1MinorTickDisplay(true);
setY1MajorTickDisplay(true);
```



See [getTickCount\(\)](#) on page 822.

setTickLength()

The method sets the length of tick marks to be drawn on a specified axis.

Syntax:

```
void setTickLength(IdentObj id, int newValue)
```

where:

newValue

Is the length of tick marks in virtual coordinates.

id

Is the object ID returned by a [getObject\(\)](#) method for axis tick marks (for example, [getY1MajorTick\(\)](#)).

Return: *void*;

See [setTickCount\(\)](#) on page 1009 for an example. Also see [getTickLength\(\)](#) on page 822.

setTickStep()

When automatic calculation of tick steps is disabled with [setTickStepAuto\(axis, false\)](#), this method sets the interval at which to draw tick marks on an axis.

Syntax:

```
void setTickStep(IdentObj id, double newValue)
```

where:

newValue

Is the interval at which to draw ticks on an axis.

id

Is the object ID returned by a [getObject\(\)](#) method for axis tick marks (for example, [getY1MajorTick\(\)](#)).

Return: *void*;

Note: The axis-specific properties may also be used to enable/disable automatic calculation of tick steps or define a number tick steps (for example, [Y1MinorTickStepAuto](#)).

See [setTickStepAuto\(\)](#) on page 1012 for an example. Also see [getTickStep\(\)](#) on page 823.

setTickStepAuto()**Syntax:**

```
void setTickStepAuto(IdentObj id, boolean newValue)
```

where:

newValue

Is the interval at which to draw ticks on an axis.

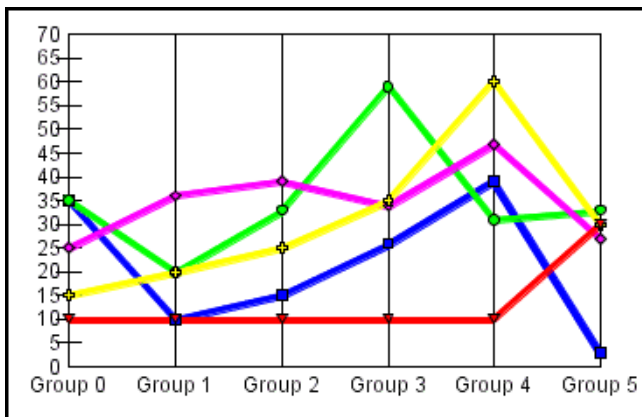
id

Is the object ID returned by a *getObject()* method for axis tick marks (for example, *getY1MajorTick()*).

Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setTickLength(getY1MajorTick(), 1000);
setY1MajorGridDisplay(false);
setY1MinorTickDisplay(false);
setY1MajorTickDisplay(true);
setTickStep(getY1MajorTick(), 5); setTickStepAuto(getY1MajorTick(), false);
```



Note: The axis-specific properties may also be used to enable/disable automatic calculation of tick steps or define a number tick steps (for example, *Y1MinorTickStepAuto*).

See [getTickStepAuto\(\)](#) on page 823.

setTickStyle()

The method sets the style of tick marks to be drawn on a particular axis.

Syntax:

```
void setTickStyle(IdentObj id, int newValue)
```

where:

newValue

Can be one of the following:

- 0 for small tick marks from frame edge outward.
- 1 for small tick marks from frame edge outward.
- 2 for small tick marks from frame edge inward.
- 3 for small tick marks from frame edge outward.
- 4 for small tick marks span across the frame edge.

id

Is the object ID returned by a *getObject()* method for axis tick marks (for example, *getY1MajorTick()*).

Return: *void*;

See [getTickStyle\(\)](#) on page 823.

setTickWidth()

The method sets the width of tick marks to be drawn on a particular axis.

Syntax:

```
void setTickWidth(IdentObj id, int newValue)
```

where:

newValue

Is the width of ticks in device coordinates.

id

Is the object ID returned by a *getObject()* method for axis tick marks (for example, *getY1MajorTick()*).

Return: *void*;

See [getTickWidth\(\)](#) on page 824.

setToolTipBackdropColor()

When tool tips are enabled with `setToolTipDisplay(true)`, this method can be used to assign a backdrop (shadow) color to the tool tip.

Syntax:

```
void setToolTipBackdropColor(Color colorBackdrop)
```

where:

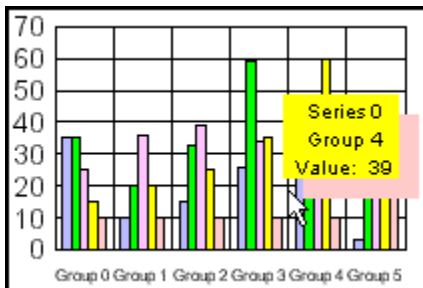
colorBackdrop

Is the tool tip backdrop color.

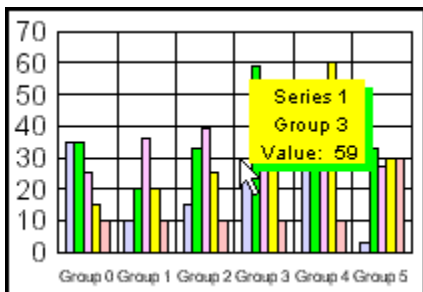
Return: *void*;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);  setToolTipBackdropColor(new Color(255,0,0));
```



```
setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);  setToolTipBackdropColor(new Color(0,255,0));
```



See [setToolTipBackdropSize\(\)](#) on page 1015.

setToolTipBackdropSize()

When tool tips are enabled with `setToolTipDisplay(true)`, this method can be used to define the size of a backdrop (shadow).

Syntax:

```
void setToolTipBackdropSize(int nBackdropSize)
```

where:

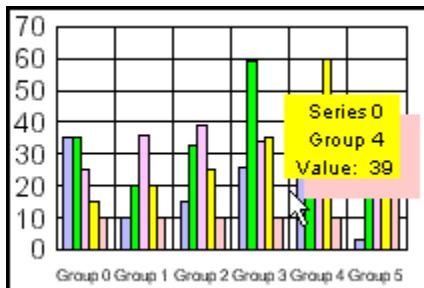
nBackdropSize

Is the tool tip backdrop size.

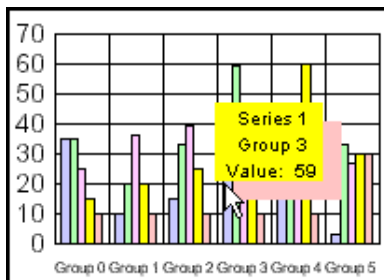
Return: *void*;

Example:

```
setDepthAngle(0); setDepthRadius(0);  
setToolTipDisplay(true); setToolTipBackdropColor(new  
Color(255,0,0));setToolTipBackdropSize(10);
```



```
setDepthAngle(0); setDepthRadius(0);  
setToolTipDisplay(true); setToolTipBackdropColor(new  
Color(255,0,0));setToolTipBackdropSize(20);
```



See [setToolTipBackdropColor\(\)](#) on page 1014.

setToolTipBackgroundColor()

This method can be used to define the color of background behind the tool tip text.

Syntax:

```
void setToolTipBackgroundColor(Color colorBackground)
```

where:

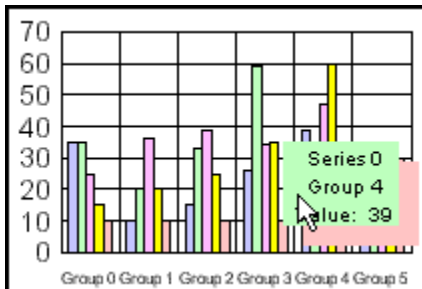
colorBackground

Is the tool tip background color.

Return: void;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true); setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(255,0,0));
setToolTipBackdropSize(10);
```



See [setToolTipBackdropColor\(\)](#) on page 1014, [setToolTipBackdropSize\(\)](#) on page 1015, [setToolTipColor\(\)](#) on page 1016.

setToolTipColor()

This method can be used to define the color of tool tip text.

Syntax:

```
void setToolTipColor(Color colorToolTip)
```

where:

colorToolTip

Is the tool tip text color.

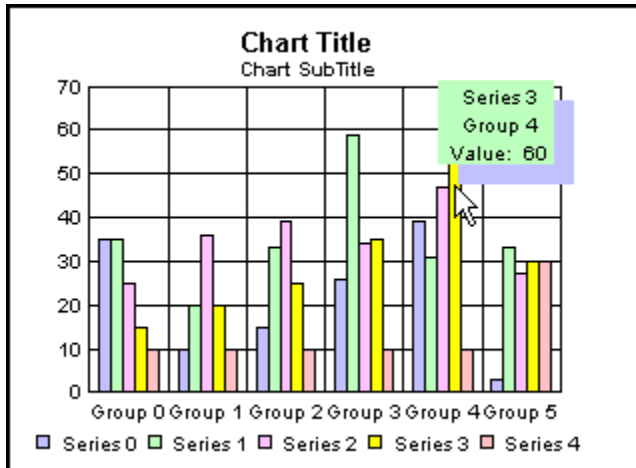
Return: void;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);
setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(0,0,255));  setToolTipColor(new
Color(255,0,0));
setToolTipBackdropSize(10);

```



See [setToolTipBackdropColor\(\)](#) on page 1014, [setToolTipBackdropSize\(\)](#) on page 1015, [setToolTipBackgroundColor\(\)](#) on page 1016, [setToolTipFontName\(\)](#) on page 1017, [setToolTipFontSize\(\)](#) on page 1019, [setToolTipFontStyle\(\)](#) on page 1020.

setToolTipFontName()

This method assigns the font type value for a tool tip.

Syntax:

```
void setToolTipFontName(String strFontName)
```

where:

strFontName

Is the name of the font (for example, Arial, Courier New, Times New Roman).

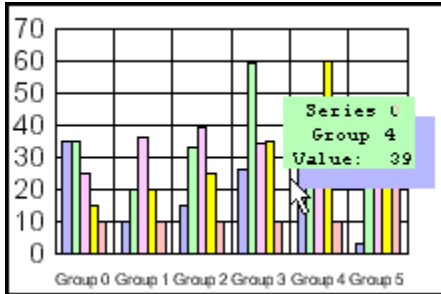
Return: void;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);
setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(0,0,255));
setToolTipColor(new Color(255,0,0));
setToolTipBackdropSize(10);  setToolTipFontName("Courier New");

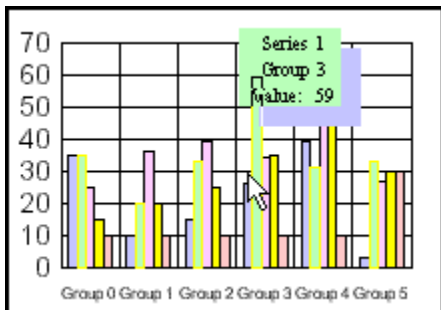
```



```

setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);
setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(0,0,255));
setToolTipColor(new Color(255,0,0));
setToolTipBackdropSize(10);  setToolTipFontName("Times New Roman");

```



See [setToolTipColor\(\)](#) on page 1016, [setToolTipFontSize\(\)](#) on page 1019, [setToolTipFontStyle\(\)](#) on page 1020.

setToolTipFontSize()

This method assigns a font size (in points) to tool tip strings.

Syntax:

```
void setToolTipFontSize(int nFontSize)
```

where:

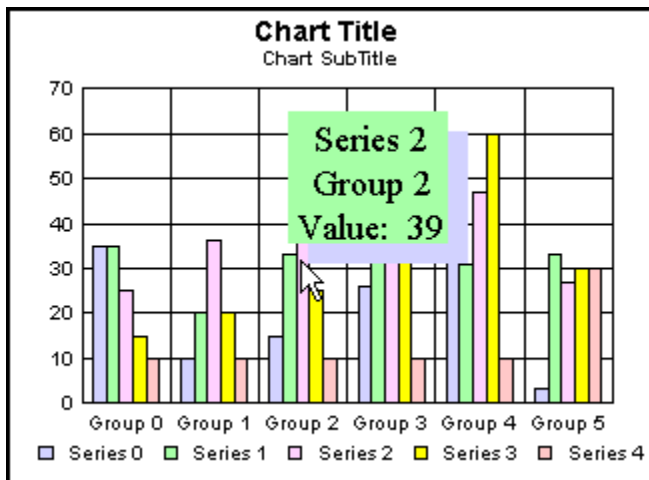
nFontSize

Is the font size in points.

Return: void;

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);
setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(0,0,255));
setToolTipColor(new Color(255,0,0));
setToolTipBackdropSize(10);
setToolTipFontName("Times New Roman"); setToolTipFontSize(18);
setToolTipDelay(1);
```



See [setToolTipColor\(\)](#) on page 1016, [setToolTipFontName\(\)](#) on page 1017, [setToolTipFontStyle\(\)](#) on page 1020.

setToolTipFontStyle()

This method assigns a font style to tool tip strings (for example, bold, italic, underline).

Syntax:

```
void setToolTipFontStyle(int nFontStyle)
```

where:

nFontStyle

Is the font style. Values can be:

- ☐ 0 = Plain
- ☐ 1 = *Italic*
- ☐ 2 = **Bold**
- ☐ 3 = ***Italic/Bold***
- ☐ 4 = Underline
- ☐ 5 = *Italic/Underline*
- ☐ 6 = **Bold/Underline** .
- ☐ 7 = ***Bold/Italic/Underline***

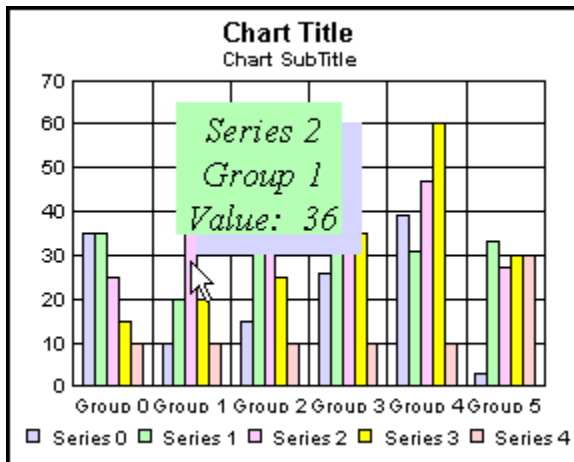
Return: *void*;

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setToolTipDisplay(true);
setToolTipBackgroundColor(new Color (0,255,0));
setToolTipBackdropColor(new Color(0,0,255));
setToolTipColor(new Color(255,0,0));
setToolTipBackdropSize(10);
setToolTipFontName("Times New Roman");
setToolTipFontSize(18); setToolTipFontStyle(2);
setToolTipDelay(1);

```



See [setToolTipColor\(\)](#) on page 1016, [setToolTipFontName\(\)](#) on page 1017, [setToolTipFontSize\(\)](#) on page 1019.

setToolTipOff()

This method turns off the generation of events for tool tips until a triggering event such as dwelling over some other object generates a new tool tip event.

Syntax:

```
void setToolTipOff();
```

Return: void;

setToolTipOn()

This method turns on the generation of events for tool tips.

Syntax:

```
void setToolTipOn();
```

Return: *void*;

See [setToolTipOff\(\)](#) on page 1021.

setTransparentBorderColor()

This method applies a transparent border color to an object.

Syntax:

```
void setTransparentBorderColor(boolean newValue);
```

```
void setTransparentBorderColor(IdentObj id,  
    boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method.

newValue

TRUE applies transparent border color.

FALSE draws the border in black (the default) or the specified color.

Return: *void*;

Note: If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [getTransparentBorderColor\(\)](#) on page 824, [getBorderColor\(\)](#) on page 723, [setBorderColor\(\)](#) on page 860, [setSeriesBorderColor\(\)](#) on page 976.

setTransparentFillColor()

This method applies a transparent fill color to an object.

Syntax:

```
void setTransparentFillColor(boolean newValue);
```

```
void setTransparentFillColor(IdentObj id,  
    boolean newValue);
```

where:

id

Is an object ID returned by a `getObjectID()` method.

newValue

Can be one of the following values:

true, applies transparent fill color to the object.

false, does not apply transparent fill color to the object.

Return: *void*;

Note: If an object ID (*id*) is not specified, *newValue* is assigned to the first item in the selection list (if applicable).

See [getTransparentFillColor\(\)](#) on page 825, [getFillColor\(\)](#) on page 746, [setFillColor\(\)](#) on page 890, [setSeriesFillColor\(\)](#) on page 977.

setUseReferenceLineXYRatio()

This method enables/disables the use of the `setReferenceLineXYRatio()` method.

Syntax:

```
void setUseReferenceLineXYRatio(IdentObj id, boolean);
```

where:

id

The *id* is `getReferenceLineXY()`.

boolean

Can be one of the following:

True enables the use of `setReferenceLineXYRatio()`.

False disables the use of `setReferenceLineXYRatio()`. This is the default value.

setUserToolTip()

When the `ToolTipDisplay` property is enabled (*true*) and the `ToolTipMode` property selects user information mode (0), the `setUserToolTip()` method can be used to define a custom tool tip string that will be displayed when the user positions the mouse pointer over a selectable object in a graph.

Syntax:

```
void setUserToolTip (String CustomToolTip);
```

where:

CustomToolTip

Is any string of characters which may or may not include the following macros:

- ❑ [ON] = Object Name (not supported with HTML5 graphs)
- ❑ [OID] = Object ID (not supported with HTML5 graphs)
- ❑ [R] = New Line (not supported with HTML5 graphs)

Note: If your browser is Firefox, the [R] macro adds a vertical bar (|) instead of a new line, as Firefox ignores the new line.

- ❑ [OIN] = Object Instance (not supported with HTML5 graphs)
- ❑ [OD] = Object Description (not supported with HTML5 graphs)
- ❑ [SL] = Series Label
- ❑ [GL] = Group Label
- ❑ [CUMSTKV] = Cumulative Stacked Value
- ❑ [CUMPCTV] = Cumulative Percentage Value
- ❑ [PIEPCTV] = Pie Percentage Value
- ❑ [HV] = Hilo High Value
- ❑ [LV] = Hilo Low Value
- ❑ [OV] = Hilo Open Value
- ❑ [CV] = Hilo Close Value
- ❑ [VV] = Hilo Volume Value
- ❑ [XV] = X Value
- ❑ [YV] = Y Value
- ❑ [ZV] = Z Value
- ❑ [PV] = Pareto Value

Example: [ON] is ([OID]) [R] Instance # [OIN]

Return: *void*;

See [ToolTipDisplay](#) on page 556, [ToolTipMode](#) on page 556.

setUserToolTipDefault()

The setUserToolTipDefault() method selects the default user tool tip strings that are provided with the application. When ToolTipMode selects developer information, the default tool tip string consists of the following: series number, group number, and value.

Example:

Series: 3Group: 4Value: 60.0

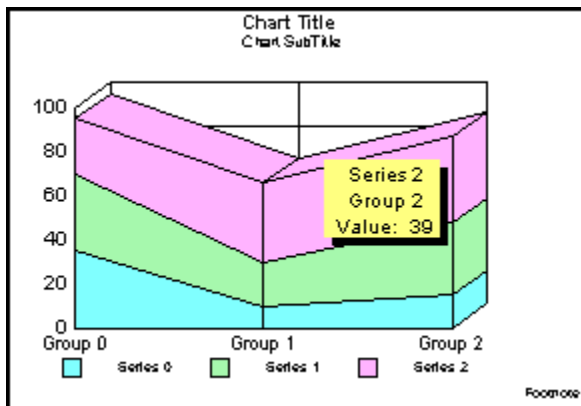
Syntax:

```
void setUserToolTipDefault( );
```

Return: *void*;

Example:

```
setGraphType(32);  
setToolTipDisplay(true);  setUserToolTipDefault();  
setDataRange(0,0,2,2);
```



See [ToolTipDisplay](#) on page 556, [ToolTipMode](#) on page 556.

setTextBackgroundColor()

This method can be used to enable (true) / disable (false), the background color for a text object that can be set by setTextBackgroundColor(). When disabled (false), the background color of a text object is ignored. When enabled (true), the background color of a specified text object is applied.

Syntax:

```
void setUseTextBackgroundColor(
    IdentObj idObj,
    boolean newValue);
```

where:

idObj

Is the object ID returned by a *getObjectID()* method (for example, *getTitle()*)

newValue

Can be one of the following values:

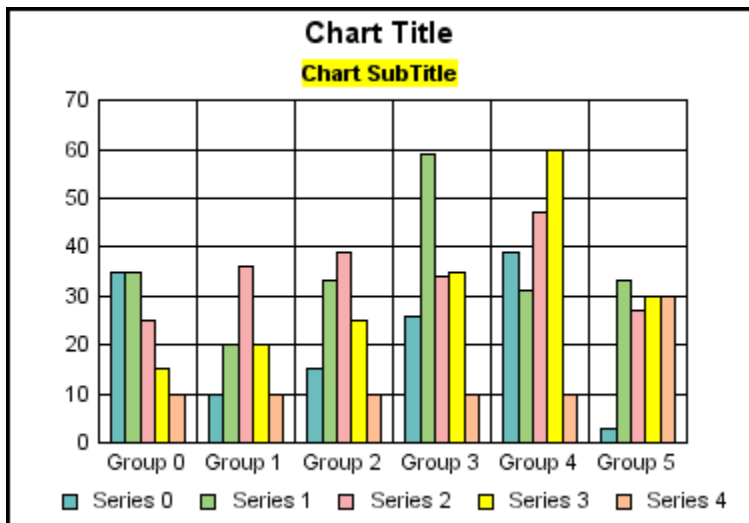
true applies background color

false does not apply background color

Return: *void*;

Example:

```
setUseTextBackgroundColor(getTitle(), false);
setUseTextBackgroundColor(getSubtitle(), true);
setTextBackgroundColor(getSubtitle(), new Color(255,255,0));
```



See [setTextBackgroundColor\(\)](#) on page 993.

setWaterfallGroupMode()

This assigns the waterfall mode (normal, subtotal, total or extra) to a specified group in a waterfall graph.

Syntax:

```
void setWaterfallGroupMode(int nGroup, int nGroupMode)
```

where:

nGroup

Is a group number.

nGroupMode

Is a group mode. Values can be:

- ☐ WATERFALLGROUPMODE_NORMAL = 0 Normal
- ☐ WATERFALLGROUPMODE_SUBTOTAL = 1 Subtotal
- ☐ WATERFALLGROUPMODE_TOTAL = 2 Total
- ☐ WATERFALLGROUPMODE_EXTRA = 3 Extra

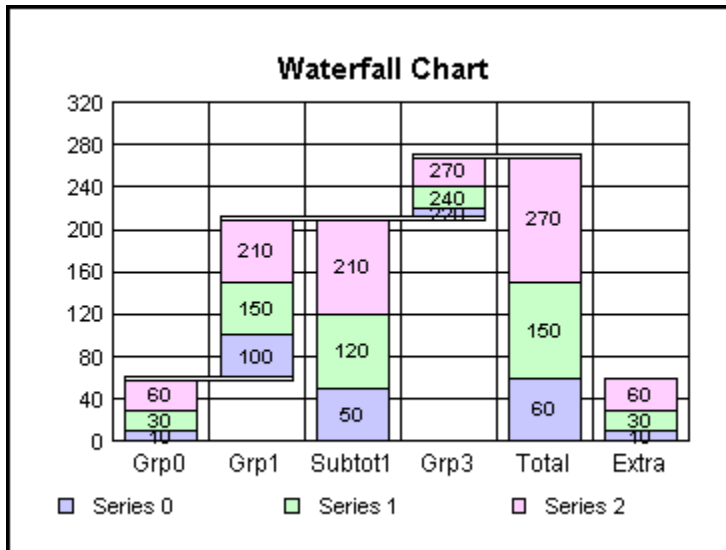
Return: *void;*

Example:

```
setGraphType(100);
setUseSampleData(false);
setDepthRadius(0);
setDepthAngle(0);
setData(0, 0, 10);
setData(1, 0, 20);
setData(2, 0, 30);
setGroupLabel(0, "Grp0");
setData(0, 1, 40);
setData(1, 1, 50);
setData(2, 1, 60);
setGroupLabel(1, "Grp1");
setData(0, 2, 0);
setData(1, 2, 0);
setData(2, 2, 0); setWaterfallGroupMode(2, 1);
setGroupLabel(getGroup(2), "Subtot1");
setData(0, 3, 10);
setData(1, 3, 20);
setData(2, 3, 30);
setGroupLabel(3, "Grp3");
setData(0, 4, 0);
setData(1, 4, 0);
setData(2, 4, 0); setWaterfallGroupMode(4, 2);
setGroupLabel(getGroup(4), "Total");
setData(0, 5, 10);
setData(1, 5, 20);
setData(2, 5, 30); setWaterfallGroupMode(5, 3);
setGroupLabel(getGroup(5), "Extra");
setStackedDataValueSum(true);
setDataTextDisplay(true);
setTextJustHoriz(getDataText(), 1);
setFontSizeAbsolute(getDataText(), true);
setFontSize(getDataText(), 10);
setFontSizeAbsolute(getO1Label(0), true);
setFontSize(getO1Label(0), 12);
setDataRangeToExtent();
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
```



```
setTitleString("Waterfall Chart");
```



Note: For a Normal group, series risers are stacked on top of each other and each successive series represents the value of its series/group plus any preceding series/group. For a subtotal or total group, the data for the group is ignored and the cumulative total up to this point will be shown for this group. Only one group in a graph can be designated as a total. An extra group is not used when performing any waterfall cumulative sums. It will appear just as stacked data. When used, it should be defined as the last group in the graph.

setX1()

This method defines the custom X1 coordinate based on the reference object.

Syntax:

```
void setX1(IdentObj IDobj, value);
```

where:

IDobj

Is the object ID of a reference object. The reference objects are `getReferenceLineXY()`, `getReferenceRect()`, and `getReferenceMarker()`.

value

Can be one of the following:

(0...1) when `setReferencePercentageMode()` is set to true.

The real value of the X-axis when `setReferencePercentageMode()` is set to false. This is the default value.

See [getReferenceLineXY\(\)](#) on page 799, [getReferenceRect\(\)](#) on page 801, [getReferenceMarker\(\)](#) on page 800.

setX2()

This method defines the custom X2 coordinate based on the reference object.

Syntax:

```
void setX2(IdentObj IDobj, value);
```

where:

IDobj

Is the object ID of a reference object. The reference objects are `getReferenceLineXY()` and `getReferenceRect()`.

value

Can be one of the following:

(0...1) when `setReferencePercentageMode()` is set to true.

The real value of the X-axis when `setReferencePercentageMode()` is set to false. This is the default value.

See [getReferenceLineXY\(\)](#) on page 799, [getReferenceRect\(\)](#) on page 801.

setY1()

This method defines the custom Y1 coordinate based on the reference object.

Syntax:

```
void setY1(IdentObj IDobj, value);
```

where:

IDobj

Is the object ID of a reference object. The reference objects are `getReferenceLineXY()`, `getReferenceRect()`, and `getReferenceMarker()`.

value

Can be one of the following:

(0...1) when `setReferencePercentageMode()` is set to true.

The real value of the Y-axis when `setReferencePercentageMode()` is set to false. This is the default value.

See [getReferenceLineXY\(\)](#) on page 799, [getReferenceRect\(\)](#) on page 801, [getReferenceMarker\(\)](#) on page 800.

setY2()

This method defines the custom Y2 coordinate based on the reference object.

Syntax:

```
void setY2(IdentObj IDobj, value);
```

where:

IDobj

Is the object ID of a reference object. The reference objects are `getReferenceLineXY()` and `getReferenceRect()`.

value

Can be one of the following:

(0...1) when `setReferencePercentageMode()` is set to true.

The real value of the Y-axis when `setReferencePercentageMode()` is set to false. This is the default value.

See [getReferenceLineXY\(\)](#) on page 799, [getReferenceRect\(\)](#) on page 801.

setZoomIndex()

This function sets the value of the current zoom state.

When data zooming is enabled with the zoom properties/methods or via the user interface (`setSelectionEnable(5)`), a zoom index is maintained that keeps track of the number of times a graph has been zoomed in and the state of zooming. As an example, assume `ZoomPercentage` is set to 10%, and the user clicks once. The graph is now zoomed to 90% of its original value, and the zoom index is now 1. Now the user clicks a second time. The graph is now zoomed to 80% of its original value, and the zoom index now is 2. Or, actually, the zoom index now has two entries: one for the first, 90% state, and a second for the second, 80% state.

You can keep adding to the zoom index as the user continues to click. Continuing with the example and the same values, assume the user has clicked eight consecutive times. The zoom index now contains eight items:

1. First Level: Zoom Factor 90%
2. Second Level: Zoom Factor 80%
3. Third Level: Zoom Factor 70%
4. Fourth Level: Zoom Factor 60%
5. Fifth Level: Zoom Factor 50%
6. Sixth Level: Zoom Factor 40%
7. Seventh Level: Zoom Factor 30%
8. Eighth Level: Zoom Factor 20%

Your application can go back and forth in this list. For example, assume you want to switch from the current level (the last level the user clicked, the eighth zoom level which currently shows 20% of the original state). You could use `setZoomIndex(4)` to immediately switch to the fourth level which had a zoom factor of 60%. You can use `setZoomIndex()` to switch back and forth between any zoom level set.

Syntax:

```
void setZoomIndex(int index);
```

where:

index

Is the new zoom index value.

Return: *void*;

Methods That Return an Object ID

The following topics describe methods that return an Object ID. The methods are listed in alphabetical order.

Object ID Methods

The following methods return an object ID.

A

[*getAllSeries\(\)*](#)

[*getAnnotationLabel\(\)*](#)

[*getAreaRiserBottom\(\)*](#)

[*getAnnotation\(\)*](#)

[*getAnnotationLine\(\)*](#)

[*getAreaRiserTop\(\)*](#)

[*getAnnotationBox\(\)*](#)

[*getAreaRiser\(\)*](#)

B

[*getBarRiser\(\)*](#)

[*getBarRiserBottom\(\)*](#)

[*getBarRiserSide\(\)*](#)

C

[*getChartBackground\(\)*](#)

[*getCubeLeftWallGridZ\(\)*](#)

[*getCubeRiserGridZ\(\)*](#)

[*getColorByHeight\(\)*](#)

[*getCubeRightWall\(\)*](#)

[*getCurveFitCorrelationText\(\)*](#)

[*getCubeFloor\(\)*](#)

[*getCubeRightWallGridX\(\)*](#)

[*getCurveFitCorrelationTextbox\(\)*](#)

[*getCubeFloorGridX\(\)*](#)

[*getCubeRightWallGridZ\(\)*](#)

[*getCurveFitEquationText\(\)*](#)

[*getCubeFloorGridZ\(\)*](#)

[*getCubeRiserFace\(\)*](#)

[*getCurveFitEquationTextbox\(\)*](#)

[*getCubeLeftWall\(\)*](#)

[*getCubeRiserGridX\(\)*](#)

[*getCurveFitLineRiser\(\)*](#)

[*getCubeLeftWallGridY\(\)*](#)

D

[*getDataText\(\)*](#)

E

getExceptionalRiser()

F

getFootnote()

getFrameSeparator()

getFunnelFeeler()

getFootnoteBox()

getFrameSide()

getFunnelLabel()

getFrame()

getFunnelCrust()

getFunnelValueLabel()

getFrameBottom()

G

getGaugeArea()

getGaugeMinorTick()

getGroupScroller()

getGaugeBackground()

getGaugeNeedle()

getGroupScrollerEditBox()

getGaugeBand1...5()

getGaugeScaleArea()

getGroupScrollerEditLeft()

getGaugeBand1...5Label()

getGaugeTitle()

getGroupScrollerEditRight()

getGaugeLabel()

getGlobal()

getGroupScrollerEditThumb()

getGaugeMajorTick()

getGroup()

getGroupScrollerThumb()

H

getHistogramRiser()

L

<i>getLegendArea()</i>	<i>getLegendText()</i>	<i>getLineRiser()</i>
<i>getLegendMarker()</i>	<i>getLegendTextBox()</i>	<i>getLineRiserSide()</i>
<i>getLegendScroller()</i>	<i>getLineMarker()</i>	<i>getLineRiserTop()</i>
<i>getLegendScrollerThumb()</i>		

N

<i>getNestedO1LabelO...8()</i>	<i>getNestedO1LabelLine()</i>	<i>getNonZeroBaseLineZeroLine()</i>
<i>getNestedO1LabelO...8Box()</i>		

O

<i>getO1Axis()</i>	<i>getO1MinorTick()</i>	<i>getO2Label()</i>
<i>getO1AxisLine()</i>	<i>getO1Title()</i>	<i>getO2MajorGrid()</i>
<i>getO1Label()</i>	<i>getO1TitleBox()</i>	<i>getO2MinorGrid()</i>
<i>getO1MajorGrid()</i>	<i>getO2Axis()</i>	<i>getO2Title()</i>
<i>getO1MajorTick()</i>		<i>getO2TitleBox()</i>
<i>getO1MinorGrid()</i>		

P

<i>getParetoLineMarker()</i>	<i>getPieBarConnectLineBottom()</i>	<i>getPieLabel()</i>
<i>getParetoLineRiser()</i>	<i>getPieBarConnectLines()</i>	<i>getPieRingLabel()</i>
<i>getParetoLineRiserSide()</i>	<i>getPieBarLabel()</i>	<i>getPieSliceFeelerPercent()</i>
<i>getParetoLineRiserSide1()</i>	<i>getPieBarRiserTop()</i>	<i>getPieSliceFeelerValue()</i>
<i>getParetoLineRiserSide2()</i>	<i>getPieBarSlice()</i>	<i>getPieSliceLabel()</i>
<i>getParetoLineRiserTop()</i>	<i>getPieBarSliceCrust()</i>	<i>getPlaceHorzO1LabelsTitle()</i>
<i>getParetoLineRiserTop1()</i>	<i>getPieFeelerLine()</i>	<i>getPlaceVertO1LabelsTitle()</i>
<i>getParetoLineRiserTop2()</i>	<i>getPieFrame()</i>	

Q

getQuadrantLine()

R

<i>getReferenceLine()</i>	<i>getReferenceLineY2()</i>
<i>getReferenceLineLegendText()</i>	<i>getReferenceLineText01()</i>
<i>getReferenceLine01()</i>	<i>getReferenceLineText02()</i>
<i>getReferenceLine02()</i>	<i>getReferenceLineTextX1()</i>
<i>getReferenceLineX1()</i>	<i>getReferenceLineTextY1()</i>
<i>getReferenceLineY1()</i>	<i>getReferenceLineTextY2()</i>

S

<i>getScatterQuadrantLine()</i>	<i>getSliceCrust()</i>	<i>getStockLowerRiser()</i>
<i>getSeries()</i>	<i>getSliceRing()</i>	<i>getStockLowLine()</i>
<i>getSeriesGroup()</i>	<i>getSpectralLegendMarker()</i>	<i>getStockMALine()</i>
<i>getSeriesScroller()</i>	<i>getSpectralMarker()</i>	<i>getStockOpenTick()</i>
<i>getSeriesScrollerEditBox()</i>	<i>getStackConnectLine()</i>	<i>getStockRiser()</i>
<i>getSeriesScrollerEditLeft()</i>	<i>getStock52WeekHighLine()</i>	<i>getStockRisingRiser()</i>
<i>getSeriesScrollerEditRight()</i>	<i>getStock52WeekLowLine()</i>	<i>getStockVolumeRiser()</i>
<i>getSeriesScrollerEditThumb()</i>	<i>getStockCloseTick()</i>	<i>getSubTitle()</i>
<i>getSeriesScrollerThumb()</i>	<i>getStockFallingRiser()</i>	<i>getSubTitleBox()</i>
<i>getSlice()</i>	<i>getStockHighLine()</i>	

T

<i>getThermGaugeArea()</i>	<i>getThermGaugeNeedle()</i>	<i>getTitle()</i>
<i>getThermGaugeBackground()</i>	<i>getTimeScaleLevel()</i>	<i>getTitleBox()</i>

W

getWaterFallLine()

X

<i>getX1Axis()</i>	<i>getX1MajorTick()</i>	<i>getX1Title()</i>
<i>getX1AxisLine()</i>	<i>getX1MinorGrid()</i>	<i>getX1TitleBox()</i>
<i>getX1Label()</i>	<i>getX1MinorTick()</i>	<i>getX1ZeroLine()</i>
<i>getX1MajorGrid()</i>		

Y

<i>getY1Axis()</i>	<i>getY2Title()</i>	<i>getY4MajorTick()</i>
<i>getY1AxisLine()</i>	<i>getY2TitleBox()</i>	<i>getY4MinorGrid()</i>
<i>getY1FrontZeroLine()</i>	<i>getY2ZeroLine()</i>	<i>getY4MinorTick()</i>
<i>getY1Label()</i>	<i>getY3Axis()</i>	<i>getY4Title()</i>
<i>getY1MajorGrid()</i>	<i>getY3AxisLine()</i>	<i>getY4TitleBox()</i>
<i>getY1MajorTick()</i>	<i>getY3FrontZeroLine()</i>	<i>getY4ZeroLine()</i>
<i>getY1MinorGrid()</i>	<i>getY3Label()</i>	<i>getY5Axis()</i>
<i>getY1MinorTick()</i>	<i>getY3MajorGrid()</i>	<i>getY5AxisLine()</i>
<i>getY1Title()</i>	<i>getY3MajorTick()</i>	<i>getY5Label()</i>
<i>getY1TitleBox()</i>	<i>getY3MinorGrid()</i>	<i>getY5MajorGrid()</i>
<i>getY1ZeroLine()</i>	<i>getY3MinorTick()</i>	<i>getY5MajorTick()</i>
<i>getY2Axis()</i>	<i>getY3Title()</i>	<i>getY5MinorGrid()</i>
<i>getY2AxisLine()</i>	<i>getY3TitleBox()</i>	<i>getY5MinorTick()</i>
<i>getY2FrontZeroLine()</i>	<i>getY3ZeroLine()</i>	<i>getY5Title()</i>
<i>getY2Label()</i>	<i>getY4Axis()</i>	<i>getY5TitleBox()</i>
<i>getY2MajorGrid()</i>	<i>getY4AxisLine()</i>	<i>getY5ZeroLine()</i>
<i>getY2MajorTick()</i>	<i>getY4Label()</i>	
<i>getY2MinorGrid()</i>	<i>getY4MajorGrid()</i>	
<i>getY2MinorTick()</i>		

getAllSeries()

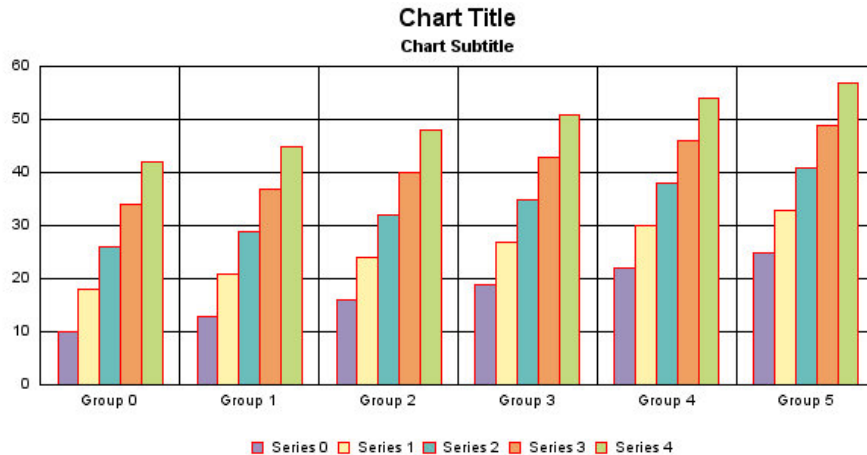
This method returns the object IDs of all series in the data matrix.

Syntax:

```
IdentObj getAllSeries();
```

Example:

```
setTransparentBorderColor(getAllSeries(),true);  
setBorderColor(getAllSeries(),new Color(255,0,0));
```



getAnnotation()

This method returns the object ID of an annotation object.

Syntax:

```
IdentObj getAnnotation(int index);
```

where:

index

Is the creational index of the Annotation.

getAnnotationBox()

This method returns the object ID of an annotation box object.

Syntax:

```
IdentObj getAnnotationBox(int index);
```

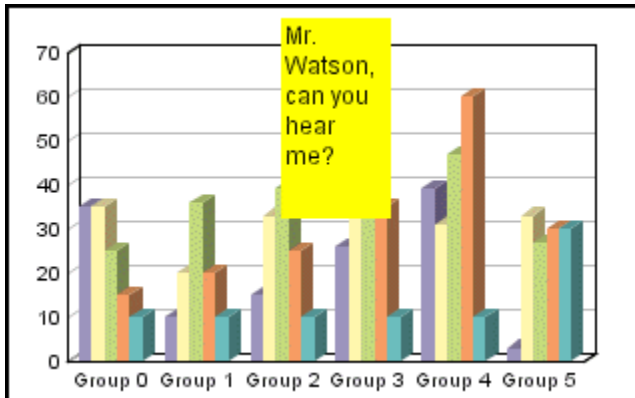
where:

index

Is the creational index of the annotation.

Example:

```
setDisplay(getAnnotation(0), true);
setTextString(getAnnotation(0), "Mr. Watson, can you hear me?");
setFillColor(getAnnotationBox(0), new Color (255,255,0));
setFillColor(getAnnotationLabel(0), new Color(0,0,0));
```



getAnnotationLabel()

This method returns the object ID of the specified annotation label object.

Syntax:

```
IdentObj getAnnotationLabel(int index);
```

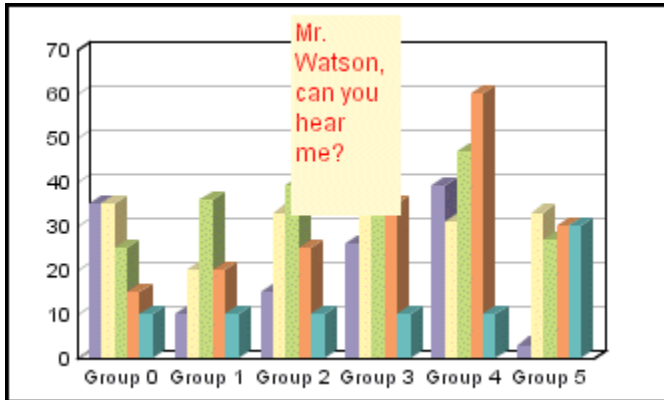
where:

index

Is the creational index of the Annotation.

Example:

```
setDisplay(getAnnotation(0), true);  
setTextString(getAnnotation(0), "Mr. Watson, can you hear me?");  
setFillColor(getAnnotationLabel(0), new Color (255,0,0));
```



For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getAnnotationLine()

This method returns the object ID of the specified annotation line object.

Syntax:

```
IdentObj getAnnotationLine(int index);
```

where:

index

Is the creational index of the Annotation.

getAreaRiser()

This method returns the object ID of the risers in an area graph. You may specify a series and group number to select a particular area riser.

Syntax:

```
IdentObj getAreaRiser();
```

```
IdentObj getAreaRiser(int s, int g);
```

where:

s

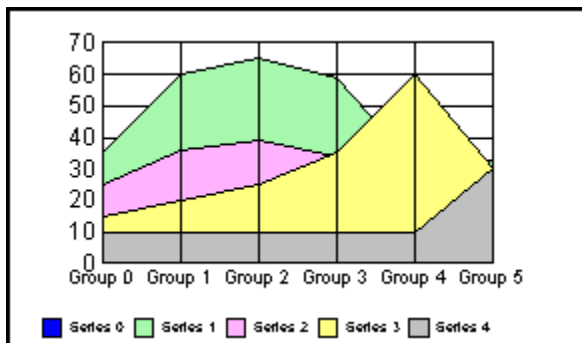
Is a series number.

g

Is a group number.

Example:

```
setDepthRadius(0);
setGraphType(31);
setDataValue(getAreaRiser(1,1),60);
setDataValue(getAreaRiser(1,2),65);
```



See [getAreaRiserBottom\(\)](#) on page 1041, [getAreaRiserTop\(\)](#) on page 1041.

getAreaRiserBottom()

This method returns the object ID of the bottom of an area riser.

Syntax:

```
IdentObj getAreaRiserBottom\(\);
```

See [getAreaRiser\(\)](#) on page 1040, [getAreaRiserTop\(\)](#) on page 1041.

getAreaRiserTop()

This method returns the object ID of the top of an area riser.

Syntax:

```
IdentObj getAreaRiserTop\(\);
```

See [getAreaRiser\(\)](#) on page 1040, [getAreaRiserBottom\(\)](#) on page 1041.

getBarRiser()

This method returns the object ID riser in a bar graph. You may specify a series and group number to select a particular bar riser.

Syntax:

```
IdentObj getBarRiser();
IdentObj getBarRiser(int s, int g);
```

where:

s

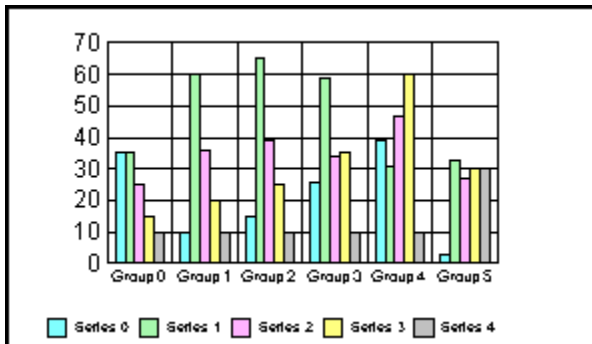
Is a series number.

g

Is a group number.

Example:

```
setDepthRadius(0);
setDataValue(
    getBarRiser(1,1),60);
setDataValue(
    getBarRiser(1,2),65);
```



See [getBarRiserBottom\(\)](#) on page 1042, [getBarRiserSide\(\)](#) on page 1043.

getBarRiserBottom()

This method returns the object ID of the bottom of a bar riser.

Syntax:

```
IdentObj getBarRiserBottom();
```

See [getBarRiser\(\)](#) on page 1042, [getBarRiserSide\(\)](#) on page 1043.

getBarRiserSide()

This method returns the object ID of the side of a bar riser.

Syntax:

```
IdentObj getBarRiserSide();
```

See [getBarRiser\(\)](#) on page 1042, [getBarRiserBottom\(\)](#) on page 1042.

getChartBackground()

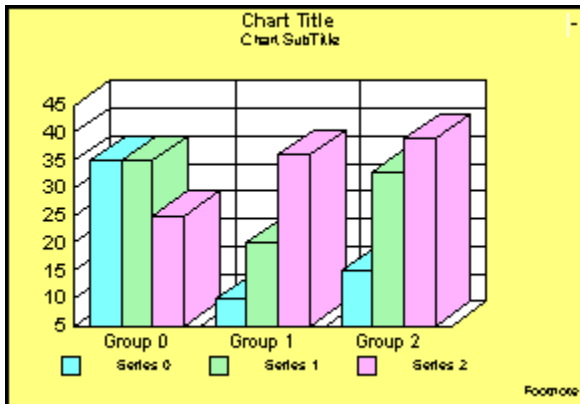
This method returns the object ID of the background area of a graph.

Syntax:

```
IdentObj getChartBackground();
```

Example:

```
setFillColor(getChartBackground(), new Color(255,255,0));
```



getColorByHeight()

When a graph is colored by height (rather than by series or by group), the graphing engine chooses the color of quantitative data representations based on the data values supplied to the graph. This happens in spectral map graphs and in three-dimensional surface graphs. If, for example, the data ranges from 7 to 74 and the Y1-axis scale, therefore, ranges from 0 to 80, the graphing engine might assign the colors blue to zero, red to 80, and various interpolated shades for values between zero and 80.

Note: Spectral Map graphs include a Y1-axis even though it is not visible in the graph.

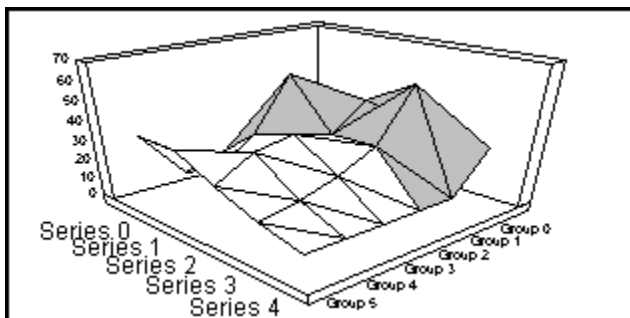
The `getColorByHeight()` method returns the `IdentObj` of the color by height gradient (in this case a blue to red one). With the object ID, your application can modify the gradient. Its end colors can be changed or the gradient could be made more complex. For example, in a spectral map of some topography, you could specify that the first half of the gradient representing ocean depths from -20,000 feet to zero go from dark blue to light blue. The second half of the gradient might show land heights from zero to 20,000 feet and could be represented by a gradient from green to brown.

Syntax:

```
IdentObj getColorByHeight();
```

Example:

```
setGraphType(12);
setGradientNumPins(getColorByHeight(),4);
setGradientPinLeftColor(
    getColorByHeight(),new Color(255,0,0),1);
setGradientPinPosition(getColorByHeight(),0.0,0);
setGradientPinPosition(getColorByHeight(),1.0,1);
setGradientPinRightColor(
    getColorByHeight(),new Color(0,0,255),0);
```



See [setGradientNumPins\(\)](#) on page 903, [setGradientPinLeftColor\(\)](#) on page 904, [setGradientPinPosition\(\)](#) on page 906, [setGradientPinRightColor\(\)](#) on page 908, [ExactColorByHeight](#) on page 352.

getCubeFloor()

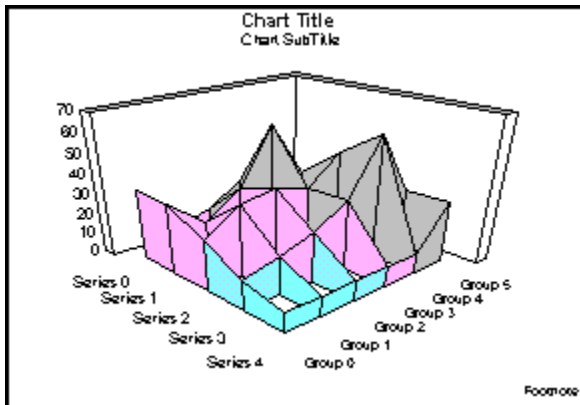
This method returns the object ID of the cube floor in a three-dimensional graph.

Syntax:

```
IdentObj getCubeFloor();
```

Example:

```
setGraphType(14);
setDisplay(getCubeFloor(), false);
```



See [getCubeFloorGridX\(\)](#) on page 1045, [getCubeFloorGridZ\(\)](#) on page 1046.

getCubeFloorGridX()

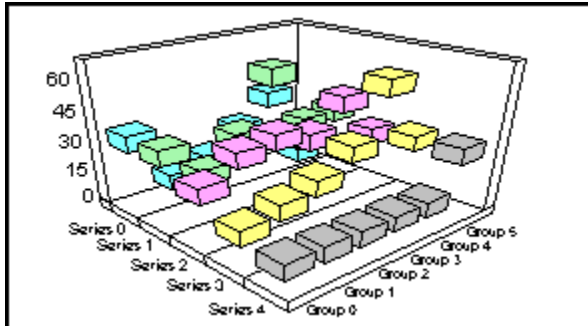
This method returns the object ID of X-axis grid lines that may appear on the cube floor in a three-dimensional graph.

Syntax:

```
IdentObj getCubeFloorGridX();
```

Example:

```
setGraphType(4);  
setDisplay(getCubeFloorGridX(),true);
```



See [getCubeFloor\(\)](#) on page 1044, [getCubeFloorGridZ\(\)](#) on page 1046.

getCubeFloorGridZ()

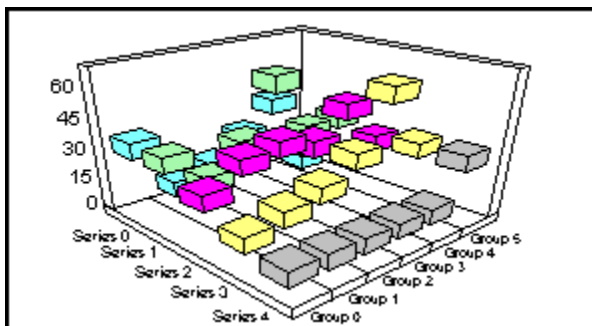
This method returns the object ID of Z-axis grid lines that may appear on the cube floor in a three-dimensional graph.

Syntax:

```
IdentObj getCubeFloorGridZ();
```

Example:

```
setGraphType(4);  
setLegendDisplay(false);  
setDisplay(getCubeFloorGridZ(),true);  
setAutoshading(false);
```



See [getCubeFloor\(\)](#) on page 1044, [getCubeFloorGridX\(\)](#) on page 1045

getCubeLeftWall()

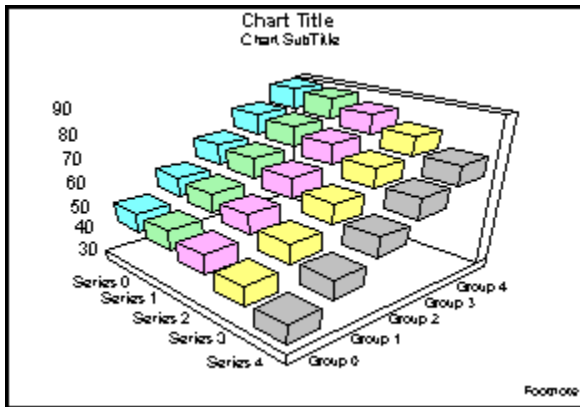
This method returns the object ID of the left wall of the cube in a three-dimensional graph.

Syntax:

```
IdentObj getCubeLeftWall();
```

Example:

```
setGraphType(4);  
setLegendDisplay(false);  
setDisplay(getCubeLeftWall(), false);
```



See [getCubeLeftWallGridY\(\)](#) on page 1047, [getCubeLeftWallGridZ\(\)](#) on page 1048.

getCubeLeftWallGridY()

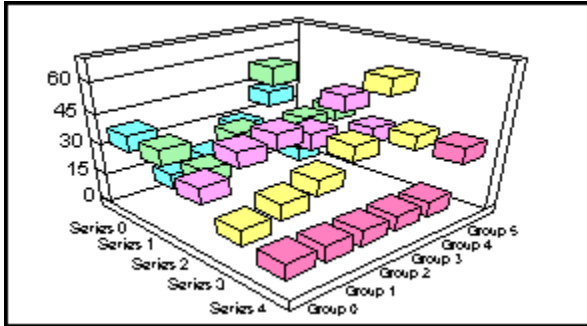
This method returns the object ID of Y-axis grid lines that may appear on the left wall in a three-dimensional graph.

Syntax:

```
IdentObj getCubeLeftWallGridY();
```

Example:

```
setGraphType(4);  
setLegendDisplay(false);  
setDisplay(getCubeLeftWallGridY(),true);  
setAutoshading(false);
```



See [getCubeLeftWall\(\)](#) on page 1047, [getCubeLeftWallGridZ\(\)](#) on page 1048.

getCubeLeftWallGridZ()

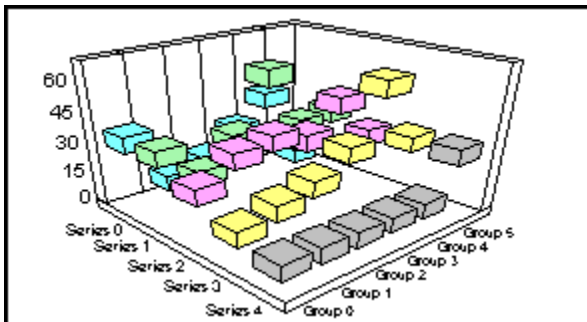
This method returns the object ID of Z-axis grid lines that may appear on the left wall in a three-dimensional graph.

Syntax:

```
IdentObj getCubeLeftWallGridZ();
```

Example:

```
setGraphType(4);  
setLegendDisplay(false);  
setDisplay(getCubeLeftWallGridZ(),true);  
setAutoshading(false);
```



See [getCubeLeftWall\(\)](#) on page 1047, [getCubeLeftWallGridY\(\)](#) on page 1047.

getCubeRightWall()

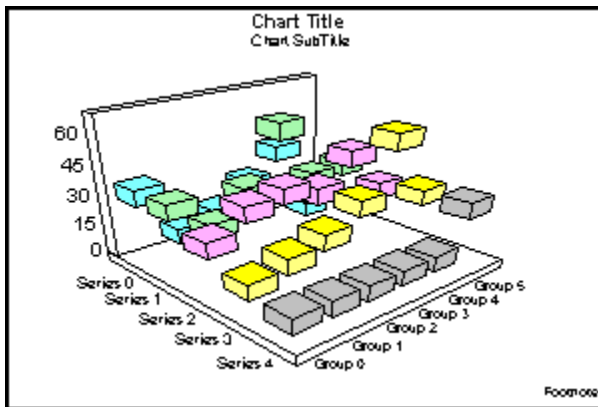
This method returns the object ID of the right wall of the cube in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRightWall();
```

Example:

```
setGraphType(4);  
setDisplay(getCubeRightWall(), false);
```



See [getCubeRightWallGridX\(\)](#) on page 1049, [getCubeRightWallGridZ\(\)](#) on page 1050.

getCubeRightWallGridX()

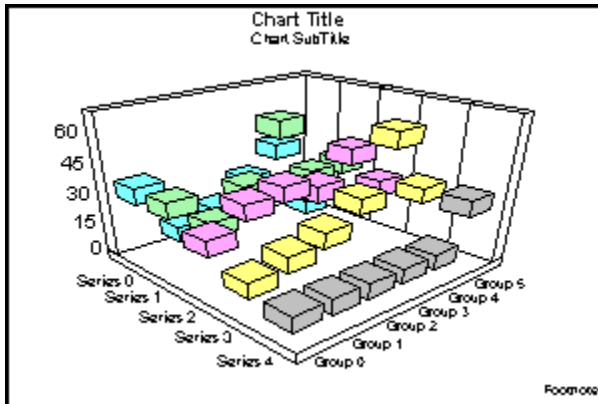
This method returns the object ID of X-axis grid lines that may appear on the cube right wall in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRightWallGridX();
```

Example:

```
setGraphType(4);  
setDisplay(getCubeRightWallGridX(),true);
```



See [getCubeRightWall\(\)](#) on page 1049, [getCubeRightWallGridZ\(\)](#) on page 1050

getCubeRightWallGridZ()

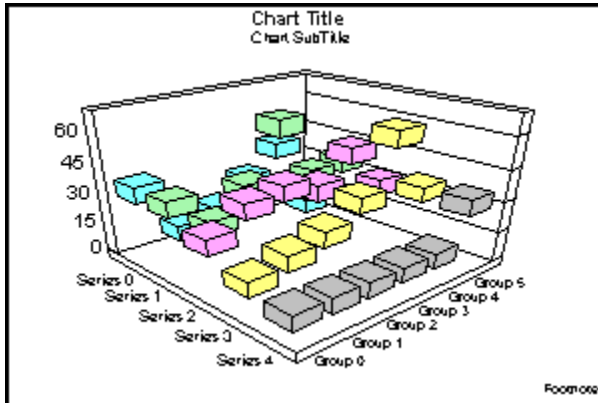
This method returns the object ID of Y-axis grid lines that may appear on the cube right wall in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRightWallGridY();
```

Example:

```
setGraphType(4);
setLegendDisplay(false);
setDisplay(getCubeRightWallGridY\(\),true);
```



See [getCubeRightWall\(\)](#) on page 1049, [getCubeRightWallGridX\(\)](#) on page 1049.

getCubeRiserFace()

This method returns the object ID of the face of risers in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRiserFace\(\);
```

See [getCubeRiserGridX\(\)](#) on page 1051, [getCubeRiserGridY\(\)](#) on page 1052, [getCubeRiserGridZ\(\)](#) on page 1053.

getCubeRiserGridX()

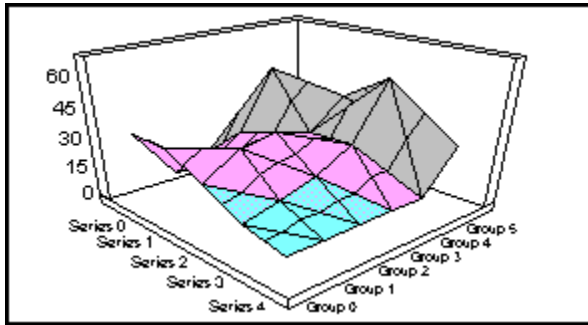
This method returns the object ID of X-axis grid lines that may appear on the risers in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRiserGridX\(\);
```

Example:

```
setGraphType(12);
setLegendDisplay(false);
setDisplay(getCubeRiserGridX(),true);
```



See [getCubeRiserFace\(\)](#) on page 1051, [getCubeRiserGridY\(\)](#) on page 1052, [getCubeRiserGridZ\(\)](#) on page 1053.

getCubeRiserGridY()

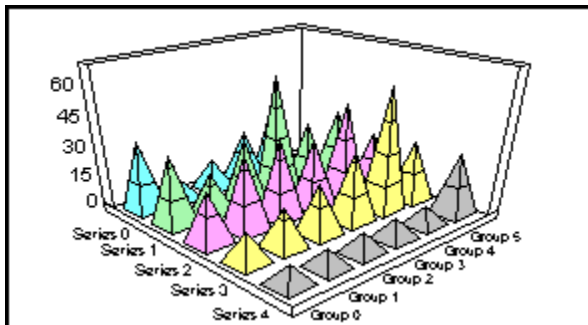
This method returns the object ID of Y-axis grid lines that may appear on the risers in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRiserGridY();
```

Example:

```
setGraphType(1);
setLegendDisplay(false);
setDisplay(getCubeRiserGridY(),true);
setAutoshading(false);
```



See [getCubeRiserFace\(\)](#) on page 1051, [getCubeRiserGridX\(\)](#) on page 1051, [getCubeRiserGridZ\(\)](#) on page 1053.

getCubeRiserGridZ()

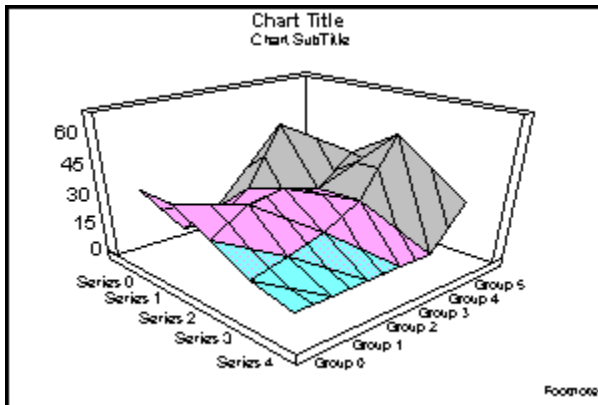
This method returns the object ID of Z-axis grid lines that may appear on the risers in a three-dimensional graph.

Syntax:

```
IdentObj getCubeRiserGridZ();
```

Example:

```
setGraphType(12);  
setLegendDisplay(false);  
setDisplay(getCubeRiserGridZ(), true);  
setAutoshading(false);
```



See [getCubeRiserFace\(\)](#) on page 1051, [getCubeRiserGridX\(\)](#) on page 1051, [getCubeRiserGridY\(\)](#) on page 1052.

getCurveFitCorrelationText()

The method returns the object ID of the curve fit correlation text. A curve fit line is drawn with the `setCurveFitType()` method.

Syntax:

```
IdentObj getCurveFitCorrelationText()
```

getCurveFitCorrelationTextbox()

This method returns the object ID of the text box behind the curve fit correlation text.

Syntax:

```
IdentObj getCurveFitCorrelationTextbox()
```

getCurveFitEquationText()

The method returns the object ID of the curve fit equation text. A curve fit line is drawn with the setCurveFitType() method. The CurveFitEquationDisplay property can be used to show/hide the equation text.

Syntax:

```
IdentObj getCurveFitEquationText();
IdentObj getCurveFitEquationText(int s);
```

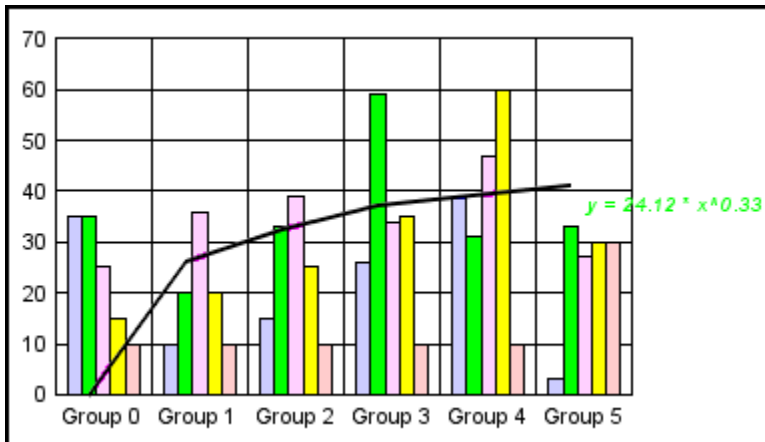
where:

s

Is an optional series number.

Example:

```
setDepthRadius(0);
setCurveFitType(1,11);
setFontStyle(getCurveFitEquationText(), 3);
setFillColor(getCurveFitEquationText(),new Color(0,255,0));
```



See [CurveFitEquationDisplay](#) on page 303, [setCurveFitType\(\)](#) on page 868.

getCurveFitEquationTextbox()

The method returns the object ID of the text box that surrounds the equation text when a curve fit line is shown with the setCurveFitType() method.

Syntax:

```
IdentObj getCurveFitEquationTextBox()
```

See [CurveFitEquationDisplay](#) on page 303, [getCurveFitCorrelationText\(\)](#) on page 1053, [getCurveFitType\(\)](#) on page 726, [setCurveFitType\(\)](#) on page 868.

getCurveFitLineRiser()

The method returns the object ID of the curve fit line that is drawn by the [setCurveFitType\(\)](#) method.

Syntax:

```
IdentObj getCurveFitLineRiser();
IdentObj getCurveFitLineRiser(int s);
```

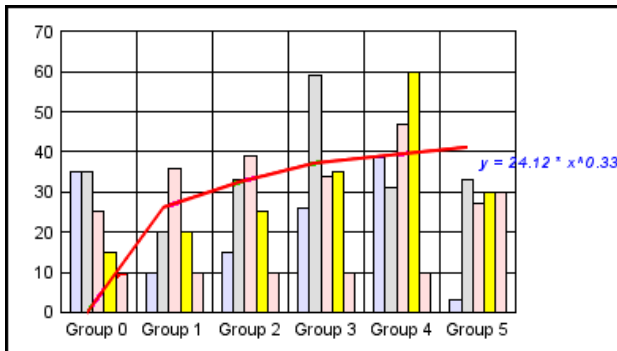
where:

s

Is an optional series number.

Example:

```
setDepthRadius(0);
setCurveFitType(1,11);
setFontStyle(getCurveFitEquationText(), 3);
setFillColor(getCurveFitEquationText(),
              new Color (0,0,255));
setBorderColor(getCurveFitLineRiser(1),
              new Color(255,0,0));
```



See [getCurveFitType\(\)](#) on page 726, [setCurveFitType\(\)](#) on page 868.

getDataText()

This method returns the object ID of a graph's data text.

Syntax:

```
IdentObj getDataText();
IdentObj getDataText(int s, int g);
```

where:

s

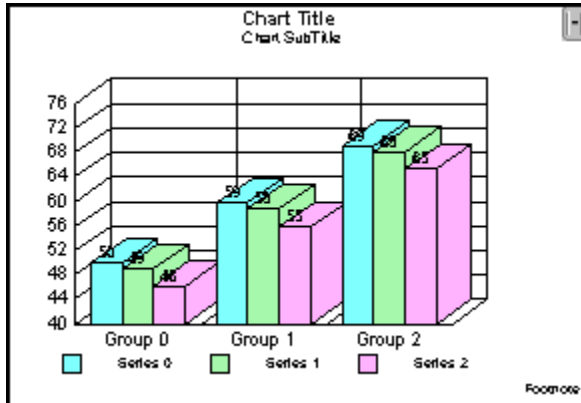
Is a series number.

g

Is a group number.

Example:

```
setDisplay(getDataText(), true);
setDataTextFormat(1);
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
```



See [getDataTextAngle\(\)](#) on page 733, [getDataTextRadius\(\)](#) on page 734.

getExceptionalRiser()

This method returns the object ID of an exceptional riser at series (*s*) and group (*g*).

Note: *s* and *g* are absolute series and group (not relative to the current viewable data range).

Syntax:

```
IdentObj getExceptionalRiser (int s, int g);
```

where:

s

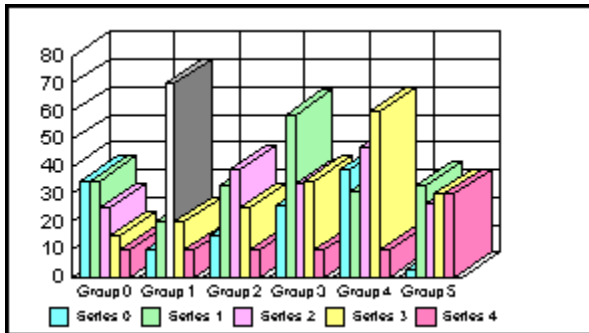
Is a series number.

g

Is a group number.

Example:

```
setExceptionalRiser(2,1);
setDataValue(getExceptionalRiser(2,1),70);
```



See [getExceptionalRiser\(\)](#) on page 1056, [setExceptionalRiser\(\)](#) on page 887, [setNoExceptionalRiser\(\)](#) on page 944.

getFootnote()

This method returns the object ID of the graph footnote object.

Syntax:

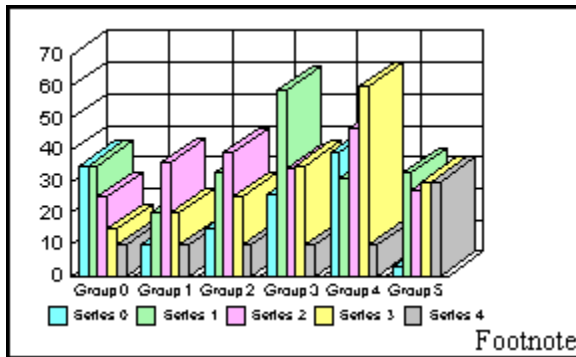
```
IdentObj getFootnote();
```

Example:

```

setAutofit(getFootnote(), false);
setFontName(getFootnote(), "TimesRoman");
setFontSizeVC(getFootnote(), 2318);

```

**getFootnoteBox()**

This method returns the object ID of the box that surrounds the graph footnote.

Syntax:

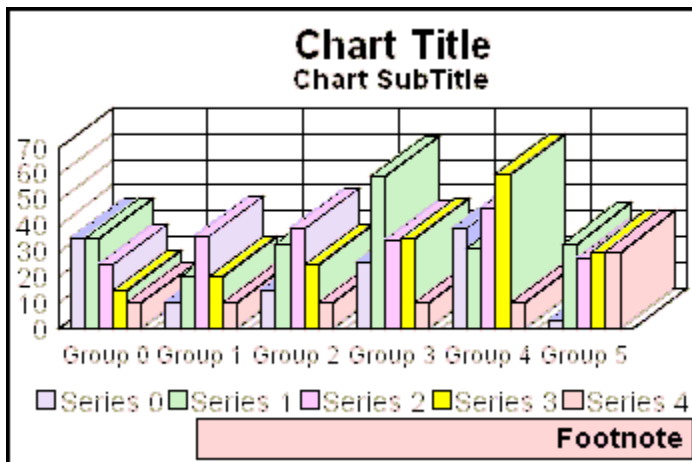
```
IdentObj getFootnoteBox();
```

Example:

```

setDisplay(getFootnoteBox(), true);
setFillColor(getFootnoteBox(), new Color(255,0,0));

```



getFrame()

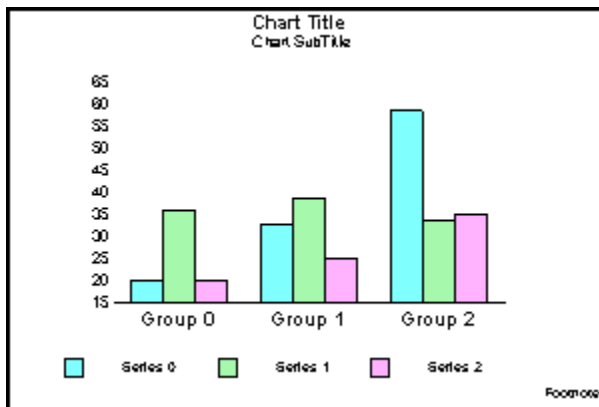
This method returns the object ID of the graph frame.

Syntax:

```
IdentObj getFrame();
```

Example:

```
setDisplay(getY1AxisLine(), false);
setDisplay(getY1MajorGrid(), false);
setDisplay(getO1MajorGrid(), false);
setDisplay(getFrame(), false);
setDataRange(1, 1, 3, 3);
```



getFrameBottom()

This method returns the object ID of the bottom of the graph frame.

Syntax:

```
IdentObj getFrameBottom();
```

See [getFrame\(\)](#) on page 1059, [getFrameSeparator\(\)](#) on page 1059, [getFrameSide\(\)](#) on page 1060.

getFrameSeparator()

This method returns the object ID of the line that separates the two halves of the graph frame in a dual-axis graph.

Syntax:

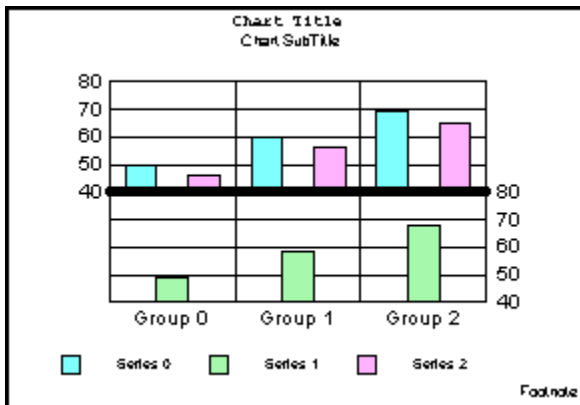
```
IdentObj getFrameSeparator();
```

Example:

```

setDataRange(1,1,3,3);
setViewableSeries(3);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setDataScrollerPresenceSeries(0);
setDepthAngle(0);
setDepthRadius(0);
setGraphType(21);
setLineWidth(getFrameSeparator\(\),8);

```



See [getFrame\(\)](#) on page 1059, [getFrameBottom\(\)](#) on page 1059, [getFrameSide\(\)](#) on page 1060.

getFrameSide()

This method returns the object ID of the graph frame side.

Syntax:

```
IdentObj getFrameSide\(\);
```

See [getFrame\(\)](#) on page 1059, [getFrameBottom\(\)](#) on page 1059, [getFrameSeparator\(\)](#) on page 1059.

getFunnelCrust()

This method returns the object ID of the crust of a funnel graph (GraphType 106).

Syntax:

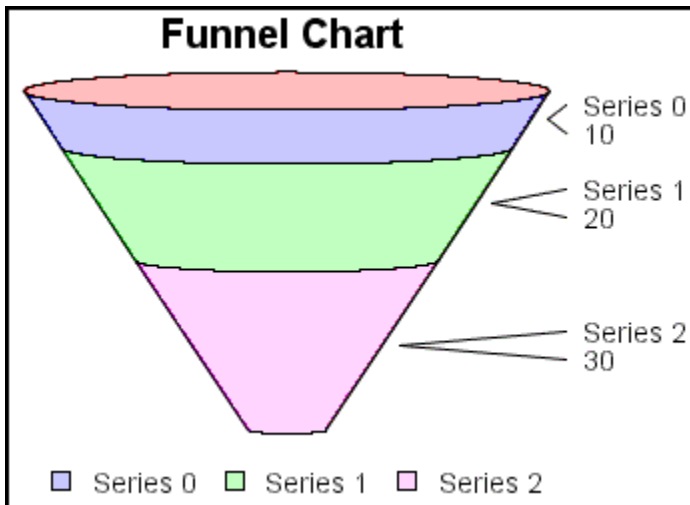
```
IdentObj getFunnelCrust\(\);
```


Example:

```

setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTitleString("Funnel Chart");
setTransparentBorderColor(getFrame(), true);
setPieTilt(25);
setFillColor(getFunnelCrust(), new Color(255,0,0));

```



See [getFunnelFeeler\(\)](#) on page 1061, [getFunnelLabel\(\)](#) on page 1062, [getFunnelValueLabel\(\)](#) on page 1063.

getFunnelFeeler()

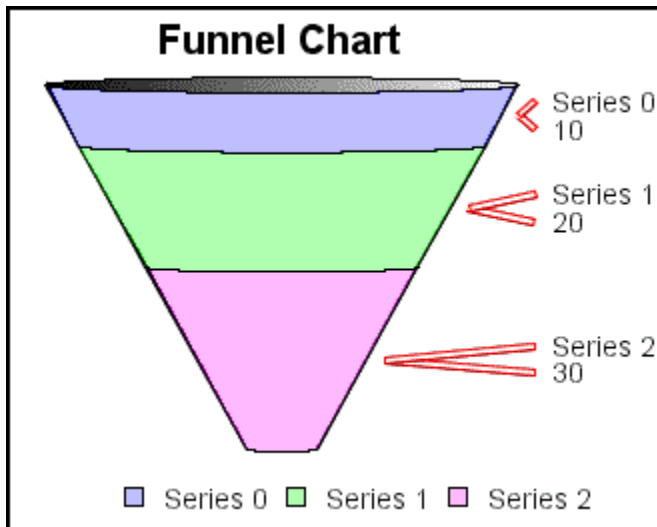
This method returns the object ID feelers in a funnel graph (GraphType 106).

Syntax:

```
IdentObj getFunnelFeeler();
```

Example:

```
setData(0,0,10);  
setData(1,0,20);  
setData(2,0,30);  
setSeriesLabel(0, "Series 0");  
setSeriesLabel(1, "Series 1");  
setSeriesLabel(2, "Series 2");  
setDisplayFunnelGroup(0);  
setDataRangeToExtent();  
setGraphType(106);  
setTitleString("Funnel Chart");  
setTransparentBorderColor(getFrame(), true);  
setLineWidth(getFunnelFeeler(), 3);  
setBorderColor(getFunnelFeeler(), new Color(255,0,0));
```



See [getFunnelCrust\(\)](#) on page 1060, [getFunnelLabel\(\)](#) on page 1062, [getFunnelValueLabel\(\)](#) on page 1063.

getFunnelLabel()

This method returns the object ID of labels in a funnel graph (GraphType 106).

Syntax:

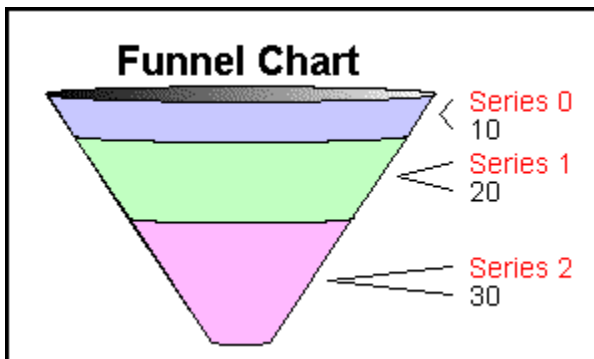
```
IdentObj getFunnelLabel();
```

Example:

```

setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTitleString("Funnel Chart");
setTransparentBorderColor(getFrame(), true);
setFillColor(getFunnelLabel(), new Color(255,0,0));

```



See [getFunnelCrust\(\)](#) on page 1060, [getFunnelFeeler\(\)](#) on page 1061, [getFunnelValueLabel\(\)](#) on page 1063.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getFunnelValueLabel()

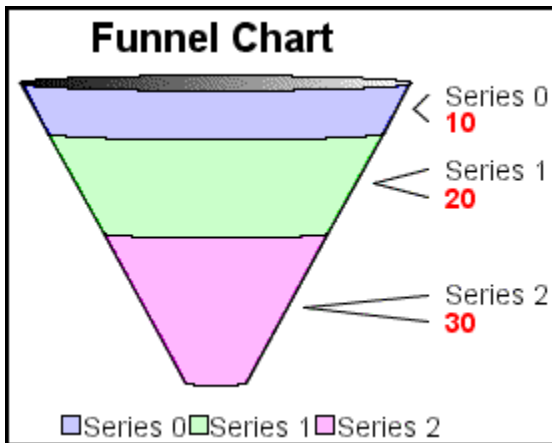
This method returns the object ID of value labels in a funnel graph (GraphType 106).

Syntax:

```
IdentObj getFunnelValueLabel();
```

Example:

```
setData(0,0,10);
setData(1,0,20);
setData(2,0,30);
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setDisplayFunnelGroup(0);
setDataRangeToExtent();
setGraphType(106);
setTitleString("Funnel Chart");
setTransparentBorderColor(getFrame(), true);
setFillColor(getFunnelValueLabel(), new Color(255,0,0));
setFontStyle(getFunnelValueLabel(), 2);
```



See [getFunnelCrust\(\)](#) on page 1060, [getFunnelFeeler\(\)](#) on page 1061, [getFunnelLabel\(\)](#) on page 1062.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getGaugeArea()

This method returns the object ID of the gauge area.

Syntax:

```
IdentObj getGaugeArea();
```

getGaugeBackground()

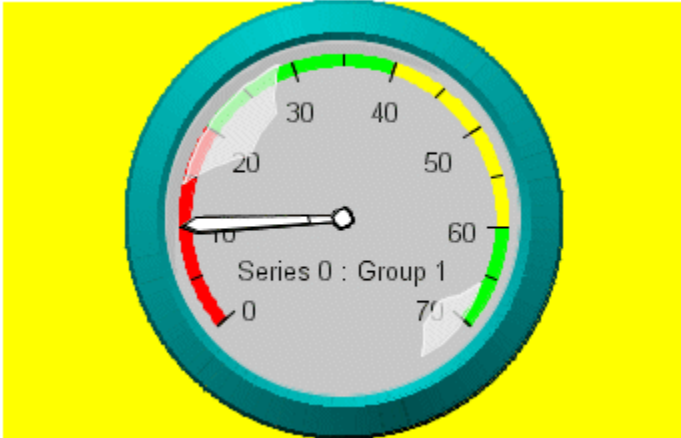
This method returns the object ID of the background in a gauge graph.

Syntax:

```
IdentObj getGaugeBackground();
```

Example:

```
setDisplay(getFrame(), false);
setFillColor(getGaugeBackground(), new Color(255,255,0));
```

**getGaugeBand1...5()**

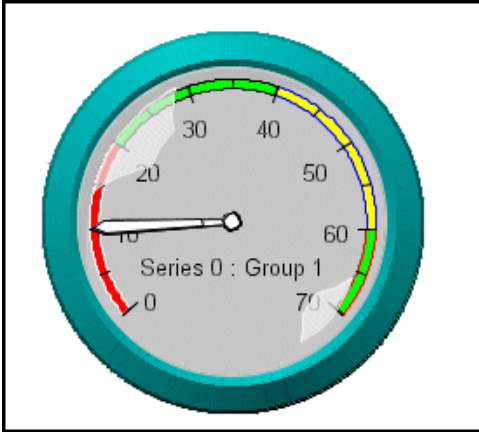
These methods return the object ID of the segments of the band in a gauge graph.

Syntax:

```
IdentObj getGaugeBand1();
IdentObj getGaugeBand2();
IdentObj getGaugeBand3();
IdentObj getGaugeBand4();
IdentObj getGaugeBand5();
```

Example:

```
setBorderColor(getGaugeBand1(), new Color(255, 255, 255));  
setBorderColor(getGaugeBand2(), new Color(0, 0, 0));  
setBorderColor(getGaugeBand3(), new Color(0, 0, 255));  
setBorderColor(getGaugeBand4(), new Color(255, 0, 0));  
setBorderColor(getGaugeBand5(), new Color(255, 255, 255));  
setDisplay(getFrame(), false);
```



getGaugeBand1...5Label()

These methods return the object ID of the labels that may appear next to the segments of the band in a gauge graph.

Syntax:

```
IdentObj getGaugeBand1Label();  
IdentObj getGaugeBand2Label();  
IdentObj getGaugeBand3Label();  
IdentObj getGaugeBand4Label();  
IdentObj getGaugeBand5Label();
```

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getGaugeLabel()

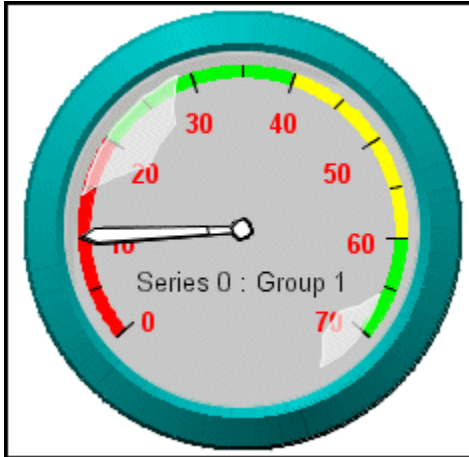
This method returns the object ID of the numeric value labels in a gauge graph.

Syntax:

```
IdentObj getGaugeLabel();
```

Example:

```
setDisplay(getFrame(), false);
setFillColor(getGaugeLabel(), new Color(255,0,0));
setFontStyle(getGaugeLabel(), 2);
```



For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getGaugeMajorTick()

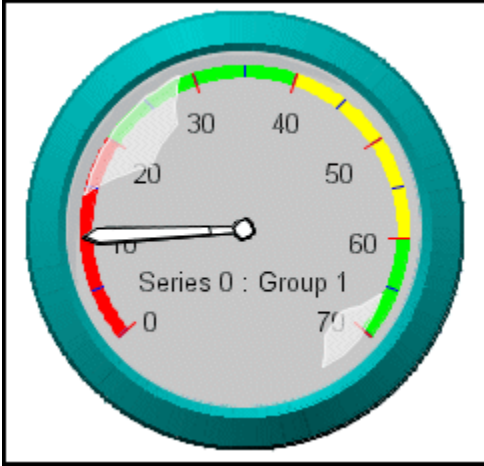
This method returns the object ID of the major tick marks in a gauge graph.

Syntax:

```
IdentObj getGaugeMajorTick();
```

Example:

```
setDisplay(getFrame(), false);  
setBorderColor(getGaugeMinorTick(), new Color(0,0,255));  
setBorderColor(getGaugeMajorTick(), new Color(255,0,0));
```



getGaugeMinorTick()

This method returns the object ID of the minor tick marks in a gauge graph.

Syntax:

```
IdentObj getGaugeMinorTick();
```

Example:

See [getGaugeMajorTick\(\)](#) on page 1067.

getGaugeNeedle()

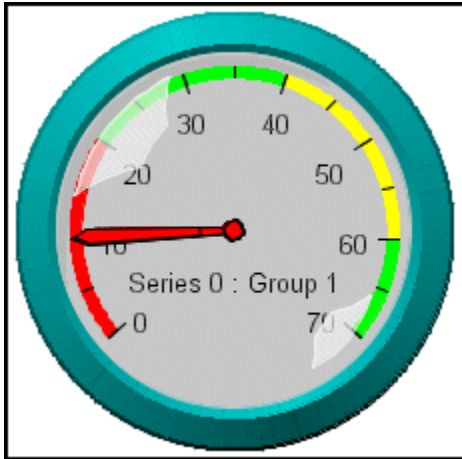
This method returns the object ID of the needle in a gauge graph.

Syntax:

```
IdentObj getGaugeNeedle();
```


Example:

```
setDisplay(getFrame(), false);
setFillColor(getGaugeNeedle(), new Color(255,0,0));
```

**getGaugeScaleArea()**

This method returns the object ID of the scale area in a gauge graph.

Syntax:

```
IdentObj getGaugeScaleArea();
```

getGaugeTitle()

This method returns the object ID of the title in a gauge graph.

Syntax:

```
IdentObj getGaugeTitle();
```

Example:

```
setDisplay(getFrame(), false);  
setFillColor(getGaugeTitle(), new Color(255,0,0));  
setFontStyle(getGaugeTitle(), 2);
```



getGlobal()

This method returns the global IdentObj, which can be passed to any method that takes an IdentObj (for example, getGradientDirection()). This method is useful for setting a global default values, for example, color and line width.

Note: This method is not completely implemented in this release of WebFOCUS Graphics.

Syntax:

```
IdentObj getGlobal();
```

getGroup()

This method returns the object ID of group number nGroup.

Syntax:

```
IdentObj getGroup(int nGroup);
```

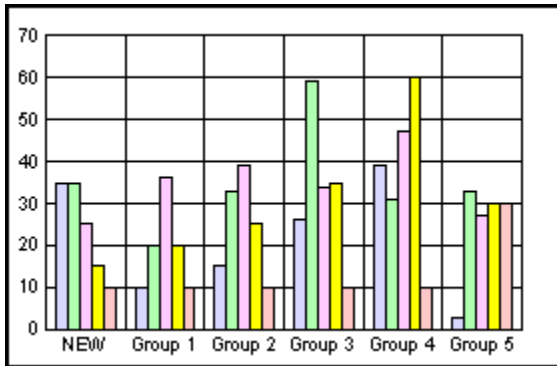
where:

nGroup

Is any valid group number.

Example:

```
setDepthRadius(0);
setGroupLabel(getGroup(0), "NEW");
```



See [getGroupLabel\(\)](#) on page 761, [getSeries\(\)](#) on page 1122, [getSeriesGroup\(\)](#) on page 1123.

getGroupScroller()

This method returns the object ID of the group data scrolling bar that can be enabled and displayed with the DataScrollerPresenceGroups property.

Syntax:

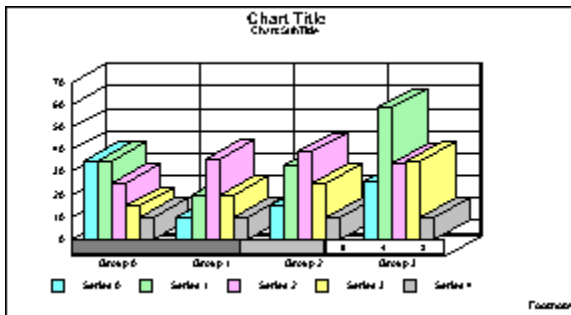
```
IdentObj getGroupScroller();
```

Example:

```

setAutofit(getGroupScrollerEditThumb(), false);
setAutofit(getGroupScrollerEditLeft(), false);
setAutofit(getGroupScrollerEditRight(), false);
setAxisTextAutofitMode(0);
setFontSizeVC(getGroupScrollerEditThumb(), 927);
setFontSizeVC(getGroupScrollerEditLeft(), 927);
setFontSizeVC(getGroupScrollerEditRight(), 927);
setGridStep(getY1MajorGrid(), 10.0);
setRect(getGroupScroller(),
        new Rectangle(-12351, -9382, 15852, 2927));
setScrollOffsetGroup(0);
setScrollOffsetSeries(0);
setViewableGroups(4);
setViewableSeries(0);

```



See [getGroupScrollerEditLeft\(\)](#) on page 1073, [getGroupScrollerEditBox\(\)](#) on page 1072, [getGroupScrollerEditRight\(\)](#) on page 1073, [getGroupScrollerEditThumb\(\)](#) on page 1074, [getGroupScrollerThumb\(\)](#) on page 1074, [DataScrollerPresenceGroups](#) on page 310.

getGroupScrollerEditBox()

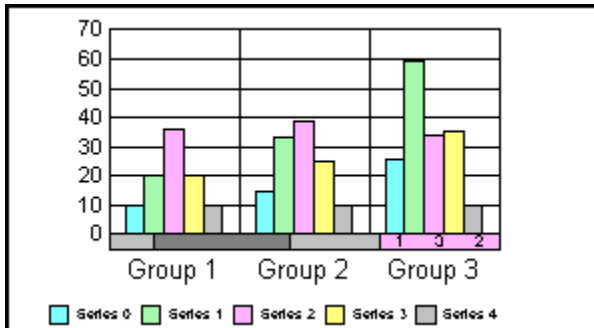
This method returns the object ID of the group scroller edit box. The group scroller bar is enabled and displayed with the `DataScrollerPresenceGroups` property. The edit box is the editable area of the scroll bar that shows three numbers representing the total groups, viewable groups, and group offset.

Syntax:

```
IdentObj getGroupScrollerEditBox();
```

Example:

```
setViewableGroups(3);
setScrollOffsetGroup(1);
setFillColor(getGroupScrollerEditBox\(\),
    new Color(206,0,121));
setDepthAngle(0);
setDepthRadius(0);
```



See [getGroupScroller\(\)](#) on page 1071, [getGroupScrollerEditLeft\(\)](#) on page 1073, [getGroupScrollerEditRight\(\)](#) on page 1073, [getGroupScrollerEditThumb\(\)](#) on page 1074, [getGroupScrollerThumb\(\)](#) on page 1074.

getGroupScrollerEditLeft()

This method returns the object ID of the group data scrolling left edit bar that can be enabled and displayed with the `DataScrollerPresenceGroups` property. This object is the left most number in the editable area of the scroll bar that shows three numbers representing the total groups, viewable groups, and group offset.

Syntax:

```
IdentObj getGroupScrollerEditLeft\(\);
```

See [getGroupScroller\(\)](#) on page 1071 for an example. Also see [getGroupScrollerEditBox\(\)](#) on page 1072, [getGroupScrollerEditRight\(\)](#) on page 1073, [getGroupScrollerEditThumb\(\)](#) on page 1074, [getGroupScrollerThumb\(\)](#) on page 1074.

getGroupScrollerEditRight()

This method returns the object ID of the group data scrolling right edit bar that can be enabled and displayed with the `DataScrollerPresenceGroups` property. This object is the right most number in the editable area of the scroll bar that shows three numbers representing the total groups, viewable groups, and group offset.

Syntax:

```
IdentObj getGroupScrollerEditRight();
```

See [getGroupScroller\(\)](#) on page 1071 for an example. Also see [getGroupScrollerEditLeft\(\)](#) on page 1073, [getGroupScrollerEditBox\(\)](#) on page 1072, [getGroupScrollerEditThumb\(\)](#) on page 1074, [getGroupScrollerThumb\(\)](#) on page 1074.

getGroupScrollerEditThumb()

This method returns the object ID of the group data scrolling bar edit thumb that can be enabled and displayed with the DataScrollerPresenceGroups property. The edit thumb is the center number in the editable area of the scroll bar that shows three numbers representing the total groups, viewable groups, and group offset.

Syntax:

```
IdentObj getGroupScrollerEditThumb();
```

See [getGroupScroller\(\)](#) on page 1071 for an example. Also see [getGroupScrollerEditLeft\(\)](#) on page 1073, [getGroupScrollerEditBox\(\)](#) on page 1072, [getGroupScrollerEditRight\(\)](#) on page 1073, [getGroupScrollerThumb\(\)](#) on page 1074.

getGroupScrollerThumb()

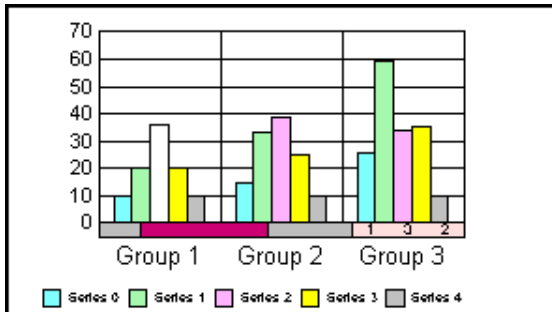
This method returns the object ID of the group scroller thumb. The group scroll bar is enabled and displayed with the DataScrollerPresenceGroups property. The thumb is the movable area of the scroll bar that can be used to adjust the group offset.

Syntax:

```
IdentObj getGroupScrollerThumb();
```

Example:

```
setViewableGroups(3);
setScrollOffsetGroup(1);
setFillColor(getGroupScrollerThumb\(\),
             new Color(206,0,121));
setDepthAngle(0);
setDepthRadius(0);
```



See [getGroupScroller\(\)](#) on page 1071, [getGroupScrollerEditLeft\(\)](#) on page 1073, [getGroupScrollerEditBox\(\)](#) on page 1072, [getGroupScrollerEditRight\(\)](#) on page 1073, [getGroupScrollerEditThumb\(\)](#) on page 1074, [DataScrollerPresenceGroups](#) on page 310.

getHistogramRiser()

This method returns the object ID of the risers in a histogram graph.

Syntax:

```
IdentObj getHistogramRiser();
```

Example:

```
setFillColor(getHistogramRiser\(\), new Color(0,174,227));
setGraphType(85);
```

getLegendArea()

This method returns the object ID of the graph's legend area.

Syntax:

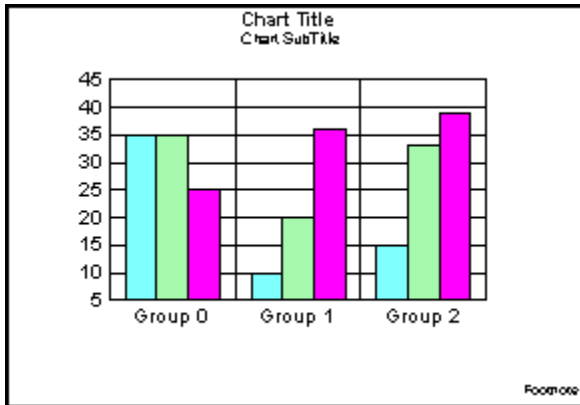
```
IdentObj getLegendArea();
```

Example:

```

setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setDisplay(getLegendArea(), false);
setDepthRadius(0);

```

**getLegendMarker()**

This method returns the object ID of markers in the graph's legend area. Use the input parameters to select the legend marker for a particular series.

Syntax:

```

IdentObj getLegendMarker();
IdentObj getLegendMarker(int s);

```

where:

s

Is a series number.

See [getLegendArea\(\)](#) on page 1075, [getLegendText\(\)](#) on page 1078, [getLegendScroller\(\)](#) on page 1076, [getLegendScrollerThumb\(\)](#) on page 1077, [getLegendText\(\)](#) on page 1078, [getLegendTextBox\(\)](#) on page 1079.

getLegendScroller()

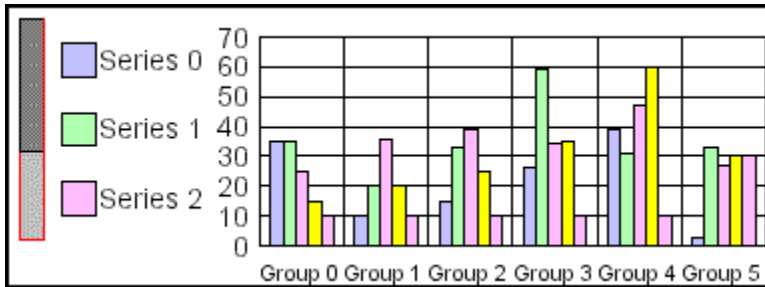
This method returns the object ID of the scroll bar in the legend area. The legend scroll bar must be enabled with the LegendScroll property.

Syntax:

```
IdentObj getLegendScroller();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(true);
setLegendScroll(2);
setLegendSeriesCount(3);
setLegendSeriesStart(0);
setLegendOrient(0);
setLegendRect(new Rectangle(-15340, -2478, 5948, 17550));
setBorderColor(getLegendScroller(), new Color(255,0,0));
```



See [getLegendArea\(\)](#) on page 1075, [getLegendMarker\(\)](#) on page 1076, [getLegendScrollerThumb\(\)](#) on page 1077, [getLegendText\(\)](#) on page 1078, [getLegendTextBox\(\)](#) on page 1079.

getLegendScrollerThumb()

This method returns the object ID of the legend scroll bar thumb. The legend scroll bar must be enabled with the LegendScroll property.

Syntax:

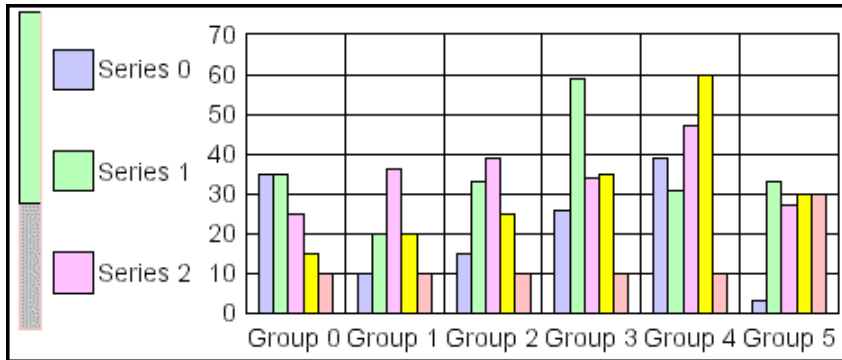
```
IdentObj getLegendScrollerThumb();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setSubtitleDisplay(false);
setTitleDisplay(false);
setLegendAutomatic(true);
setLegendScroll(2);
setLegendSeriesCount(3);
setLegendSeriesStart(0);
setLegendOrient(0);
setLegendRect(new Rectangle(-15340,-2478,5948,17550));
setBorderColor(getLegendScroller(), new Color(255,0,0));
setFillColor(getLegendScrollerThumb(),
              new Color(0,255,0));

```



See [getLegendArea\(\)](#) on page 1075, [getLegendMarker\(\)](#) on page 1076, [getLegendScroller\(\)](#) on page 1076, [getLegendText\(\)](#) on page 1078, [getLegendArea\(\)](#) on page 1075.

getLegendText()

This method returns the object ID of text in the graph's legend text.

Syntax:

```

IdentObj getLegendText();
IdentObj getLegendText (int s);

```

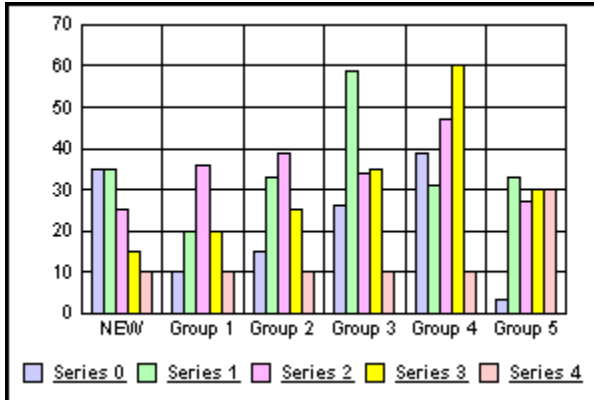
where:

s

Is a series number.

Example:

```
setDepthRadius(0);
setFontStyle(getLegendText(),6);
```



See [getLegendArea\(\)](#) on page 1075, [getLegendMarker\(\)](#) on page 1076, [getLegendScroller\(\)](#) on page 1076, [getLegendScrollerThumb\(\)](#) on page 1077 [getLegendTextBox\(\)](#) on page 1079.

getLegendTextBox()

This method returns the object ID of the textbox behind the graph's legend text.

Syntax:

```
IdentObj getLegendTextBox();
```

See [getLegendArea\(\)](#) on page 1075, [getLegendText\(\)](#) on page 1078.

getLineMarker()

This method returns the object ID of markers in a line graph. Use the optional series numbers parameter to select a particular series of markers in a line graph.

Syntax:

```
IdentObj getLineMarker();
IdentObj getLineMarker (int s, int g);
```

where:

s

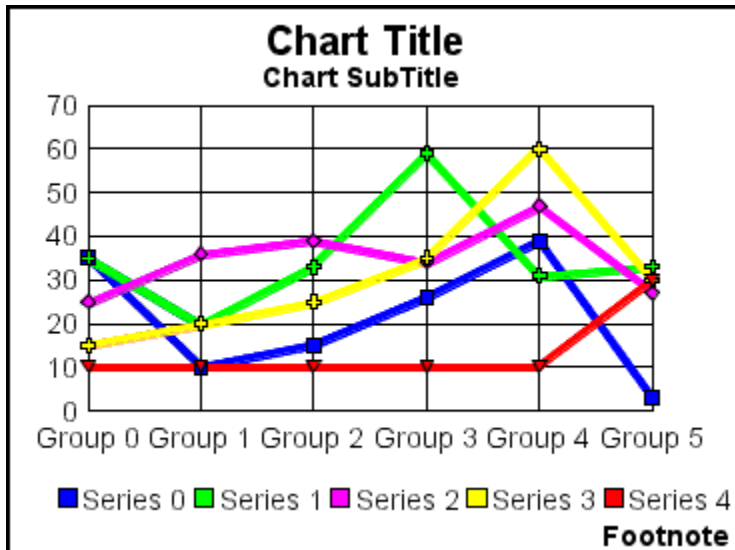
Is an optional series number.

g

Is an optional group number.

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setMarkerShape(getLineMarker(1,3), 4);
```



See [getLineRiser\(\)](#) on page 1080, [getLineRiserSide\(\)](#) on page 1081, [getLineRiserTop\(\)](#) on page 1081.

getLineRiser()

This method returns the object ID of risers in a line graph. Use the optional series numbers parameter to get the object ID of a riser for a particular series in a line graph.

Syntax:

```
IdentObj getLineRiser();
IdentObj getLineRiser (int s, int g);
```

where:*s*

Is an optional series number.

g

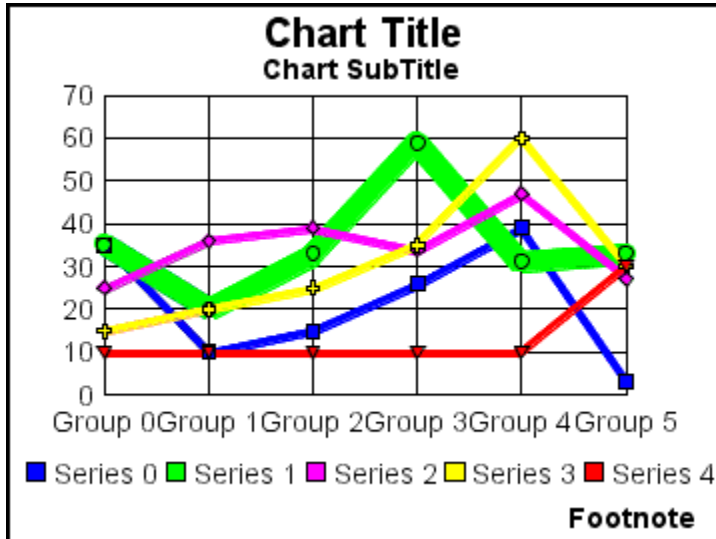
Is an optional group number.

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setGraphType(41);
setLineWidth(getLineRiser(1,3), 10);

```



See [getLineMarker\(\)](#) on page 1079, [getLineRiserSide\(\)](#) on page 1081, [getLineRiserTop\(\)](#) on page 1081.

getLineRiserSide()

This method returns the object ID of the side of risers in a line graph.

Syntax:

```
IdentObj getLineRiserSide();
```

See [getLineMarker\(\)](#) on page 1079, [getLineRiser\(\)](#) on page 1080, [getLineRiserTop\(\)](#) on page 1081.

getLineRiserTop()

This method returns the object ID of the top of risers in a line graph.

Syntax:

```
IdentObj getLineRiserTop();
```

See [getLineMarker\(\)](#) on page 1079, [getLineRiser\(\)](#) on page 1080, [getLineRiserSide\(\)](#) on page 1081.

getNestedO1Label0...8()

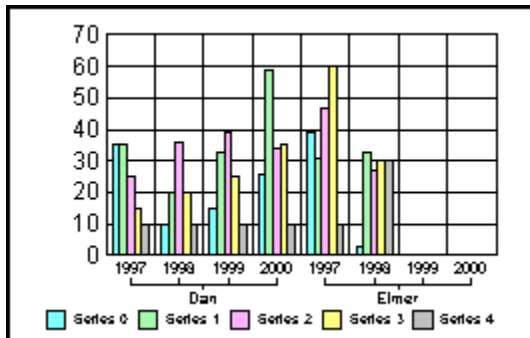
These methods return the object ID of each of the nested labels (0...8) that can be enabled with the NestedLabels property.

Syntax:

```
IdentObj getNestedO1Label0();
IdentObj getNestedO1Label1();
IdentObj getNestedO1Label2();
IdentObj getNestedO1Label3();
IdentObj getNestedO1Label4();
IdentObj getNestedO1Label5();
IdentObj getNestedO1Label6();
IdentObj getNestedO1Label7();
IdentObj getNestedO1Label8();
```

Example:

```
setDepthRadius(0);
setDepthAngle(0);
setNestedLabels(true);
setDisplay(getNestedO1Labels0(), true);
```



See [NestedLabels](#) on page 415, [getNestedO1Label0...8Box\(\)](#) on page 1082, [getNestedO1LabelLine\(\)](#) on page 1083.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getNestedO1Label0...8Box()

These methods return the object ID of each of the text boxes behind nested labels (0...8) that can be enabled with the NestedLabels property.

Syntax:

```

IdentObj getNested01Label0Box();
IdentObj getNested01Label1Box();
IdentObj getNested01Label2Box();
IdentObj getNested01Label3Box();
IdentObj getNested01Label4Box();
IdentObj getNested01Label5Box();
IdentObj getNested01Label6Box();
IdentObj getNested01Label7Box();
IdentObj getNested01Label8Box();

```

See [NestedLabels](#) on page 415, [getNested01Label0...8\(\)](#) on page 1082, [getNested01LabelLine\(\)](#) on page 1083.

getNested01LabelLine()

This method gets the object ID of the line that can be drawn below group axis nested labels.

Syntax:

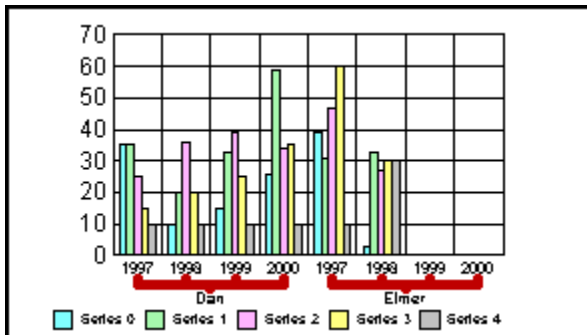
```
IdentObj getNested01LabelLine();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getNested01Label0(),true);
setLineWidth(getNested01LabelLine(),6);
setNestedLabels(true);

```



See [NestedLabels](#) on page 415, [getNested01Label0...8\(\)](#) on page 1082, [getNested01Label0...8Box\(\)](#) on page 1082.

getNonZeroBaseLineZeroLine()

When setNonZeroBaseline() and setNonZeroBaselineValue() methods define a non-zero baseline, this method returns the object of the non-zero baseline.

Syntax:

```
IdentObj getNonZeroBaseLineZeroLine(IdentObj axisObj);
```

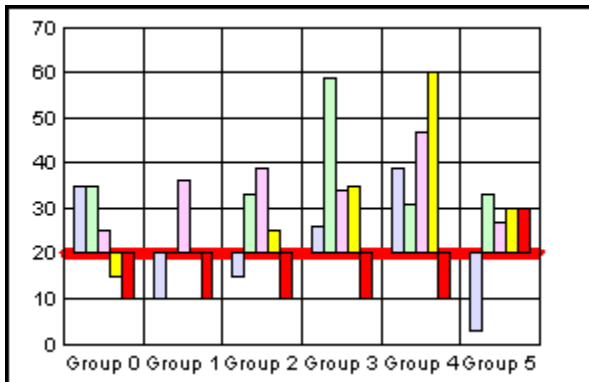
where:

axisObj

Is the object ID of an axis where a non-zero baseline is defined.

Example:

```
setDepthRadius(0);
setNonZeroBaseline(getY1Axis(), true);
setNonZeroBaselineValue(getY1Axis(), 20);
setLineWidth(getNonZeroBaseLineZeroLine(getY1Axis()), 5);
setFillColor(getNonZeroBaseLineZeroLine(getY1Axis()),
              new Color (255, 0, 0));
```



getO1Axis()

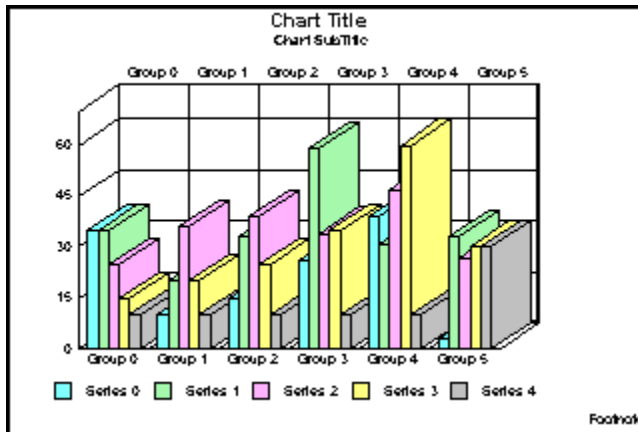
This method returns the object ID of the primary ordinal (O1) axis.

Syntax:

```
IdentObj getO1Axis();
```


Example:

```
setAxisSide(getO1Axis(),2);
```



See [getO1Label\(\)](#) on page 1086, [getO1MajorGrid\(\)](#) on page 1087, [getO1MinorGrid\(\)](#) on page 1089, [getO1Title\(\)](#) on page 1091.

getO1AxisLine()

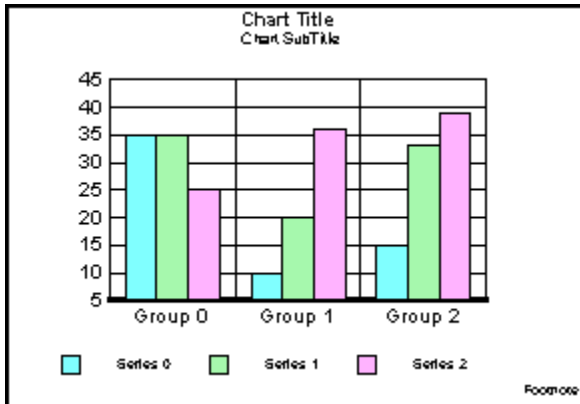
This method returns the object ID of the O1-axis base line.

Syntax:

```
IdentObj getO1AxisLine();
```

Example:

```
setDepthRadius(0);  
setLineWidth(getO1AxisLine(),5);
```



See [getO1Label\(\)](#) on page 1086, [getO1MajorGrid\(\)](#) on page 1087, [getO1MinorGrid\(\)](#) on page 1089, [getO1Title\(\)](#) on page 1091.

getO1Label()

This method returns the object ID of labels on the O1-axis.

Syntax:

```
IdentObj getO1Label();  
IdentObj getO1Label(int g);
```

where:

g

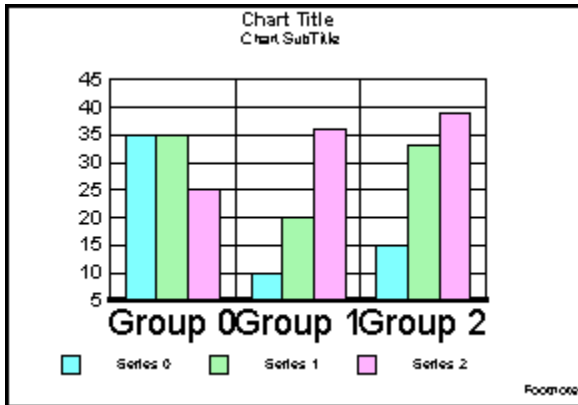
Is an optional group number.

Example:

```

setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setDepthRadius(0);
setDepthAngle(0);
setLineWidth(getO1AxisLine(),5);
setAutofit(getO1Label(),false);
setFontSizeVC(getO1Label(),2800);

```



See [getO1Axis\(\)](#) on page 1084, [getO1AxisLine\(\)](#) on page 1085, [getO1MajorGrid\(\)](#) on page 1087, [getO1MinorGrid\(\)](#) on page 1089, [getO1Title\(\)](#) on page 1091.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getO1MajorGrid()

This method returns the object ID of the primary ordinal (O1) axis grid lines.

Syntax:

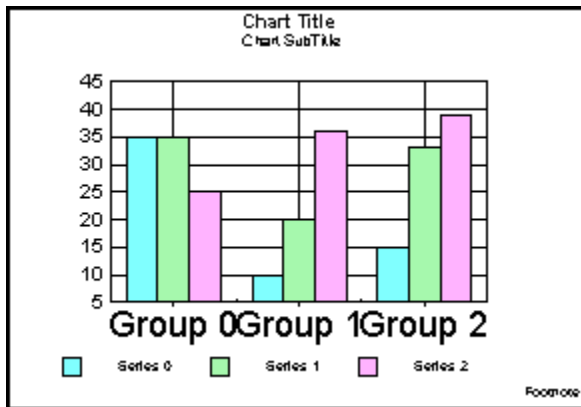
```
IdentObj getO1MajorGrid();
```

Example:

```

setDataRange(0,0,2,2);
setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setDepthRadius(0);
setDepthAngle(0);
setLineWidth(getO1AxisLine(),1);
setAutofit(getO1Label(),false);
setFontSizeVC(getO1Label(),2800);
setDisplay(getO1MinorGrid(),true);
setGridStyle(getO1MajorGrid(),2);
setGridStyle(getO1MinorGrid(),1);

```



See [getO1Axis\(\)](#) on page 1084, [getO1AxisLine\(\)](#) on page 1085, [getO1Label\(\)](#) on page 1086, [getO1MinorGrid\(\)](#) on page 1089, [getO1Title\(\)](#) on page 1091.

getO1MajorTick()

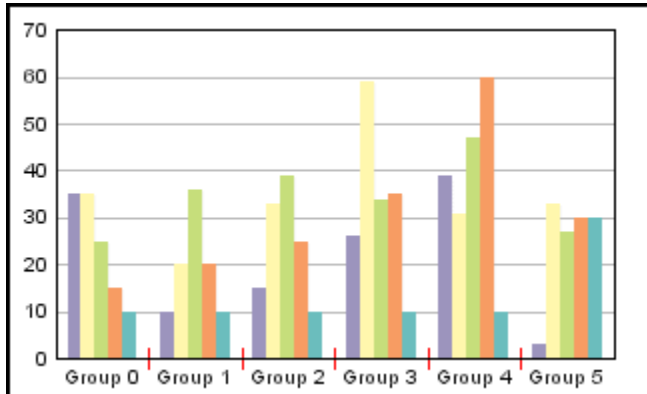
This method returns the object ID of major tick marks on the primary ordinal (O1) axis. The tick marks are enabled with the O1MajorTickDisplay property.

Syntax:

```
IdentObj getO1MajorTick();
```

Example:

```
setO1MajorTickDisplay(true);
setTickLength(getO1MajorTick(), 1000);
setBorderColor(getO1MajorTick(), new Color(255,0,0));
```

**getO1MinorGrid()**

This method returns the object ID of the primary ordinal (O1) axis minor grid lines.

Syntax:

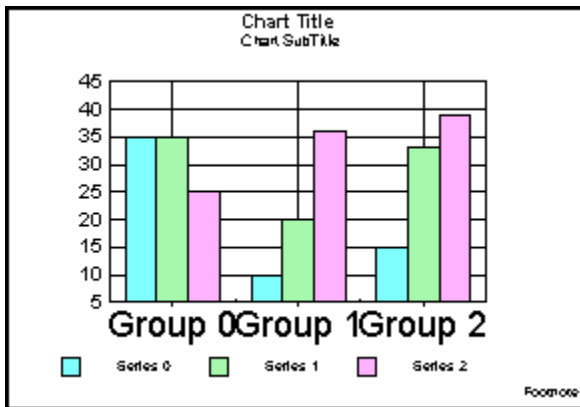
```
IdentObj getO1MinorGrid();
```

Example:

```

setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setDataRange(0,0,2,2);
setDepthRadius(0);
setDepthAngle(0);
setLineWidth(getO1AxisLine(),1);
setAutofit(getO1Label(),false);
setFontSizeVC(getO1Label(),2800);
setDisplay(getO1MinorGrid(),true);
setGridStyle(getO1MajorGrid(),2);
setGridStyle(getO1MinorGrid(),1);

```



See [getO1Axis\(\)](#) on page 1084, [getO1AxisLine\(\)](#) on page 1085, [getO1Label\(\)](#) on page 1086, [getO1MajorGrid\(\)](#) on page 1087, [getO1Title\(\)](#) on page 1091.

getO1MinorTick()

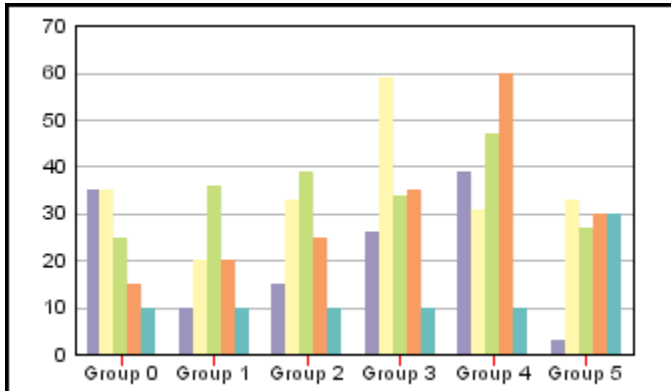
This method returns the object ID of minor tick marks on the primary ordinal (O1) axis. The tick marks are enabled with the O1MinorTickDisplay property.

Syntax:

```
IdentObj getO1MinorTick();
```

Example:

```
setO1MinorTickDisplay(true);
setTickLength(getO1MinorTick(), 1000);
setBorderColor(getO1MinorTick(), new Color(255,0,0));
```

**getO1Title()**

This method returns the object ID of the primary ordinal (O1) axis title.

Syntax:

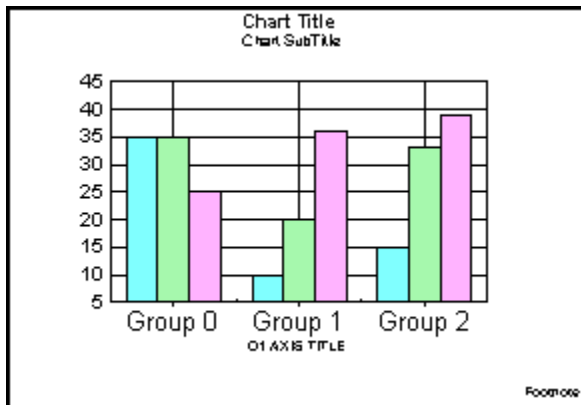
```
IdentObj getO1Title();
```

Example:

```

setViewableSeries(3);
setDataScrollerPresenceSeries(0);
setViewableGroups(3);
setDataScrollerPresenceGroups(0);
setLineWidth(getO1AxisLine(),1);
setO1LabelAutofit(false);
setFontSizeVC(getO1Label(),2000);
setO1MinorGridDisplay(true);
setO1MajorGridStyle(2);
setO1MinorGridStyle(1);
setO1TitleString("O1 AXIS TITLE");
setDisplay(getO1Title(),true);
setLegendDisplay(false);

```



See [getO1Axis\(\)](#) on page 1084, [getO1AxisLine\(\)](#) on page 1085, [getO1Label\(\)](#) on page 1086, [getO1MajorGrid\(\)](#) on page 1087, [getO1MinorGrid\(\)](#) on page 1089.

getO1TitleBox()

This method returns the object ID of the primary ordinal (O1) axis title box.

Syntax:

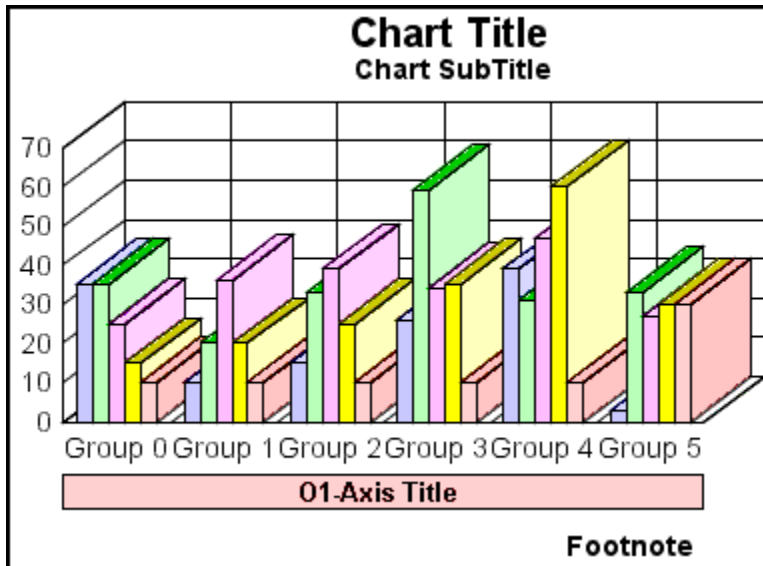
```
IdentObj getO1TitleBox();
```


Example:

```

setLegendDisplay(false);
setO1TitleString("O1-Axis Title");
setDisplay(getO1TitleBox(), true);
setFillColor(getO1TitleBox(), new Color(255,0,0));

```



See [getO1Axis\(\)](#) on page 1084, [getO1AxisLine\(\)](#) on page 1085, [getO1Label\(\)](#) on page 1086, [getO1MajorGrid\(\)](#) on page 1087, [getO1MinorGrid\(\)](#) on page 1089, [getO1Title\(\)](#) on page 1091.

getO2Axis()

This method returns the object ID of the secondary ordinal (O2) axis.

Syntax:

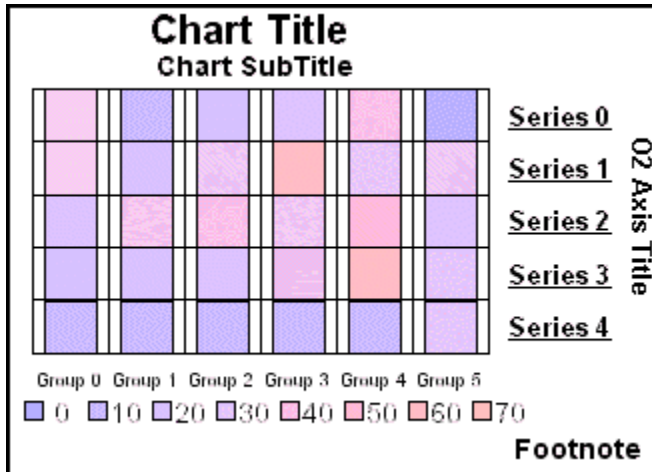
```
IdentObj getO2Axis();
```

Example:

```

setGraphType(87);
setO2TitleString("O2 Axis Title");
setFontSizeVC(getO2Label(),1000);
setFontStyle(getO2Label(),6);
setAxisSide(getO2Axis(), 1);

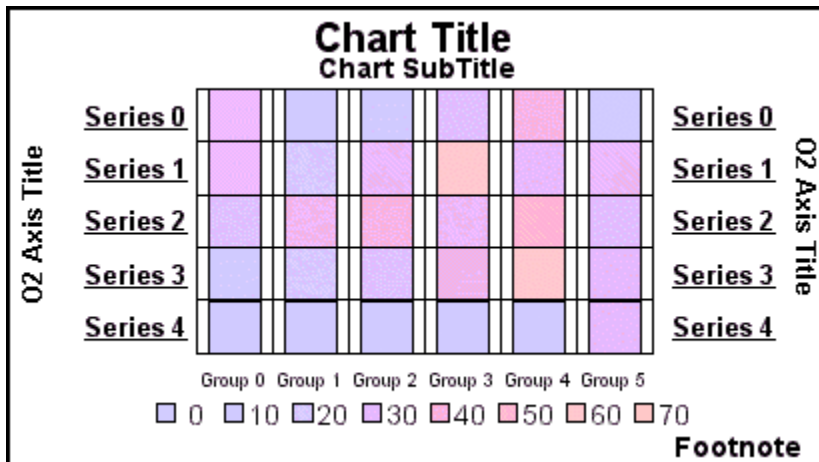
```



```

setGraphType(87);
setO2TitleString("O2 Axis Title");
setFontSizeVC(getO2Label(),1000);
setFontStyle(getO2Label(),6);
setAxisSide(getO2Axis(), 2);

```



See [getO2Label\(\)](#) on page 1095, [getO2MajorGrid\(\)](#) on page 1095, [getO2MinorGrid\(\)](#) on page 1096, [getO2Title\(\)](#) on page 1097.

getO2Label()

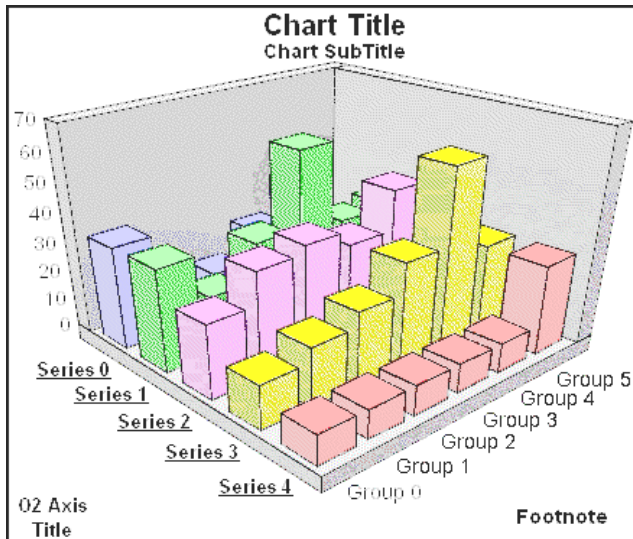
This method returns the object ID of the secondary ordinal (O2) axis labels.

Syntax:

```
IdentObj getO2Label();
```

Example:

```
setGraphType(0);
setO2TitleString("O2 Axis Title");
setFontSizeVC(getO2Label(),1000);
setFontStyle(getO2Label(),6);
```



See [getO2Title\(\)](#) on page 1097.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getO2MajorGrid()

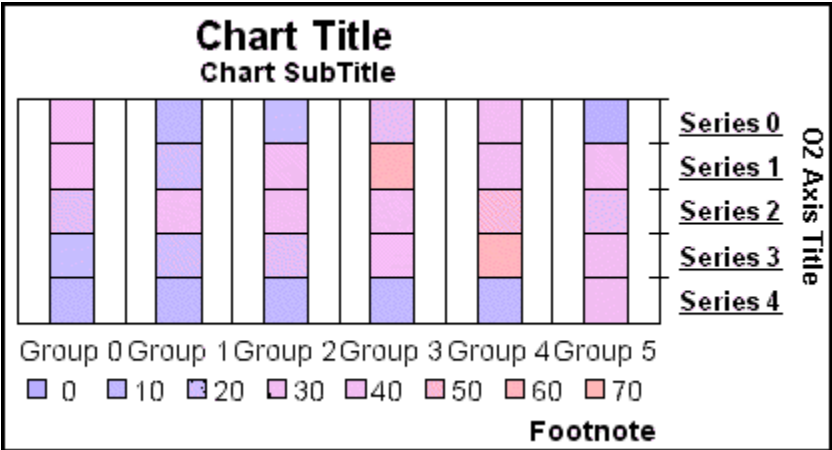
This method returns the object ID of the secondary ordinal (O2) axis major grid lines.

Syntax:

```
IdentObj getO2MajorGrid();
```

Example:

```
setGraphType(87);
setO2TitleString("O2 Axis Title");
setFontSizeVC(getO2Label(),1000);
setFontStyle(getO2Label(),6);
setAxisSide(getO2Axis(), 1);
setGridStyle(getO2MajorGrid(),4);
```



See [getO2Axis\(\)](#) on page 1093, [getO2Label\(\)](#) on page 1095, [getO2MinorGrid\(\)](#) on page 1096, [getO2Title\(\)](#) on page 1097.

getO2MinorGrid()

This method returns the object ID of the secondary ordinal (O2) axis minor grid lines.

Syntax:

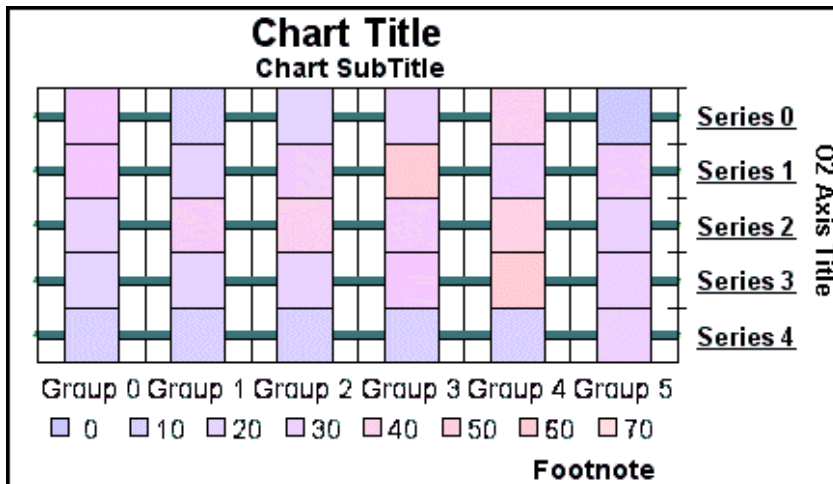
```
IdentObj getO2MinorGrid();
```

Example:

```

setGraphType(87);
setO2TitleString("O2 Axis Title");
setFontSizeVC(getO2Label(),1000);
setFontStyle(getO2Label(),6);
setAxisSide(getO2Axis(), 1);
setGridStyle(getO2MajorGrid(),4);
setDisplay(getO2MinorGrid(), true);
setLineWidth(getO2MinorGrid(), 4);
setFillColor(getO2MinorGrid(), new Color (0,255,0));

```



See [getO2Axis\(\)](#) on page 1093, [getO2Label\(\)](#) on page 1095, [getO2MajorGrid\(\)](#) on page 1095, [getO2Title\(\)](#) on page 1097.

getO2Title()

This method returns the object ID of the secondary ordinal (O2) axis title.

Syntax:

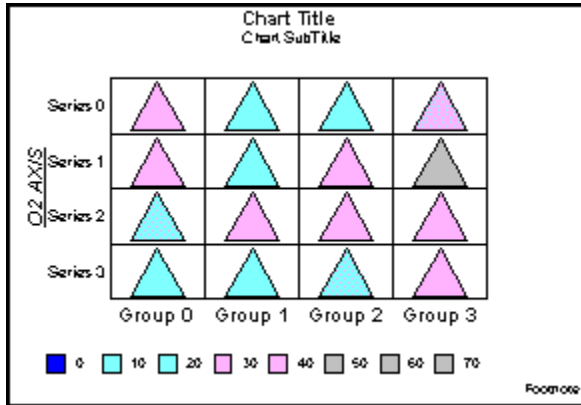
```
IdentObj getO2Title();
```

Example:

```

setDataRange(0,0,3,3);
setO2TitleString("O2 AXIS");
setGraphType(87);
setDisplay(getO2Title(),true);

```



See [getO2Label\(\)](#) on page 1095.

getO2TitleBox()

This method returns the object ID of the secondary ordinal (O2) axis title box.

Syntax:

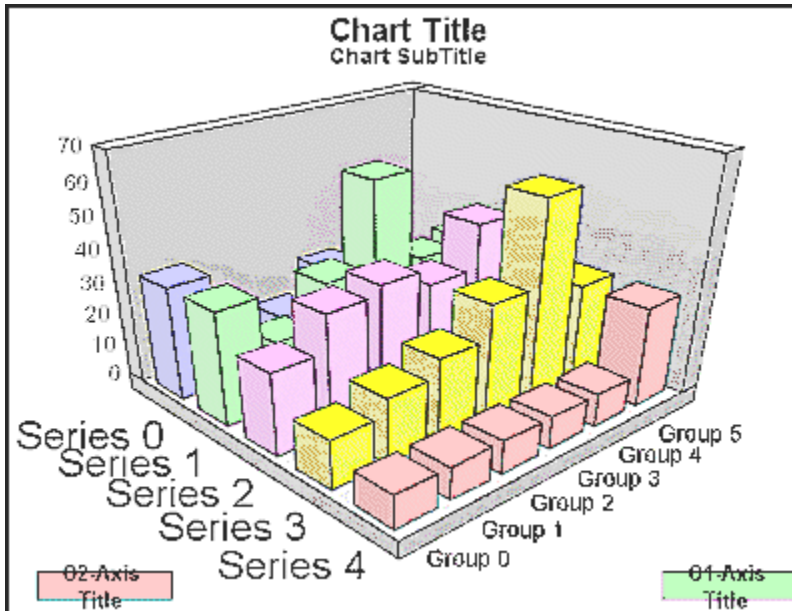
```
IdentObj getO2TitleBox();
```

Example:

```

setGraphType(0);
setLegendDisplay(false);
setO1TitleString("O1-Axis Title");
setDisplay(getO1TitleBox(), true);
setFillColor(getO1TitleBox(), new Color(0,255,0));
setO2TitleString("O2-Axis Title");
setDisplay(getO2TitleBox(), true);
setFillColor(getO2TitleBox(), new Color(255,0,0));
setFootnoteDisplay(false);

```



See [getO2Axis\(\)](#) on page 1093, [getO2Label\(\)](#) on page 1095, [getO2MajorGrid\(\)](#) on page 1095, [getO2MinorGrid\(\)](#) on page 1096, [getO2Title\(\)](#) on page 1097.

getParetoLineMarker()

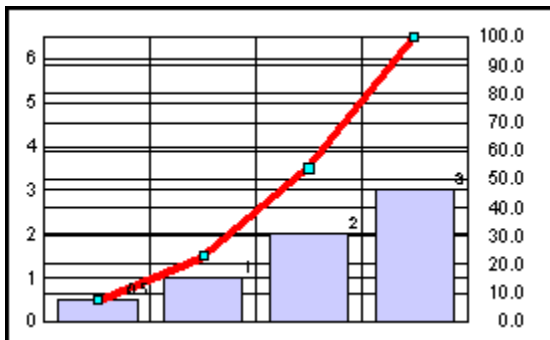
This method returns the object ID of the markers in the line riser in a Pareto graph.

Syntax:

```
IdentObj getParetoLineMarker()
```

Example:

```
setDepthRadius(0);  
setDataTextDisplay(true);  
setData(0,0,.5);  
setData(0,1,1);  
setData(0,2,2);  
setData(0,3,3);  
setDataRangeToExtent()  
setY1TitleString("Y1 Axis");  
setGraphType(102);  
setFillColor(getParetoLineMarker(),  
             new Color (0,255,255));
```



See [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiser()

This method returns the object ID of the line riser in a Pareto graph.

Syntax:

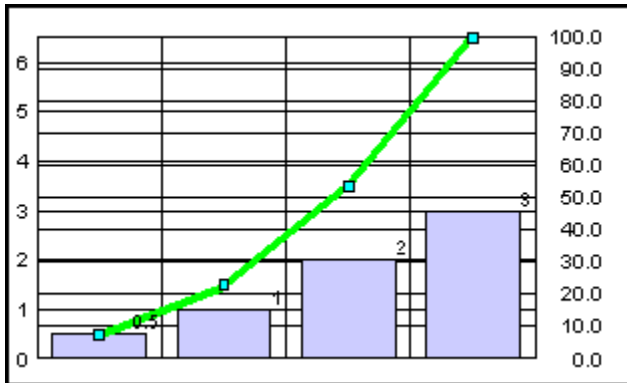
```
IdentObj getParetoLineRiser()
```


Example:

```

setDepthRadius(0);
setDataTextDisplay(true);
setData(0,0,.5);
setData(0,1,1);
setData(0,2,2);
setData(0,3,3);
setDataRangeToExtent()
setGraphType(102);
setFillColor(getParetoLineMarker(), new Color (0,255,255));
setFillColor(getParetoLineRiser(), new Color (0,255,0));

```



See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserSide()

This method returns the object ID of the side of the riser in the Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserSide()
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserSide1()

This method returns the object ID of the side of the line riser1object in a Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserSide1();
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserSide2()

This method returns the object ID of the side of the line riser2object Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserSide2()
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserTop()

This method returns the object ID of the top of risers in a Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserTop2()
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserTop1()

This method returns the object ID of the top of the riser1object in a Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserTop1();
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop2\(\)](#) on page 1102.

getParetoLineRiserTop2()

This method returns the object ID of the top of the riser2object in a Pareto graph.

Syntax:

```
IdentObj getParetoLineRiserTop2();
```

See [getParetoLineMarker\(\)](#) on page 1099, [getParetoLineRiser\(\)](#) on page 1100, [getParetoLineRiserSide\(\)](#) on page 1101, [getParetoLineRiserSide1\(\)](#) on page 1101, [getParetoLineRiserSide2\(\)](#) on page 1102, [getParetoLineRiserTop\(\)](#) on page 1102, [getParetoLineRiserTop1\(\)](#) on page 1102.

getPieBarConnectLineBottom()

This method returns the object ID of the bottom line that connects the bar to a series in a pie-bar graph (GraphType 93 or 94).

Syntax:

```
IdentObj getPieBarConnectLineBottom();
```

See [getPieBarConnectLines\(\)](#) on page 1103, [getPieBarLabel\(\)](#) on page 1104.

getPieBarConnectLines()

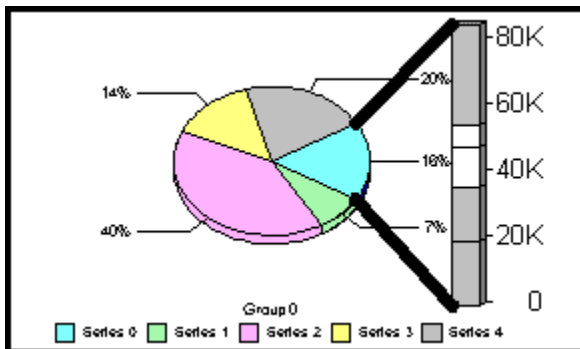
This method returns the object ID of the lines that connect the bar to a series in a pie-bar graph (GraphType 93 or 94).

Syntax:

```
IdentObj getPieBarConnectLines();
```

Example:

```
setGraphType(93);
setLineWidth(getPieBarConnectLines(),10);
```



See [getPieBarConnectLineBottom\(\)](#) on page 1103, [getPieBarLabel\(\)](#) on page 1104.

getPieBarLabel()

This method returns the object ID of the bar label to a series in a pie-bar graph (GraphType 93 or 94).

Syntax:

```
IdentObj getPieBarLabel()
```

See [getPieBarConnectLineBottom\(\)](#) on page 1103, [getPieBarConnectLines\(\)](#) on page 1103.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getPieBarRiserTop()

This method returns the object ID of the top of the bar riser in a pie-bar graph (GraphType 93 or 94).

Syntax:

```
IdentObj getPieBarRiserTop();
```

getPieBarSlice()

This method returns the object ID of a slice in a pie-bar graph (GraphType 93 or 94).

Syntax:

```
IdentObj getPieBarSlice();
```

```
IdentObj getPieBarSlice (int s, int g);
```

where:

s

Is an optional series number

g

Is an optional group number

See [getPieBarSliceCrust\(\)](#) on page 1104.

getPieBarSliceCrust()

This method returns the object ID of the slice crust in a pie-bar graph (GraphType 93 or 94).

Note: This object is only present in the graph if the GraphType property is set to 93 (Pie-Bar graph).

Syntax:

```
IdentObj getPieBarSliceCrust();
```

See [getPieBarSlice\(\)](#) on page 1104.

getPieFeelerLine()

This method returns the object ID of pie feeler lines in a pie graph.

Syntax:

```
IdentObj getPieFeelerLine();  
IdentObj getPieFeelerLine (int s, int g);
```

where:

s

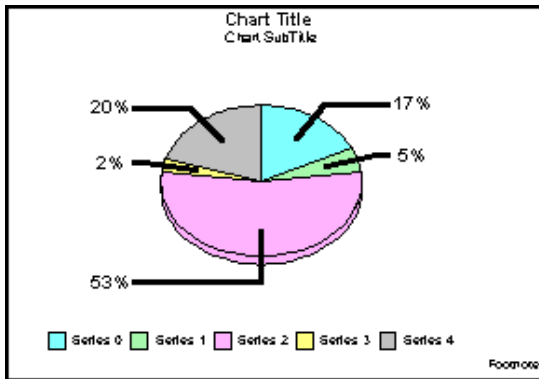
Is an optional series number

g

Is an optional group number

Example:

```
setGraphType(55);  
setDisplay(getPieLabel(),false);  
setLineWidth(getPieFeelerLine(),5);
```



See [getPieFrame\(\)](#) on page 1105, [getPieLabel\(\)](#) on page 1106, [getPieRingLabel\(\)](#) on page 1107, [getPieSliceLabel\(\)](#) on page 1110.

getPieFrame()

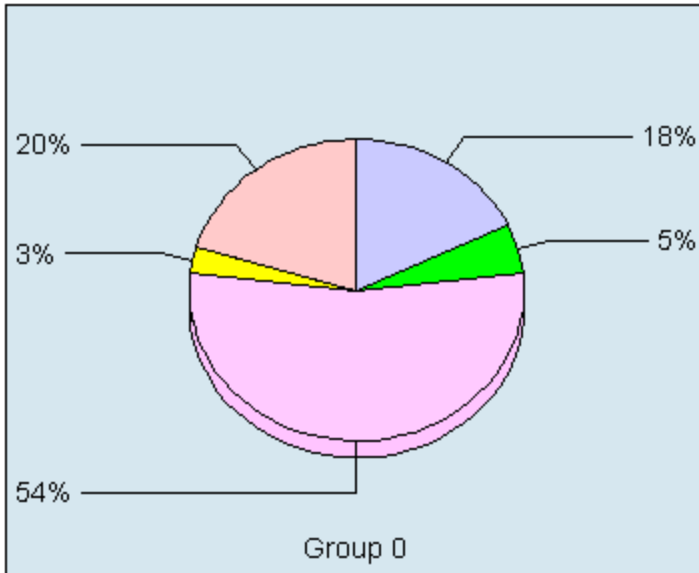
This method returns the object ID of the frame of a pie graph.

Syntax:

```
IdentObj getPieFrame();
```

Example:

```
setGraphType(55);
setDisplay(getPieFrame(), true);
setFillColor(getPieFrame(), new Color (210,225,235));
setBorderColor(getPieFrame(), new Color(0,0,0));
```



See [getPieFeelerLine\(\)](#) on page 1105, [getPieLabel\(\)](#) on page 1106, [getPieRingLabel\(\)](#) on page 1107, [getPieSliceLabel\(\)](#) on page 1110.

getPieLabel()

This method returns the object ID of group labels that may appear in multiple pie graphs.

Syntax:

```
IdentObj getPieLabel();
IdentObj getPieLabel(int g);
```

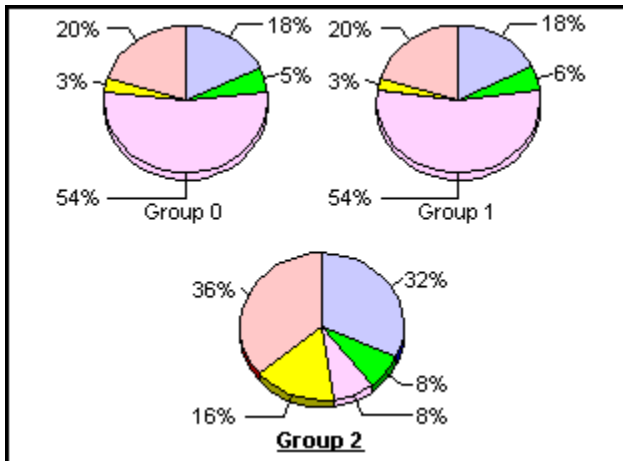
where:

g

Is an optional group number.

Example:

```
setGraphType(57);
setViewableGroups(3);
setFontStyle(getPieLabel(2), 6)
```



See [getPieFeelerLine\(\)](#) on page 1105, [getPieFrame\(\)](#) on page 1105, [getPieRingLabel\(\)](#) on page 1107, [getPieSliceLabel\(\)](#) on page 1110.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getPieRingLabel()

This method returns the object ID of the total label that may appear in the center ring of a ring pie graph.

Syntax:

```
IdentObj getPieRingLabel();
IdentObj getPieRingLabel(int g);
```

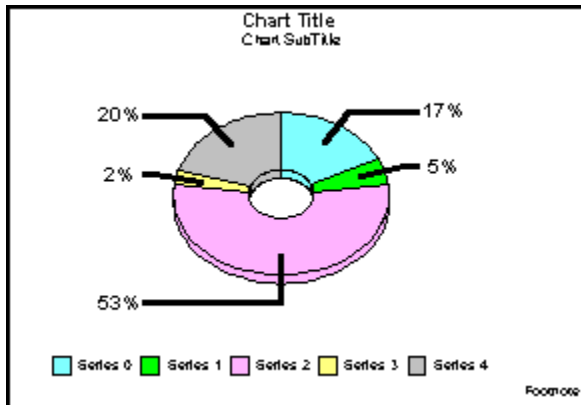
where:

g

Is an optional group number.

Example:

```
setGraphType(56);  
setDisplay(getPieLabel(), false);  
setDisplay(getPieRingLabel(), false);
```



See [getPieFeelerLine\(\)](#) on page 1105, [getPieFrame\(\)](#) on page 1105, [getPieLabel\(\)](#) on page 1106, [getPieSliceLabel\(\)](#) on page 1110.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getPieSliceFeelerPercent()

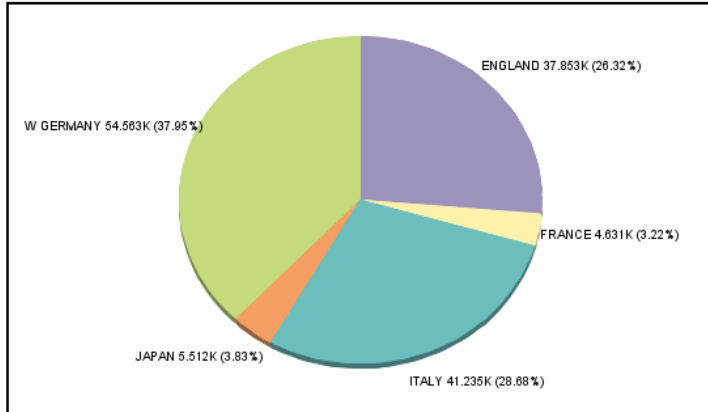
This method returns the object ID of the percent value labels in a pie chart when setPieLabelDisplay is set to 5 (absolute and percent values) or 6 (absolute value, percent value, and series label).

Syntax:

```
int getPieSliceFeelerPercent();
```


Example:

```
setPieLabelDisplay(6);
setTextFormatPreset(getPieSliceFeelerPercent(),4);
```



See [PieLabelDisplay](#) on page 481, [setTextFormatPreset\(\)](#) on page 995.

getPieSliceFeelerValue()

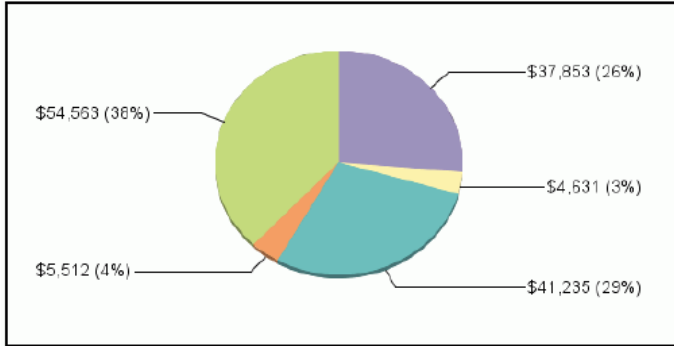
This method returns the object ID of the absolute value labels in a pie chart when `setPieLabelDisplay` is set to 5 (absolute and percent values) or 6 (absolute value, percent value, and series label).

Syntax:

```
int getPieSliceFeelerValue();
```

Example:

```
setPieLabelDisplay(5);  
setTextFormatPreset(getPieSliceFeelerValue(),6);
```



See [PieLabelDisplay](#) on page 481, [setTextFormatPreset\(\)](#) on page 995.

getPieSliceLabel()

This method returns the object ID slice labels in a pie graph.

Syntax:

```
IdentObj getPieSliceLabel();  
IdentObj getPieSliceLabel(int s, int g);
```

where:

s

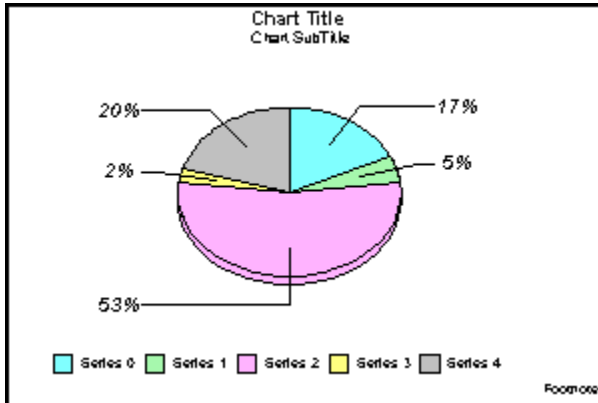
Is an optional series number.

g

Is an optional group number.

Example:

```
setFontSizeVC(getPieSliceLabel(),1623);
setFontStyle(getPieSliceLabel(),3);
setGraphType(55);
setDisplay(getPieLabel(),false);
```



See [getPieFeelerLine\(\)](#) on page 1105, [getPieFrame\(\)](#) on page 1105, [getPieLabel\(\)](#) on page 1106, [getPieRingLabel\(\)](#) on page 1107.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getPlaceHorzO1LabelsTitle()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method returns the object ID of a rectangle that is the virtual coordinate location/size of the O1-Axis Labels and Titles. This method is used for horizontal graph types. The important coordinates of this rectangle identifier are X-location and height. The `setPlaceRect()` method is used with this object.

Syntax:

```
IdentObj getPlaceHorzO1LabelsTitle();
```

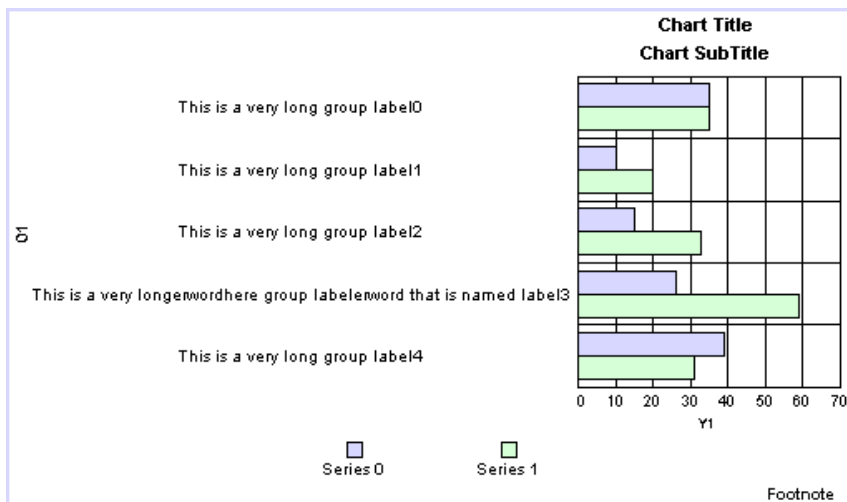
Example:

```

setGraphType(24);
setGroupLabel(getGroup(0),
    "This is a very long group label0");
setGroupLabel(getGroup(1),
    "This is a very long group label1");
setGroupLabel(getGroup(2),
    "This is a very long group label2");
setGroupLabel(getGroup(3), "This is a very
longerwordhere group labelerword that is named label3");
setGroupLabel(getGroup(4),
    "This is a very long group label4");
setDisplay(getY1Title(),true);
setDisplay(getO1Title(),true);
setTextString(getY1Title(),"Y1");
setTextString(getO1Title(),"O1");
setPlaceSkip(getO1Label(), 0);
setFontSizeAbsolute(getO1Label(), true);
setFontSize(getO1Label(), 10);
setPlaceMinFontSize(getO1Label(), 10);
setPlaceRect(getPlaceHorzO1LabelsTitle(),
    new Rectangle(-14000, -15500, 28000, 24000));
setTextJustVert(getO1Label(),1);
setDataRange(0,0,1,4);

setPlace(true);
setDepthRadius(0);

```



See [getPlaceVertO1LabelsTitle\(\)](#) on page 1113.

getPlaceVertO1LabelsTitle()

When the automatic graph layout manager is enabled with `setPlace(true)`, this method returns the object ID of a rectangle that is the virtual coordinate location and size of both the O1-Axis Labels and Titles. This method is used for vertical graph types.

Note: The important coordinates of this rectangle identifier are Y-location and height. The `setPlaceRect()` method is used with this object.

Syntax:

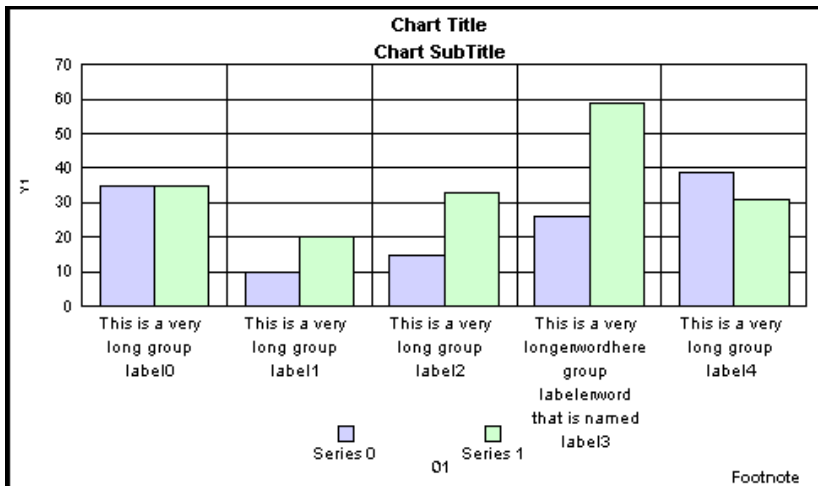
```
IdentObj getPlaceVertO1LabelsTitle();
```

Example:

```

restoreDefaults();
setGroupLabel(
    getGroup(0), "This is a very long group label0");
setGroupLabel(
    getGroup(1), "This is a very long group label1");
setGroupLabel(
    getGroup(2), "This is a very long group label2");
setGroupLabel(getGroup(3), "This is a very longerwordhere group
labelerword that is named label3");
setGroupLabel(
    getGroup(4), "This is a very long group label4");
setDisplay(getY1Title(),true);
setDisplay(getO1Title(),true);
setTextString(getY1Title(),"Y1");
setTextString(getO1Title(),"O1");
setPlaceSkip(getO1Label(), 0);
setFontSizeAbsolute(getO1Label(), true);
setFontSize(getO1Label(), 10);
setPlaceMinFontSize(getO1Label(), 10);
setPlaceRect(getPlaceVertO1LabelsTitle(),
    new Rectangle(-14000, -15500, 29000, 24000));
setTextJustVert(getO1Label(),0);
setDataRange(0,0,1,4);
setPlace(true);
setDepthRadius(0);

```



See [getPlaceHorzO1LabelsTitle\(\)](#) on page 1111.

getQuadrantLine()

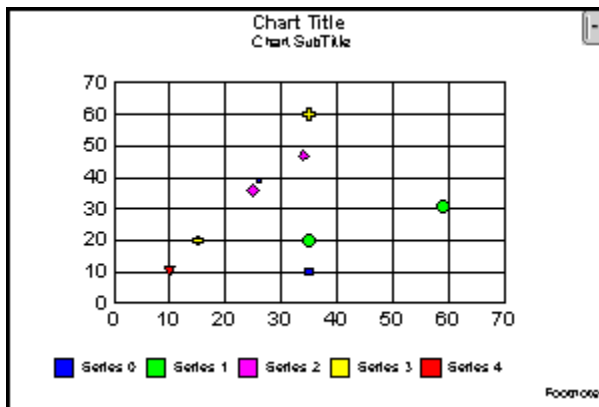
This method returns the object ID of the quadrant lines in a bubble graph.

Syntax:

```
IdentObj getQuadrantLine();
```

Example:

```
setGraphType(89);  
setDisplay(getQuadrantLine(), false);
```



See [QuadrantLineCountX](#) on page 499, [QuadrantLineCountY](#) on page 500.

getReferenceLine()

This method returns the object ID of a reference line. This object will only exist in a graph if the `setReferenceLine()` method has been used to create the line.

Syntax:

```
IdentObj getReferenceLine(IdentObj axisObj, int nIndex);
```

where:

axisObj

Is the Axis object ID.

nIndex

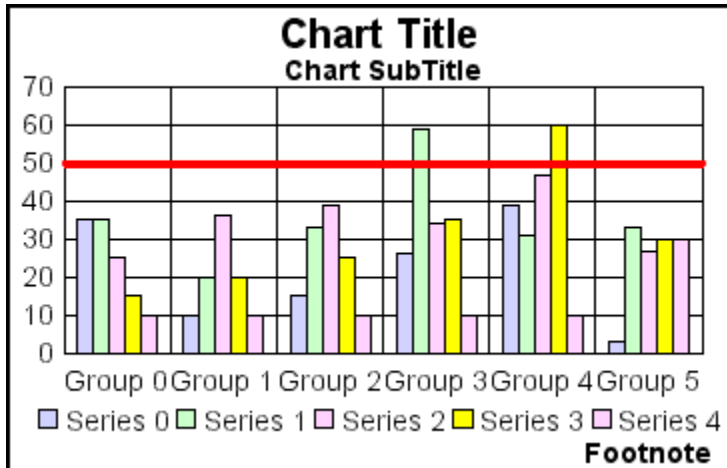
Is the integer reference line index number (0...2).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setReferenceLine(getY1Axis(),0,50);
setLineWidth(getReferenceLine(getY1Axis(),0),3);
setFillColor(getReferenceLine(getY1Axis(),0), new Color(255,0,0));

```



See [setReferenceLine\(\)](#) on page 964.

getReferenceLineLegendText()

This method returns the object ID of the legend text associated with a reference line.

Syntax:

```

IdentObj getReferenceLineLegendText;(
    IdentObj axisObj, int nIndex);

```

where:

axisObj

Is the Axis object ID.

nIndex

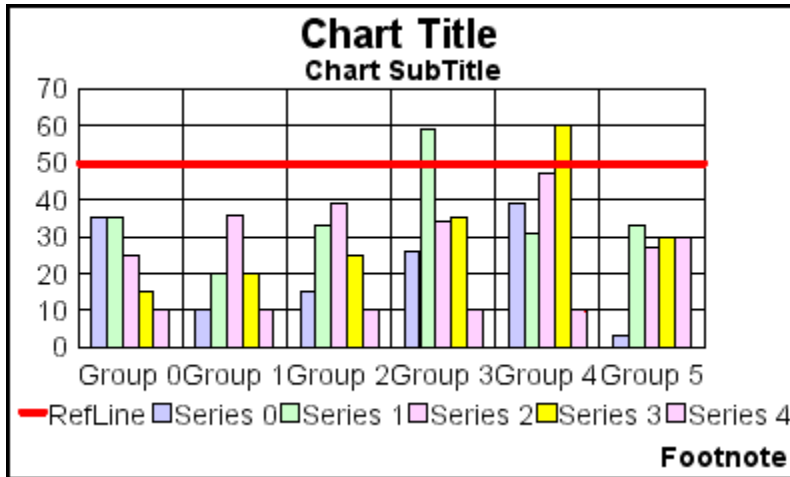
Is the integer reference line index number (0...2).

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setReferenceLine(getY1Axis(),0,50);
setLineWidth(getReferenceLine(getY1Axis(),0),3);
setFillColor(getReferenceLine(getY1Axis(),0), new Color(255,0,0));
setTextString(getReferenceLine(getY1Axis(),0),"RefLine");
setDisplay(getReferenceLineLegendText(getY1Axis(),0),true)

```



See [setReferenceLine\(\)](#) on page 964.

getReferenceLineO1()

This method gets the reference line object for axis O1. This object will only exist in a graph if the `setDisplay(getReferenceLineO1(int), true)` method is used to show the line.

Syntax:

```
IdentObj getReferenceLineO1(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineTextO1\(\)](#) on page 1120, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineO2()

This method gets the reference line object for axis O2. This object will only exist in a graph if the `setDisplay(getReferenceLineO2(int), true)` method is used to show the line.

Syntax:

```
IdentObj getReferenceLineO2(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineTextO2\(\)](#) on page 1120, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineX1()

This method gets the reference line object for axis X1. This object will only exist in a graph if the `setDisplay(getReferenceLineX1(int), true)` method is used to show the line.

Syntax:

```
IdentObj getReferenceLineX1(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineTextX1\(\)](#) on page 1120, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineY1()

This method gets the reference line object for axis Y1. This object will only exist in a graph if the `setDisplay(getReferenceLineY1(int), true)` method is used to show the line.

Syntax:

```
IdentObj getReferenceLineY1(int);
```

where:

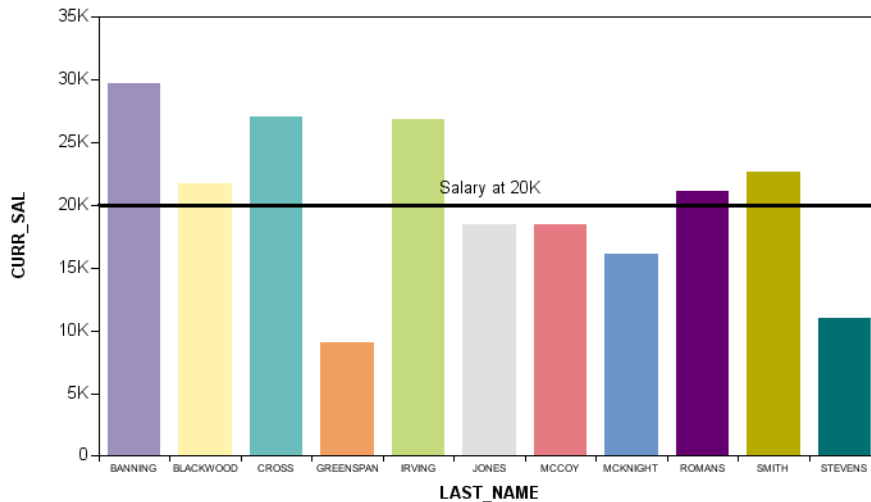
int

Is the identifier for the reference line, where 0 identifies the first reference line.

Example:

The following code turns on a reference line for the Y1 axis, sets the value at which the line displays to 20,000, creates a label for the reference line, and turns on the display of the label:

```
setDisplay(getReferenceLineY1(0), true);
setReferenceLineValueY1(0, 20000.0);
setLineWidth(getReferenceLineY1(0), 2);
setTextString(getReferenceLineTextY1(0), "Salary at 20K");
setDisplay(getReferenceLineTextY1(0), true);
```



See [getReferenceLineTextY1\(\)](#) on page 1121, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineY2()

This method gets the reference line object for axis Y2. This object will only exist in a graph if the `setDisplay(getReferenceLineY2(int), true)` method is used to show the line.

Syntax:

```
IdentObj getReferenceLineY2(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineTextY2\(\)](#) on page 1121, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineTextO1()

This method gets the text object for a reference line for axis O1. The text object only exists if the `setTextString(getReferenceLineTextO1(int),"label")` method has been used to assign a text label for the reference line.

Syntax:

```
IdentObj getReferenceLineTextO1(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineO1\(\)](#) on page 1117, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineTextO2()

This method gets the text label for a reference line for axis O2. The text object only exists if the `setTextString(getReferenceLineTextO2(int),"label")` method has been used to assign a text label for the reference line.

Syntax:

```
IdentObj getReferenceLineTextO2(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineO2\(\)](#) on page 1118, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineTextX1()

This method gets the text label for a reference line for axis X1. The text object only exists if the `setTextString(getReferenceLineTextX1(int),"label")` method has been used to assign a text label for the reference line.

Syntax:

```
IdentObj getReferenceLineTextX1(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineX1\(\)](#) on page 1118, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineTextY1()

This method gets the text label for a reference line for axis Y1. The text object only exists if the `setTextString(getReferenceLineTextY1(int), "label")` method has been used to assign a text label for the reference line.

Syntax:

```
IdentObj getReferenceLineTextY1(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineY1\(\)](#) on page 1118, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getReferenceLineTextY2()

This method gets the text label for a reference line for axis Y2. The text object only exists if the `setTextString(getReferenceLineTextY2(int), "label")` method has been used to assign a text label for the reference line.

Syntax:

```
IdentObj getReferenceLineTextY2(int);
```

where:

int

Is the identifier for the reference line, where 0 identifies the first reference line.

See [getReferenceLineY2\(\)](#) on page 1119, [getReferenceLineY1\(\)](#) on page 1118, [APIs Related to Reference Lines](#) on page 184.

getScatterQuadrantLine()

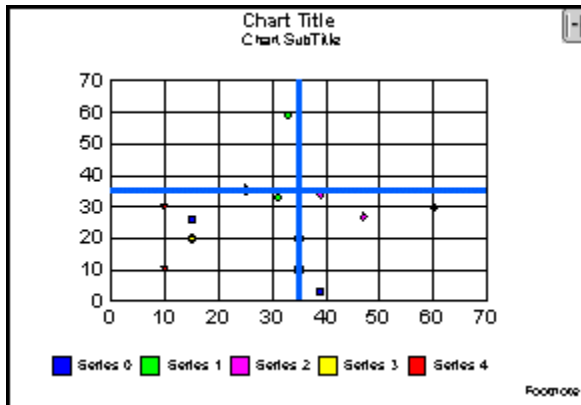
This method returns the object ID of the quadrant lines in a scatter graph.

Syntax:

```
IdentObj getScatterQuadrantLine();
```

Example:

```
setDisplay(getScatterQuadrantLine(),true);  
setGraphType(61);
```



getSeries()

This method returns the object ID of the series specified by nSeries.

Syntax:

```
IdentObj getSeries(int nSeries);
```

where:

nSeries

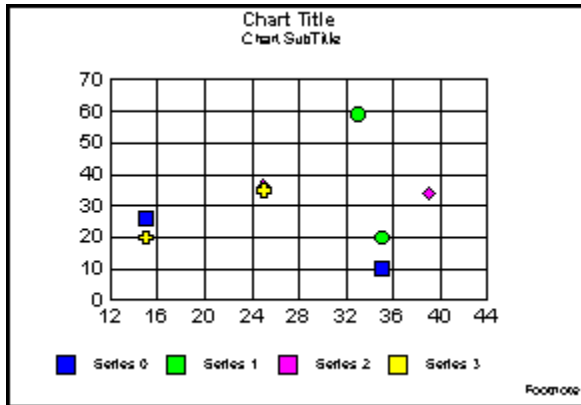
Is any valid series number.

Example:

```

setDataRange(0,0,3,3);
setDepthAngle(0);
setDepthRadius(0);
setMarkerSizeDefault(60);
setMarkerShape(getSeries(0),1);
setMarkerShape(getSeries(1),2);
setMarkerShape(getSeries(2),3);

```



See [getSeriesGroup\(\)](#) on page 1123.

getSeriesGroup()

This method returns the object ID of the series associated with nSeries and nGroup.

Syntax:

```
IdentObj getSeriesGroup(int nSeries, int nGroup);
```

where:

nSeries

Is any valid series number.

nGroup

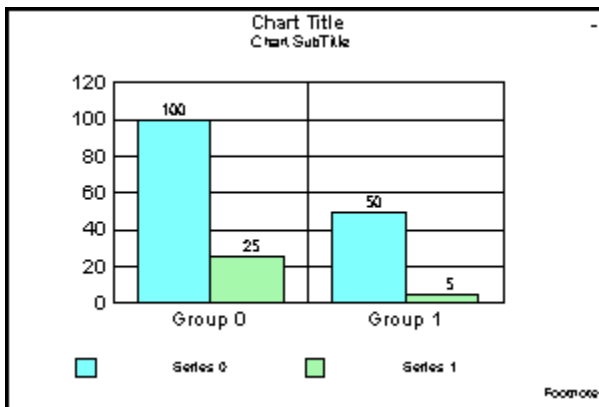
Is any valid group number.

Example:

```

setDataRange(1,1,2,2);
setViewableSeries(2);
setDataScrollerPresenceSeries(0);
setViewableGroups(2);
setDataScrollerPresenceGroups(0);
setDataTextDisplay(true);
setDataTextPosition(0);
setDataTextRadius(0,0,20);
setDataTextRadius(0,1,20);
setDataTextRadius(1,0,20);
setDataTextRadius(1,1,20);
setDataValue(getSeriesGroup(0,0),100);
setDataValue(getSeriesGroup(0,1),50);
setDataValue(getSeriesGroup(1,0),25);
setDataValue(getSeriesGroup(1,1),5);
setDepthAngle(0);
setDepthRadius(0);

```



See [getSeries\(\)](#) on page 1122.

getSeriesScroller()

This method returns the object ID of the series data scroller bar that can be enabled and displayed with the DataScrollerPresenceSeries property.

Syntax:

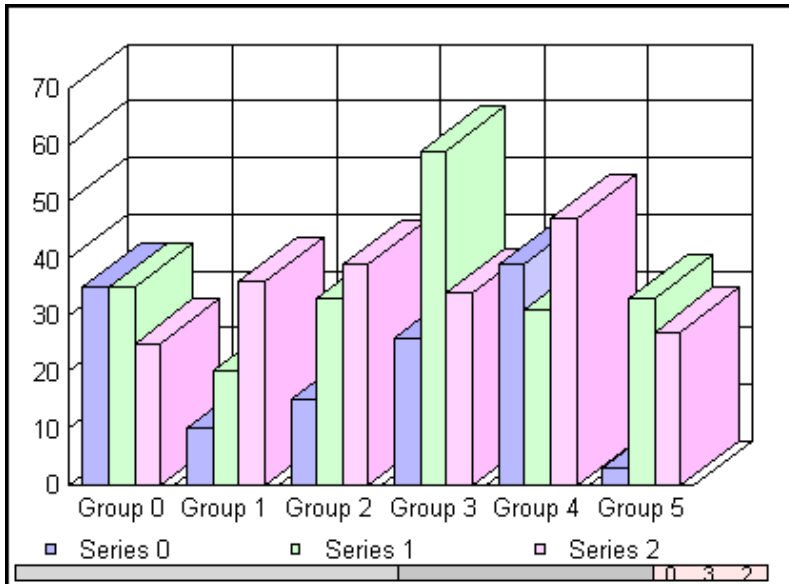
```
IdentObj getSeriesScroller();
```


Example:

```

setAutofit(getSeriesScrollerEditRight(), false);
setAutofit(getSeriesScrollerEditThumb(), false);
setAutofit(getSeriesScrollerEditLeft(), false);
setAxisTextAutofitMode(0);
setFontSizeVC(getSeriesScrollerEditRight(), 927);
setFontSizeVC(getSeriesScrollerEditThumb(), 927);
setFontSizeVC(getSeriesScrollerEditLeft(), 927);
setY1MajorGridStep(10.0);
setRect(getSeriesScroller(),
        new Rectangle(-14180, -13600, 20124, 927));
setScrollOffsetGroup(0);
setScrollOffsetSeries(0);
setViewableGroups(0);
setViewableSeries(3);

```



See [getSeriesScrollerEditBox\(\)](#) on page 1126, [getSeriesScrollerEditLeft\(\)](#) on page 1126, [getSeriesScrollerEditRight\(\)](#) on page 1126, [getSeriesScrollerEditThumb\(\)](#) on page 1127, [getGroupScrollerThumb\(\)](#) on page 1074, [DataScrollerPresenceSeries](#) on page 311.

getSeriesScrollerEditBox()

This method returns the object ID of the series data scroller edit box that can be enabled and displayed with the DataScrollerPresenceSeries property. The edit box is the right-most portion of the scroll bar that contains three numbers identifying the scroll offset series, viewable series, and total number of series. The edit box can be edited in the user interface.

Syntax:

```
IdentObj getSeriesScrollerEditBox();
```

See [getSeriesScroller\(\)](#) on page 1124, [getSeriesScrollerEditLeft\(\)](#) on page 1126, [getSeriesScrollerEditRight\(\)](#) on page 1126, [getSeriesScrollerEditThumb\(\)](#) on page 1127, [getGroupScrollerThumb\(\)](#) on page 1074, [DataScrollerPresenceSeries](#) on page 311.

getSeriesScrollerEditLeft()

This method returns the object ID of the series data scroller left edit bar that can be enabled and displayed with the DataScrollerPresenceSeries property. This object is the left-most digit in the series scroller edit box.

Syntax:

```
IdentObj getSeriesScrollerEditLeft();
```

See [getSeriesScroller\(\)](#) on page 1124 for an example. Also see [getSeriesScrollerEditBox\(\)](#) on page 1126, [getSeriesScrollerEditRight\(\)](#) on page 1126, [getSeriesScrollerEditThumb\(\)](#) on page 1127, [getGroupScrollerThumb\(\)](#) on page 1074, [DataScrollerPresenceSeries](#) on page 311.

getSeriesScrollerEditRight()

This method returns the object ID of the series data scroller right edit bar that can be enabled and displayed with the DataScrollerPresenceSeries property. This object is the right-most digits in the series scroller edit box.

Syntax:

```
IdentObj getSeriesScrollerEditRight();
```

See [getSeriesScroller\(\)](#) on page 1124 for an example. Also see [getSeriesScrollerEditBox\(\)](#) on page 1126, [getSeriesScrollerEditLeft\(\)](#) on page 1126, [getSeriesScrollerEditThumb\(\)](#) on page 1127, [getGroupScrollerThumb\(\)](#) on page 1074, [DataScrollerPresenceSeries](#) on page 311.

getSeriesScrollerEditThumb()

This method returns the object ID of the series data scroller bar edit thumb that can be enabled and displayed with the `DataScrollerPresenceSeries` property. This object is the center digit in the series scroller edit box.

Syntax:

```
IdentObj getSeriesScrollerEditThumb();
```

See [getSeriesScroller\(\)](#) on page 1124, [getSeriesScrollerEditBox\(\)](#) on page 1126, [getSeriesScrollerEditLeft\(\)](#) on page 1126, [getSeriesScrollerEditRight\(\)](#) on page 1126, [DataScrollerPresenceSeries](#) on page 311.

getSeriesScrollerThumb()

This method returns the object ID of the series scroller thumb. The series scroller bar is enabled and displayed with the `DataScrollerPresenceSeries` property. The thumb is the movable area of the scroller bar that can be used to adjust the series offset.

Syntax:

```
IdentObj getSeriesScrollerThumb();
```

See [getSeriesScroller\(\)](#) on page 1124, [getSeriesScrollerEditBox\(\)](#) on page 1126, [getSeriesScrollerEditLeft\(\)](#) on page 1126, [getSeriesScrollerEditRight\(\)](#) on page 1126, [getSeriesScrollerEditThumb\(\)](#) on page 1127, [DataScrollerPresenceSeries](#) on page 311.

getSlice()

This method returns the object ID of a slice in a pie graph.

Syntax:

```
IdentObj getSlice();  
IdentObj getSlice(int s, int g);
```

where:

s

Is an optional series number.

g

Is an optional group number.

See [getSliceCrust\(\)](#) on page 1127, [getSliceRing\(\)](#) on page 1128.

getSliceCrust()

This method returns the object ID of the slice crust in a pie graph.

Syntax:

```
IdentObj getSliceCrust();
```

See [getSlice\(\)](#) on page 1127, [getSliceRing\(\)](#) on page 1128.

getSliceRing()

This method returns the object ID of a slice in a ring pie graph.

Syntax:

```
IdentObj getSliceRing();
```

See [getSliceCrust\(\)](#) on page 1127, [getSlice\(\)](#) on page 1127.

getSpectralLegendMarker()

This method returns the object ID of the legend marker in a spectral map.

Syntax:

```
IdentObj getSpectralLegendMarker();  
IdentObj getSpectralLegendMarker(int i);
```

where:

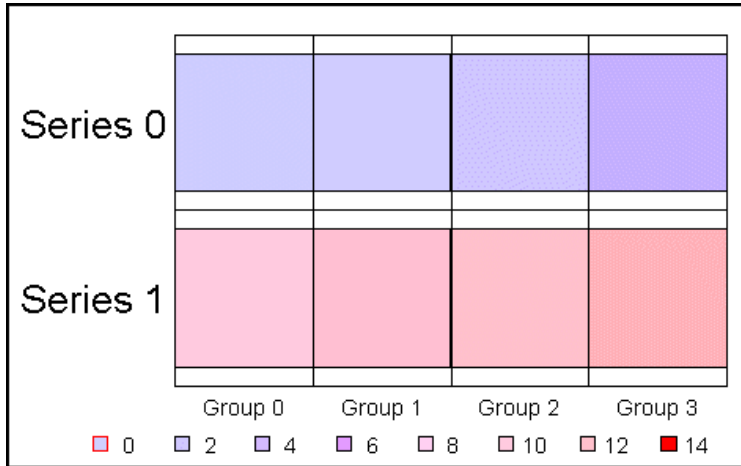
i

Is the legend marker number.

Example:

```
setGraphType(87);  
setData(0,0,.5);  
setData(0,1,1);  
setData(0,2,2);  
setData(0,3,3);  
setData(1,0,10);  
setData(1,1,11);  
setData(1,2,12);  
setData(1,3,13);  
setDataRangeToExtent();  
setSeriesLabel(0,"Series 0");  
setSeriesLabel(1,"Series 1");  
setGroupLabel(0,"Group 0");  
setGroupLabel(1,"Group 1");  
setGroupLabel(2,"Group 2");  
setGroupLabel(3,"Group 3");
```

```
setBorderColor(getSpectralLegendMarker(0), new Color (255,0,0));
```



See [getSpectralMarker\(\)](#) on page 1129.

getSpectralMarker()

This method returns the object ID of the markers in a spectral map.

Syntax:

```
IdentObj getSpectralMarker();
```

See [getSpectralLegendMarker\(\)](#) on page 1128.

getStackConnectLine()

This method returns the object ID of a connected line in a stacked chart.

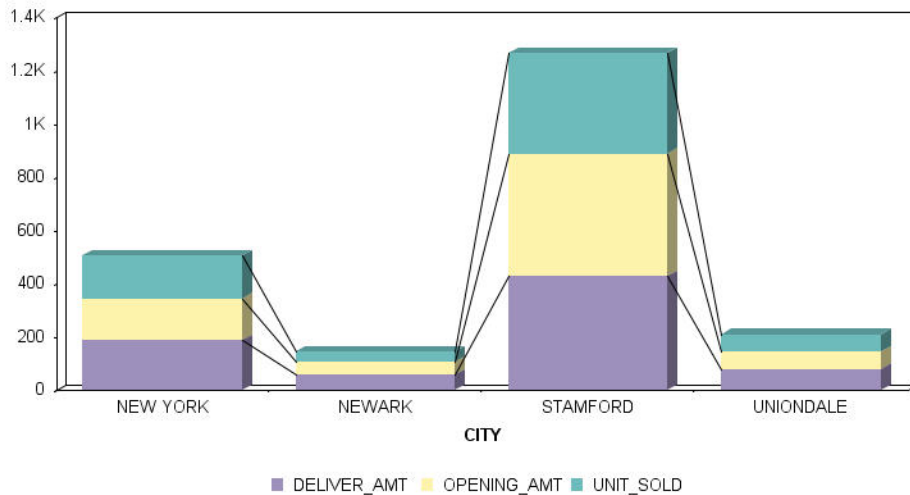
Syntax:

```
IdentObj getStackConnectLine();
```

Example

Using this method with `setDisplay` and setting the value to `true` connects the bars in a stacked bar chart with lines.

```
setDisplay(getStackConnectLine(), true);
```



getStock52WeekHighLine()

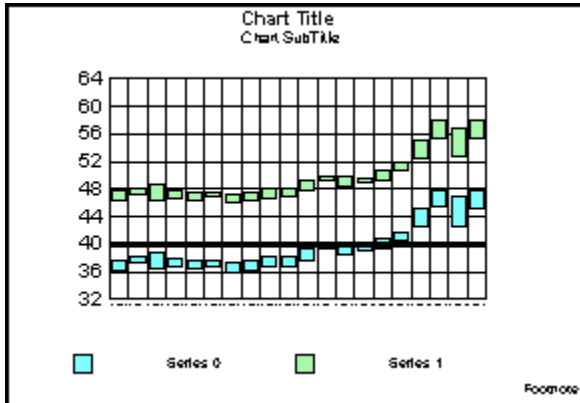
This method returns the object ID of the 52-week high line that may appear in a stock graph.

Syntax:

```
IdentObj getStock52WeekHighLine();
```

Example:

```
setDisplay(getStock52WeekHighLine\(\),true);
setGraphType(73);
setStock52WeekHighValue(40.0);
```



See [getStock52WeekHighLine\(\)](#) on page 1130, [getStock52WeekLowLine\(\)](#) on page 1131, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockHighLine\(\)](#) on page 1134, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStock52WeekLowLine()

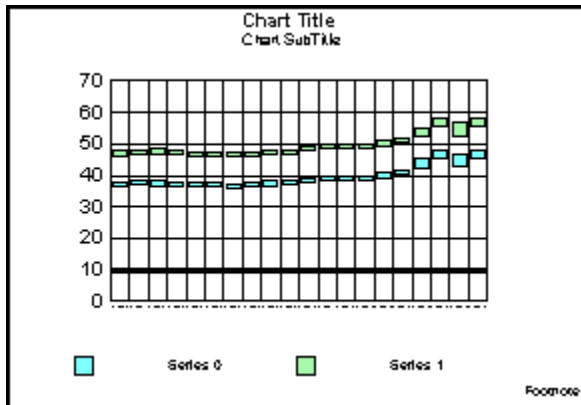
This method returns the object ID of the 52-week low line that may appear in a stock graph.

Syntax:

```
IdentObj getStock52WeekLowLine\(\);
```

Example:

```
setDisplay(getStock52WeekLowLine(), true);
setGraphType(73);
setStock52WeekLowValue(10.0);
```



See [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockHighLine\(\)](#) on page 1134, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockCloseTick()

This method returns the object ID of the close ticks that may appear in a stock graph.

Syntax:

```
IdentObj getStockCloseTick();
IdentObj getStockCloseTick(int s, int g);
```

where:

s

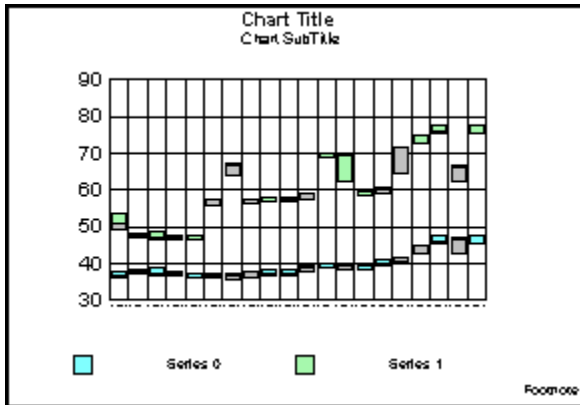
Is an optional series number.

g

Is an optional group number.

Example:

```
setGraphType(76);
setDisplay(getStockCloseTick(), false);
```



See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockFallingRiser\(\)](#) on page 1133, [getStockHighLine\(\)](#) on page 1134, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockFallingRiser()

This method returns the object ID of the falling riser that may appear in a stock graph.

Syntax:

```
IdentObj getStockFallingRiser();
IdentObj getStockFallingRiser(int s, int g);
```

where:

s

Is an optional series number.

g

Is an optional group number.

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockHighLine\(\)](#) on page 1134, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockHighLine()

This method returns the object ID of the high line that may appear in a stock graph.

Syntax:

```
IdentObj getStockHighLine();
```

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockLowerRiser()

This method returns the object ID of the lower risers that may appear in a stock graph.

Syntax:

```
IdentObj getStockLowerRiser();
```

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockLowLine()

This method returns the object ID of the low line that may appear in a stock graph.

Syntax:

```
IdentObj getStockLowLine();
```

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockMALine()

This method returns the object ID of the moving average line that may appear in a stock graph.

Syntax:

```
IdentObj getStockMALine();  
IdentObj getStockMALine(int s, int g);
```

where:*s*

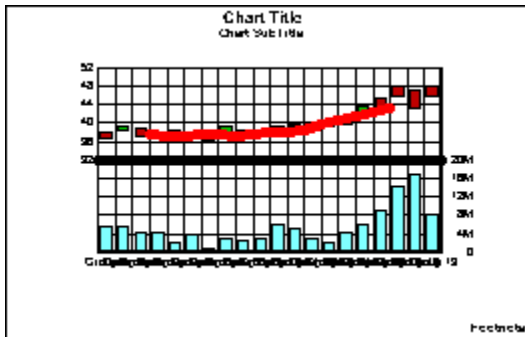
Is an optional series number.

g

Is an optional group number.

Example:

```
setDisplay(getStockMALine\(\), true);
setGraphType(84);
```



See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockOpenTick()

This method returns the object ID of open ticks that may appear in a stock graph.

Syntax:

```
IdentObj getStockOpenTick\(\);
IdentObj getStockOpenTick\(int s, int g\);
```

where:*s*

Is an optional series number.

g

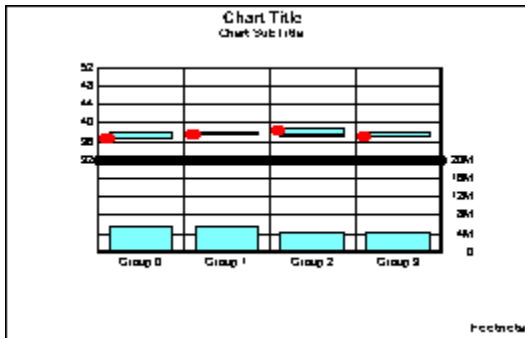
Is an optional group number.

Example:

```

setGraphType(83);
setViewableGroupsStock(4);
setDataScrollerPresenceGroups(0);
setDisplay(getStockOpenTick(), true);
setDisplay(getStockCloseTick(), false);
setLineWidth(getStockOpenTick(), 2);
setBorderColor(getStockOpenTick(), new Color(255, 0, 0));
setLegendDisplay(false);

```



See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockRiser()

This method returns the object ID of risers in a stock graph.

Syntax:

```
IdentObj getStockRiser();
```

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRisingRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockRisingRiser()

This method returns the object ID of rising risers in a stock graph. Specify a series (s) and group (g) number to get the rising riser for a particular series and group.

Syntax:

```
IdentObj getStockRisingRiser();
IdentObj getStockRisingRiser(int s, int g);
```

where:*s*

Is an optional series number.

g

Is an optional group number.

See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockVolumeRiser\(\)](#) on page 1137.

getStockVolumeRiser()

This method returns the object ID of volume risers in a stock graph. Use the group (g) parameter to get the object ID of volume risers for a particular group.

Syntax:

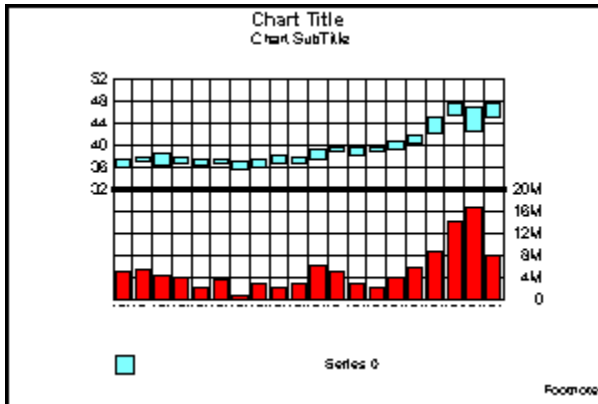
```
IdentObj getStockVolumeRiser();
IdentObj getStockVolumeRiser(int g);
```

where:*g*

Is an optional group number.

Example:

```
setFillColor(getStockVolumeRiser(), new Color(255,0,0));
setGraphType(82);
```



See [getStock52WeekHighLine\(\)](#) on page 1130, [getStockCloseTick\(\)](#) on page 1132, [getStockFallingRiser\(\)](#) on page 1133, [getStockLowerRiser\(\)](#) on page 1134, [getStockLowLine\(\)](#) on page 1134, [getStockMALine\(\)](#) on page 1134, [getStockOpenTick\(\)](#) on page 1135, [getStockRiser\(\)](#) on page 1136, [getStockRisingRiser\(\)](#) on page 1136.

getSubTitle()

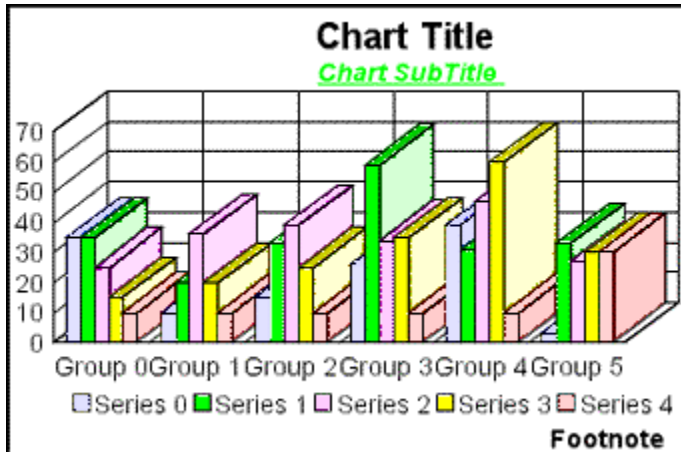
This method returns the object ID of the graph subtitle.

Syntax:

```
IdentObj getSubTitle();
```

Example:

```
setFontStyle(getSubTitle(), 7);
setFillColor(getSubTitle(), new Color (0,255,0));
```



See [getTitle\(\)](#) on page 1144, [getTitleBox\(\)](#) on page 1145.

getSubTitleBox()

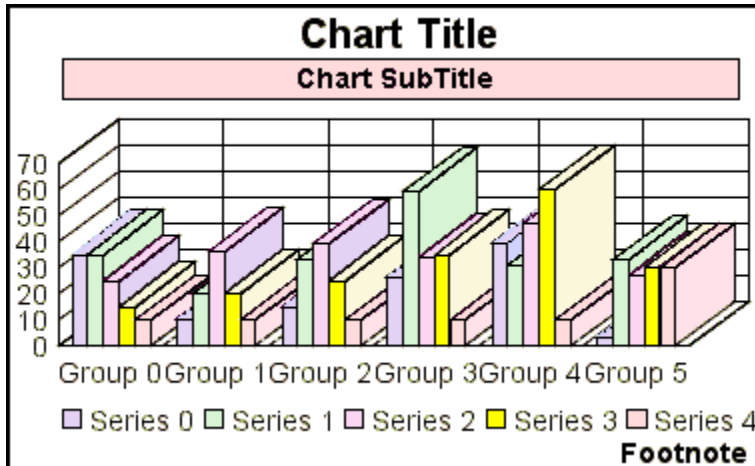
This method returns the object ID of the graph subtitle box.

Syntax:

```
IdentObj getSubTitleBox();
```

Example:

```
setDisplay(getSubTitleBox(), true);  
setFillColor(getSubTitleBox(), new Color(255,0,0));
```



See [getTitle\(\)](#) on page 1144, [getTitleBox\(\)](#) on page 1145.

getThermGaugeArea()

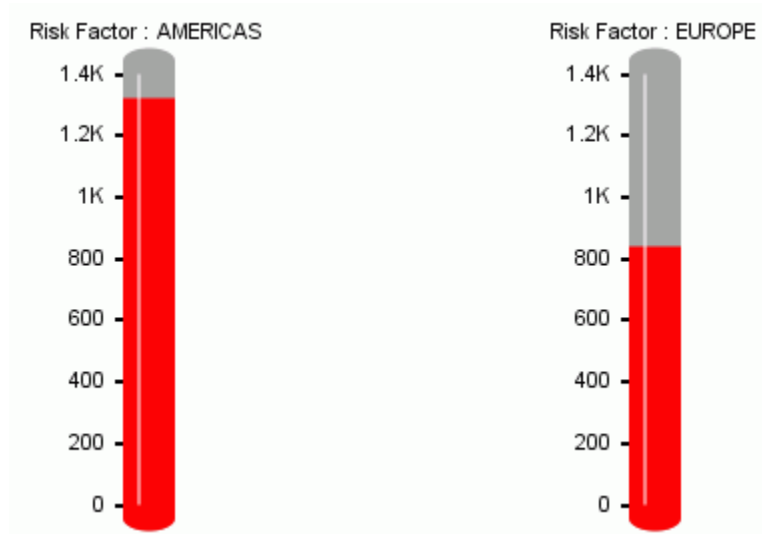
This method returns the object ID of the tube in a thermometer gauge graph.

Syntax:

```
IdentObj getThermGaugeArea()
```


Example:

```
setFillColor(getThermGaugeArea(),new Color(164,164,164));
```

**getThermGaugeBackground()**

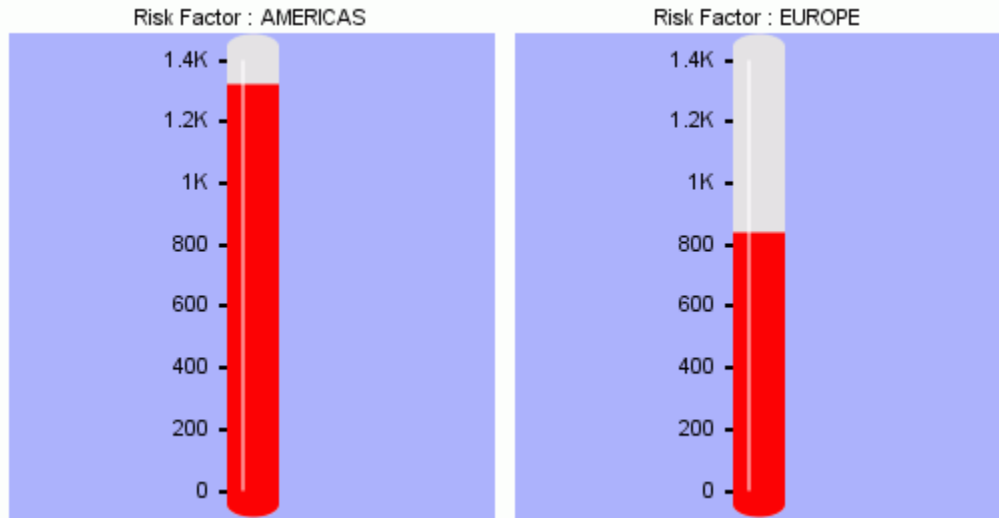
This method returns the object ID of the background area outside the tube in a thermometer gauge graph.

Syntax:

```
IdentObj getThermGaugeBackground()
```

Example:

```
setFillColor(getThermGaugeBackground(),new Color(175,179,252));
```



getThermGaugeNeedle()

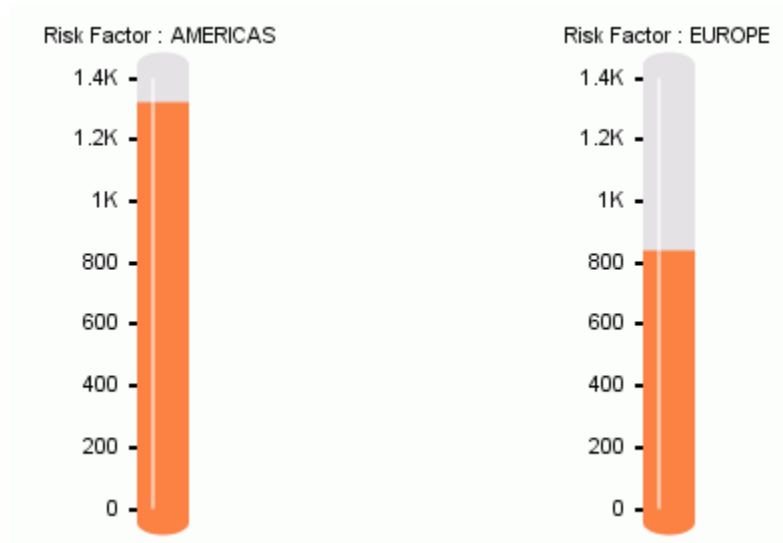
This method returns the object ID of the liquid in a thermometer gauge graph.

Syntax:

```
IdentObj getThermGaugeNeedle()
```

Example:

```
setFillColor(getThermGaugeNeedle(),new Color(255,128,64));
```

**getTimeScaleLevel()**

This method returns the object ID of a level (0...2) on a time scale axis.

Syntax:

```
IdentObj getTimeScaleLevel(level);
```

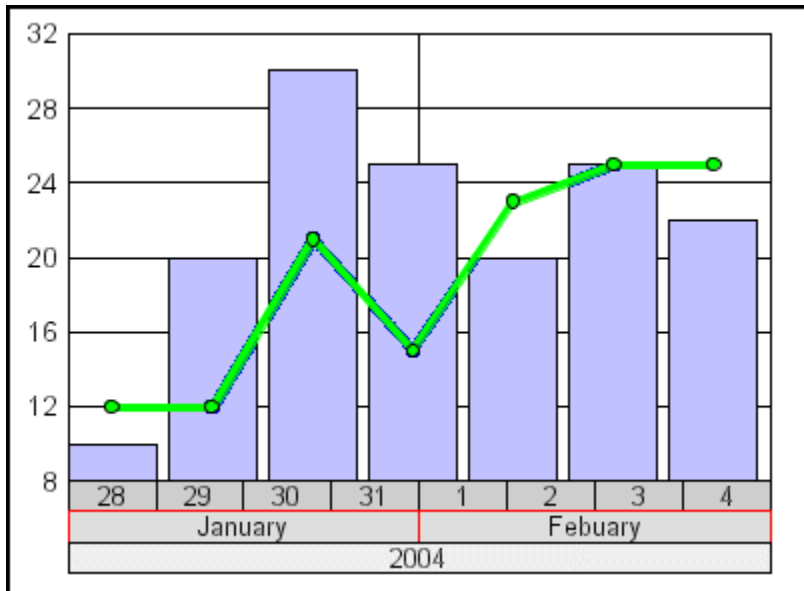
where:

level

Is a time scale axis level 0...2.

Example:

```
setLevelBorderColor(getTimeScaleLevel(1), new Color (255,0,0));
```



See [UseTimeScaleAxis](#) on page 564 for the data that was used to create this chart.

getTitle()

This method returns the object ID of the graph title.

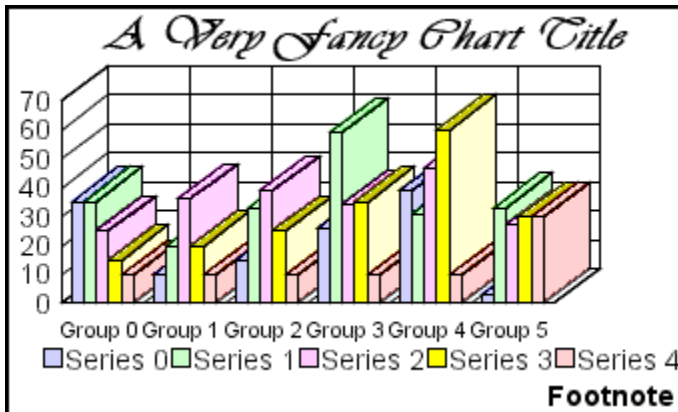
Syntax:

```
IdentObj getTitle();
```

Example:

```
setAutofit(getSubtitle(),false);
setAutofit(getTitle(),false);
setFontName(getTitle(),"Vivaldi");
setFontSize(getTitle(),48);
setFontSizeAbsolute(getTitle(),true);
setFontStyle(getTitle(),3);
setSubtitleDisplay(false);
setTitleString("A Very Fancy Chart Title");
setTextAntialiasing(true);
```

```
setRect(getTitle(), new Rectangle(-13471, 11992, 25806, 1539));
```



See [getSubTitle\(\)](#) on page 1138, [getSubTitleBox\(\)](#) on page 1139.

getTitleBox()

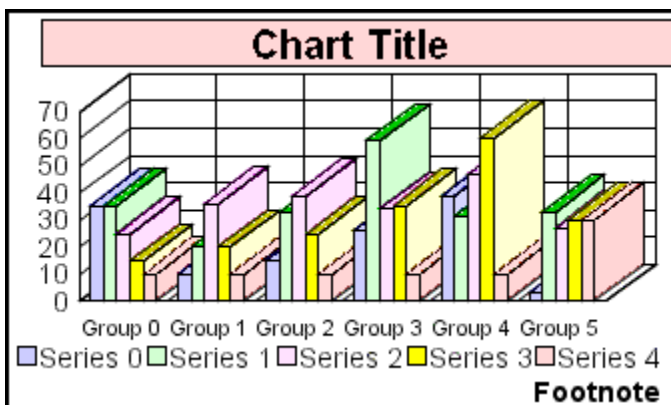
This method returns the object ID of the graph title box.

Syntax:

```
IdentObj getTitleBox()
```

Example:

```
setDisplay(getTitleBox(), true);
setFillColor(getTitleBox(), new Color(255,0,0));
setSubTitleString("");
```



See [getSubTitle\(\)](#) on page 1138, [getSubTitleBox\(\)](#) on page 1139, [getTitle\(\)](#) on page 1144.

getWaterFallLine()

This method returns the object ID of the line that connects risers in a waterfall graph.

Syntax:

```
IdentObj getWaterfallLine();
```

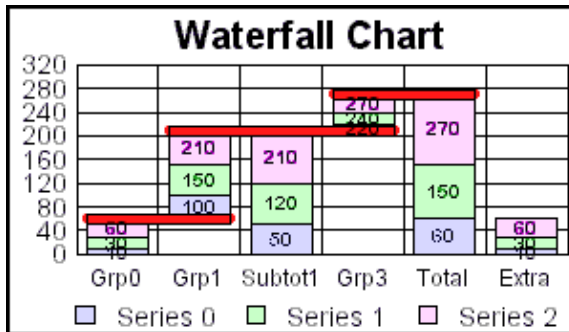
Example:

```
setGraphType(100);
setDepthRadius(0);
setDepthAngle(0);
setData(0, 0, 10);
setData(1, 0, 20);
setData(2, 0, 30);
setGroupLabel(0, "Grp0");
setData(0, 1, 40);
setData(1, 1, 50);
setData(2, 1, 60);
setGroupLabel(1, "Grp1");
setData(0, 2, 0);
setData(1, 2, 0);
setData(2, 2, 0);
setWaterfallGroupMode(2, 1);
setGroupLabel(getGroup(2), "Subtot1");
setData(0, 3, 10);
setData(1, 3, 20);
setData(2, 3, 30);
setGroupLabel(3, "Grp3");
setData(0, 4, 0);
setData(1, 4, 0);
setData(2, 4, 0);
setWaterfallGroupMode(4, 2);
setGroupLabel(getGroup(4), "Total");
setData(0, 5, 10);
setData(1, 5, 20);
setData(2, 5, 30);
setWaterfallGroupMode(5, 3);
setGroupLabel(getGroup(5), "Extra");
setStackedDataValueSum(true);
setDataTextDisplay(true);
```

```

setTextJustHoriz(getDataText(),1);
setFontSizeAbsolute(getDataText(),true);
setFontSize(getDataText(),10);
setFontSizeAbsolute(getO1Label(0),true);
setFontSize(getO1Label(0),12);
setDataRangeToExtent();
setSeriesLabel(0, "Series 0");
setSeriesLabel(1, "Series 1");
setSeriesLabel(2, "Series 2");
setTitleString("Waterfall Chart");
setLineWidth(getWaterfallLine(), 5);
setFillColor(getWaterfallLine(), new Color(255,0,0));

```



getX1Axis()

This method returns the object ID of the X1-axis.

Syntax:

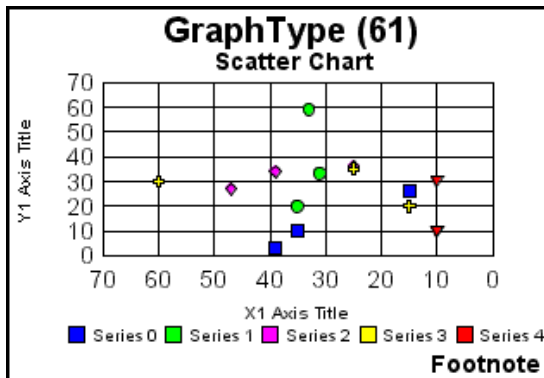
```
IdentObj getX1Axis();
```

Example:

```

setGraphType(61); setDepthAngle(0);
setDepthRadius(0);
setX1TitleString("X1 Axis Title");
setY1TitleString("Y1 Axis Title");
setTitleString("GraphType (61)");
setSubtitleString("Scatter Chart");
setAxisDescending(getX1Axis(),true);

```



See [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152, [getX1Title\(\)](#) on page 1153, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

getX1AxisLine()

This method returns the object ID of the X1-axis base line.

Syntax:

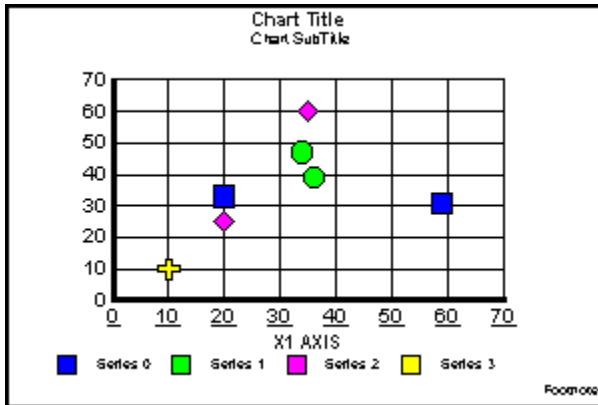
```
IdentObj getX1AxisLine();
```


Example:

```

setDisplay(getX1ZeroLine(),true);
setGraphType(61);
setLineWidth(getX1ZeroLine(),5);
setMarkerSize(90);
setX1TitleString("X1 AXIS");
setDisplay(getX1Title(),true);
setFontStyle(getX1Label(),4);
setDisplay(getX1AxisLine(),true);
setLineWidth(getX1AxisLine(),5);
setDataRange(1,1,4,4);

```



See [getX1Axis\(\)](#) on page 1147, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152, [getX1Title\(\)](#) on page 1153, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

getX1Label()

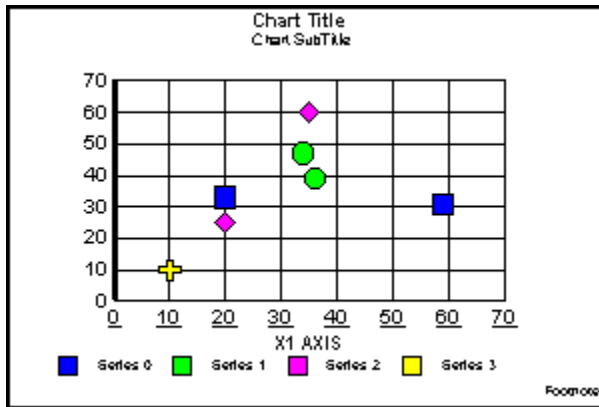
This method returns the object ID of labels on the X1-axis.

Syntax:

```
IdentObj getX1Label();
```

Example:

```
setX1ZeroLineDisplay(true);
setGraphType(61);
setLineWidth(getX1ZeroLine(),5);
setMarkerSize(90);
setX1TitleString("X1 AXIS");
setX1TitleDisplay(true);
setFontStyle(getX1Label(),4);
setDataRange(1,1,4,4);
```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152, [getX1Title\(\)](#) on page 1153, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getX1MajorGrid()

This method returns the object ID of the X1-axis major grid lines.

Syntax:

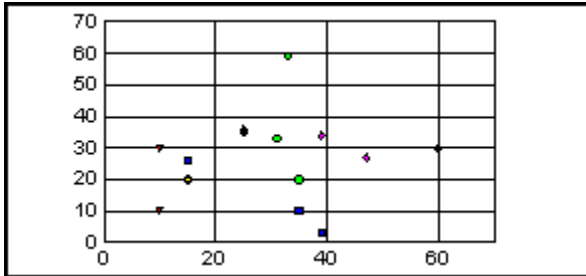
```
IdentObj getX1MajorGrid\(\);
```

Example:

```

setGraphType(61);
setGridStep(getX1MajorGrid(),20.0);
setGridStep(getY1MajorGrid(),10.0);
setGridStepAuto(getX1MinorGrid(),true);
setGridStepAuto(getX1MajorGrid(),false);

```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MinorGrid\(\)](#) on page 1152, [getX1Title\(\)](#) on page 1153, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

getX1MajorTick()

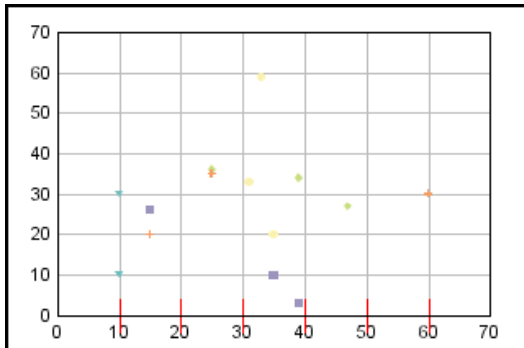
This method returns the object ID of the X1-axis major tick marks.

Syntax:

```
IdentObj getX1MajorTick\(\);
```

Example:

```
setGraphType(61);
setDepthRadius(0);
setDepthAngle(0);
setX1MajorTickDisplay(true);
setTickLength(getX1MajorTick(), 2000);
setBorderColor(getX1MajorTick(), new Color (255,0,0));
```



getX1MinorGrid()

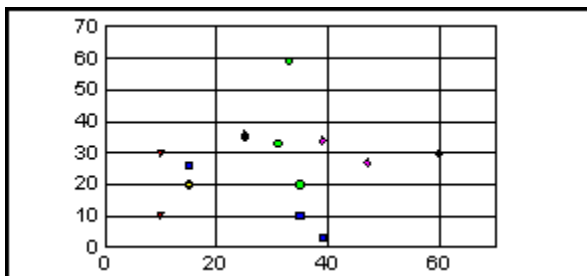
This method returns the object ID of the X1-axis minor grid lines.

Syntax:

```
IdentObj getX1MinorGrid();
```

Example:

```
setGraphType(61);
setGridStep(getX1MajorGrid(),20.0);
setGridStep(getY1MajorGrid(),10.0);
setGridStepAuto(getX1MinorGrid(),true);
setGridStepAuto(getX1MajorGrid(),false);
```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1Title\(\)](#) on page 1153, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

getX1MinorTick()

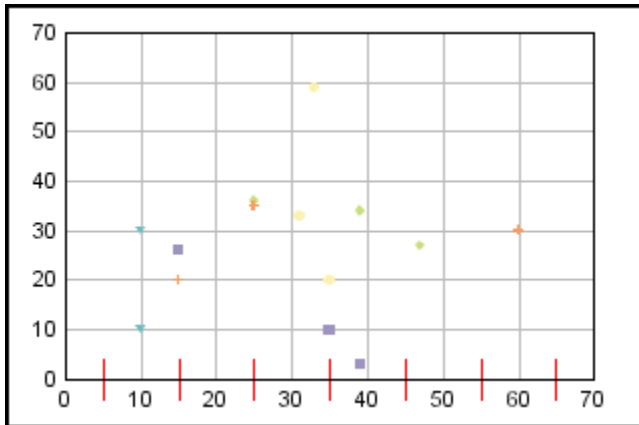
This method returns the object ID of the X1-axis minor tick marks.

Syntax:

```
IdentObj getX1MinorTick();
```

Example:

```
setDepthRadius(0);
setDepthAngle(0);
setGraphType(61);
setX1MinorTickDisplay(true);
setTickLength(getX1MinorTick(), 2000);
setBorderColor(getX1MinorTick(), new Color (255,0,0));
```



getX1Title()

This method returns the object ID of the X1-axis title.

Syntax:

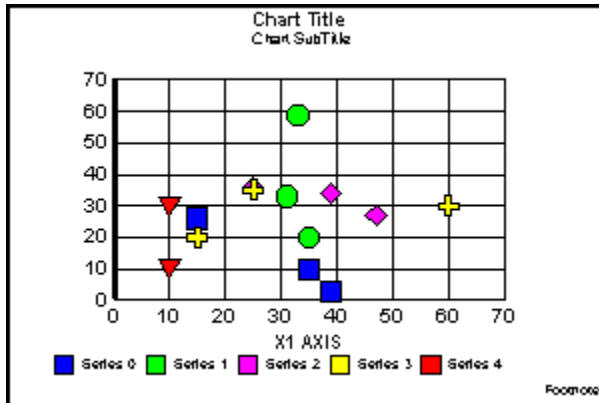
```
IdentObj getX1Title();
```

Example:

```

setDisplay(getX1ZeroLine(),true);
setGraphType(61);
setLineWidth(getX1ZeroLine(),5);
setMarkerSize(90);
setX1TitleString("X1 AXIS");
setDisplay(getX1Title(),true);

```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152, [getX1TitleBox\(\)](#) on page 1154, [getX1ZeroLine\(\)](#) on page 1155.

getX1TitleBox()

This method returns the object ID of the title box on a graph with an X1-axis.

Syntax:

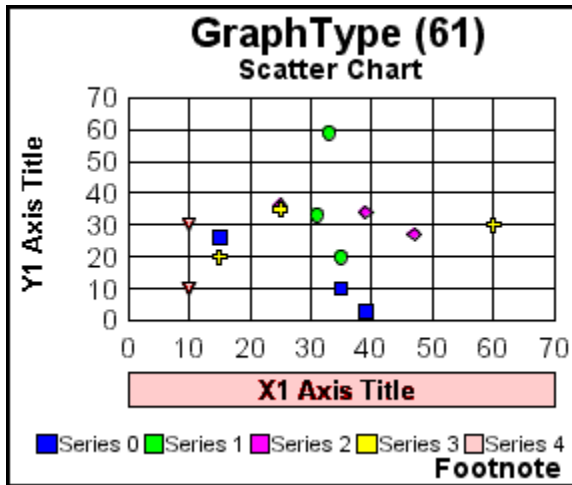
```
IdentObj getX1TitleBox();
```

Example:

```

setGraphType(61); setDepthAngle(0);
setDepthRadius(0);
setX1TitleString("X1 Axis Title");
setY1TitleString("Y1 Axis Title");
setTitleString("GraphType (61)");
setSubtitleString("Scatter Chart");
setFillColor(getX1TitleBox(), new Color(255,0,0));
setDisplay(getX1TitleBox(),true);

```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152, [getX1Title\(\)](#) on page 1153, [getX1ZeroLine\(\)](#) on page 1155.

getX1ZeroLine()

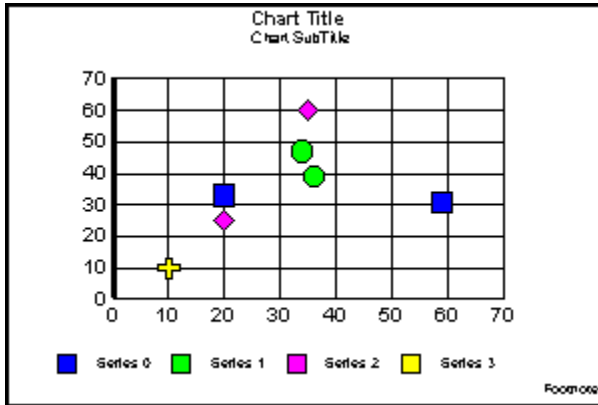
This method returns the object ID of the X1-axis zero line.

Syntax:

```
IdentObj getX1ZeroLine();
```

Example:

```
setDisplay(getX1ZeroLine(),true);
setGraphType(61);
setLineWidth(getX1ZeroLine(),5);
setMarkerSize(90);
```



See [getX1Axis\(\)](#) on page 1147, [getX1AxisLine\(\)](#) on page 1148, [getX1Label\(\)](#) on page 1149, [getX1MajorGrid\(\)](#) on page 1150, [getX1MinorGrid\(\)](#) on page 1152.

getY1Axis()

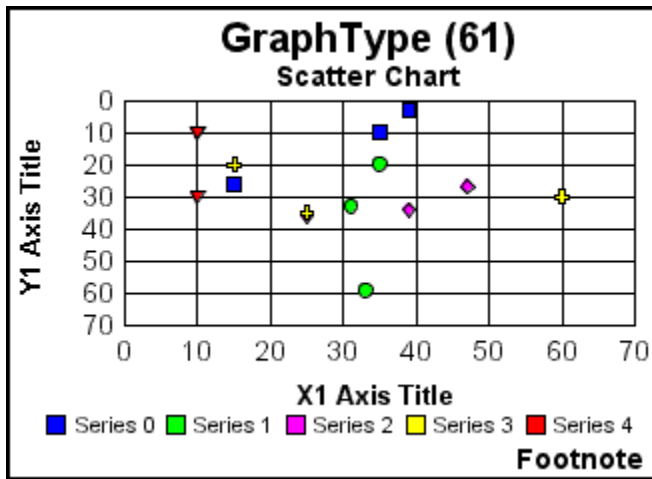
This method returns the object ID of the Y1-axis.

Syntax:

```
IdentObj getY1Axis();
```

Example:

```
setGraphType(61);
setDepthAngle(0);
setDepthRadius(0);
setX1TitleString("X1 Axis Title");
setY1TitleString("Y1 Axis Title");
setTitleString("GraphType (61)");
setSubtitleString("Scatter Chart");
setAxisDescending(getY1Axis(),true);
```

See [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

getY1AxisLine()

This method returns the object ID of the Y1-axis base line.

Syntax:

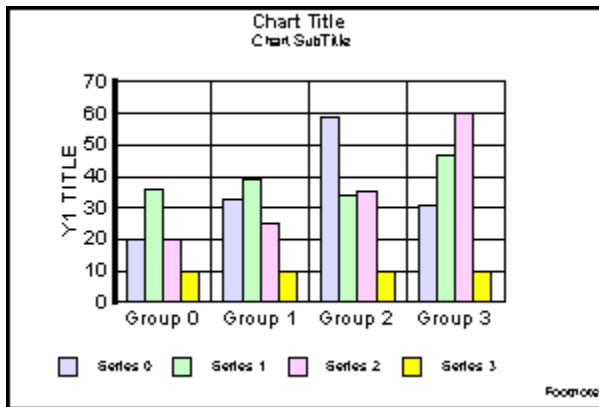
```
IdentObj getY1AxisLine();
```

Example:

```

setDataRange(1,1,4,4);
setDepthAngle(0);
setDepthRadius(0);
setY1TitleString("Y1 TITLE");
setDisplay(getY1Title(),true);
setLineWidth(getY1AxisLine(),5);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);

```



See [getY1Axis\(\)](#) on page 1156, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

getY1FrontZeroLine()

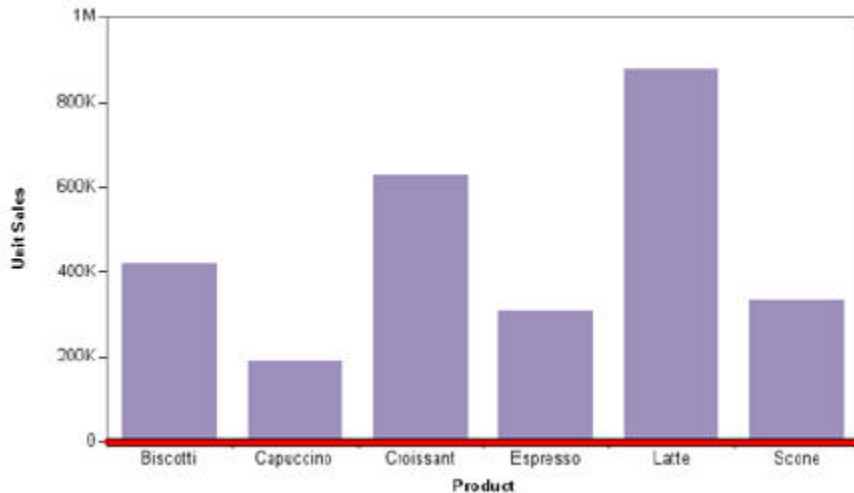
This method returns the object ID of the Y1-axis front zero line.

Syntax:

```
IdentObj getY1FrontZeroLine();
```

Example:

```
setDisplay(getY1FrontZeroLine(),true);
setDisplay(getY1ZeroLine(),true);
setLineWidth(getY1ZeroLine(),5);
setFillColor(getY1ZeroLine(),new Color(255,0,0));
```



Note: Y1ZeroLine must be displayed (see [getY1ZeroLine\(\)](#) on page 1166) in order for Y1FrontZeroLine to be drawn on top (in front) of bars.

getY1Label()

This method returns the object ID of labels on the Y1-axis.

Syntax:

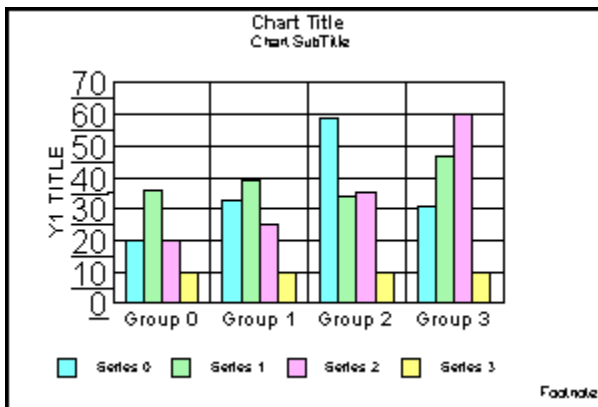
```
IdentObj getY1Label();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setDepthAngle(0);
setDepthRadius(0);
setY1TitleString("Y1 TITLE");
setDisplay(getY1Title(),true);
setAutofit(getY1Label(),false);
setFontSizeVC(getY1Label(),2500);
setFontStyle(getY1Label(),4);

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getY1MajorGrid()

This method returns the object ID of the Y1-axis major grid lines.

Syntax:

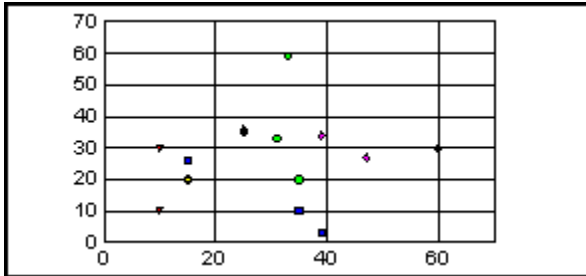
```
IdentObj getY1MajorGrid();
```

Example:

```

setGraphType(61);
setGridStep(getX1MajorGrid(),20.0);
setGridStep(getY1MajorGrid(),10.0);
setGridStepAuto(getX1MinorGrid(),true);
setGridStepAuto(getX1MajorGrid(),false);

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

getY1MajorTick()

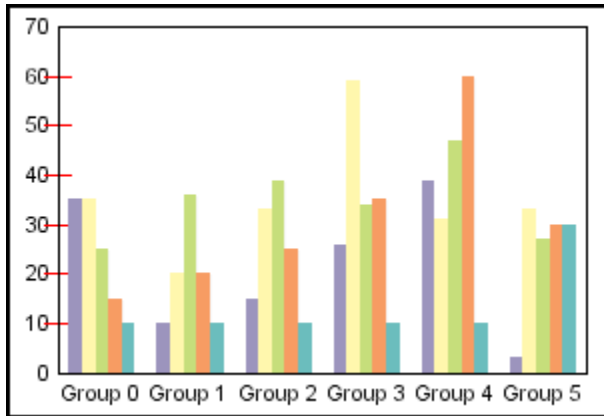
This method returns the object ID of major tick marks that may be drawn on the Y1 axis. These tick marks are enabled/disabled with the Y1MajorTickDisplay property.

Syntax:

```
IdentObj getY1MajorTick();
```

Example:

```
setDepthRadius(0);  
setDepthAngle(0);  
setY1MajorGridDisplay(false);  
setY1MajorTickDisplay(true);  
setTickLength(getY1MajorTick(), 1000);  
setBorderColor(getY1MajorTick(), new Color (255,0,0));
```



getY1MinorGrid()

This method returns the object ID of the Y1-axis minor grid lines.

Syntax:

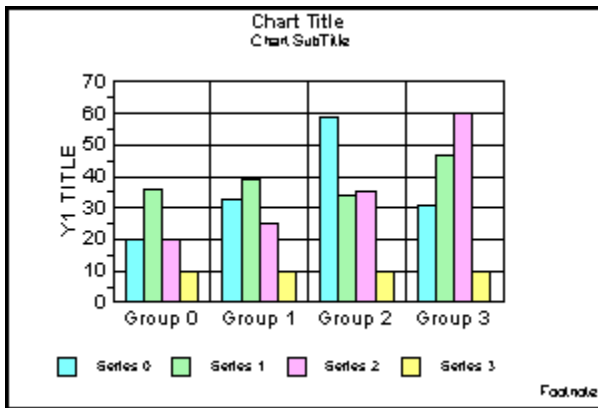
```
IdentObj getY1MinorGrid();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setDepthAngle(0);
setDepthRadius(0);
setY1TitleString("Y1 TITLE");
setDisplay(getY1Title(),true);
setDisplay(getY1MinorGrid(),true);
setGridStyle(getY1MinorGrid(),3);

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

getY1MinorTick()

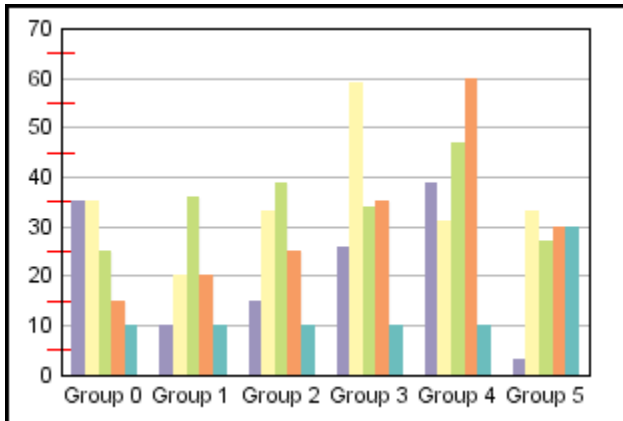
This method returns the object ID of minor tick marks that may be drawn on the Y1 axis. These tick marks are enabled/disabled with the Y1MinorTickDisplay property.

Syntax:

```
IdentObj getY1MinorTick();
```

Example:

```
setDepthRadius(0);  
setDepthAngle(0);  
setY1MinorGridDisplay(false);  
setY1MinorTickDisplay(true);  
setTickLength(getY1MinorTick(), 1000);  
setBorderColor(getY1MinorTick(), new Color (255,0,0));
```



getY1Title()

This method returns the object ID of the Y1-axis title.

Syntax:

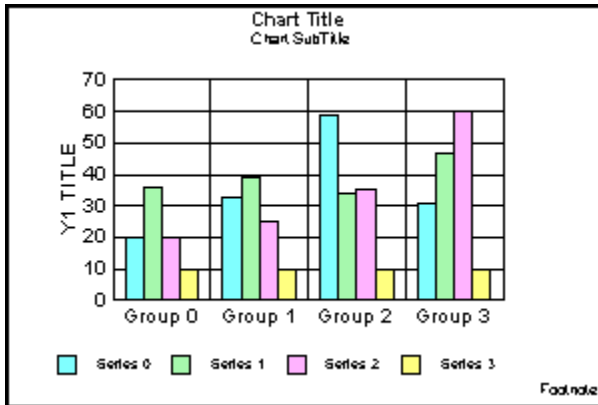
```
IdentObj getY1Title();
```


Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setDepthAngle(0);
setDepthRadius(0);
setY1TitleString("Y1 TITLE");
setDisplay(getY1Title\(\),true);

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1TitleBox\(\)](#) on page 1165, [getY1ZeroLine\(\)](#) on page 1166.

getY1TitleBox()

This method returns the object ID of the title box on a graph with an Y1-axis.

Syntax:

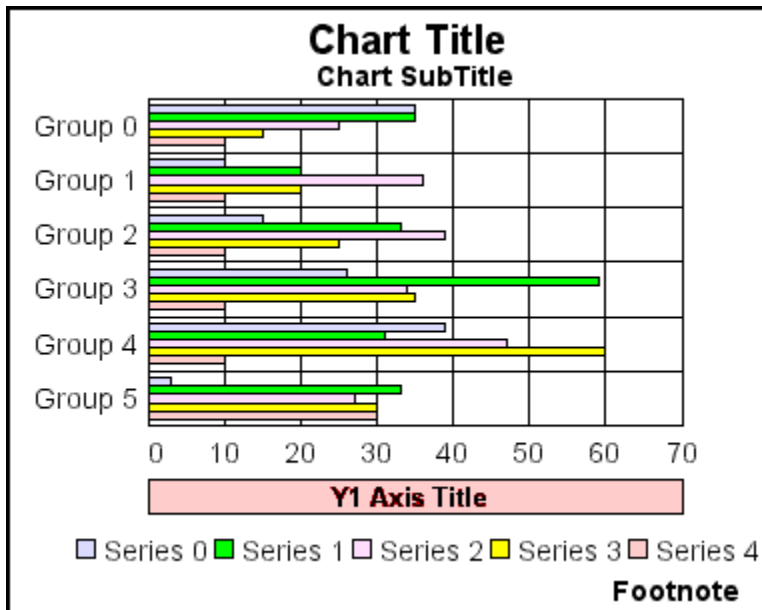
```
IdentObj getY1TitleBox\(\);
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setGraphType(24);
setY1TitleString("Y1 Axis Title");
setDisplay(getY1TitleBox(), true);
setFillColor(getY1TitleBox(), new Color (255,0,0));

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1ZeroLine\(\)](#) on page 1166.

getY1ZeroLine()

This method returns the object ID of the Y1-axis zero line.

Syntax:

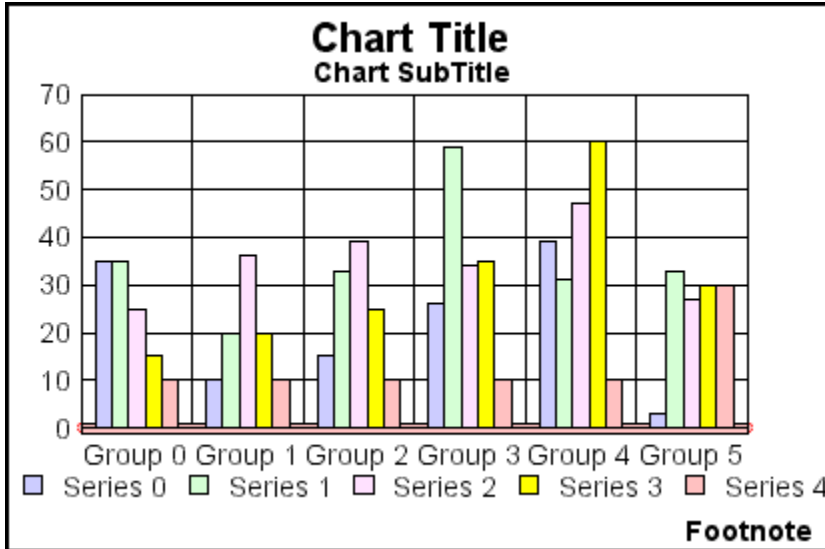
```
IdentObj getY1ZeroLine();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY1ZeroLine(), true);
setLineWidth(getY1ZeroLine(), 5);
setFillColor(getY1ZeroLine(), new Color (255,0,0));

```



See [getY1Axis\(\)](#) on page 1156, [getY1AxisLine\(\)](#) on page 1157, [getY1Label\(\)](#) on page 1159, [getY1MajorGrid\(\)](#) on page 1160, [getY1MinorGrid\(\)](#) on page 1162, [getY1Title\(\)](#) on page 1164, [getY1TitleBox\(\)](#) on page 1165.

getY2Axis()

This method returns the object ID of the Y2-axis.

Syntax:

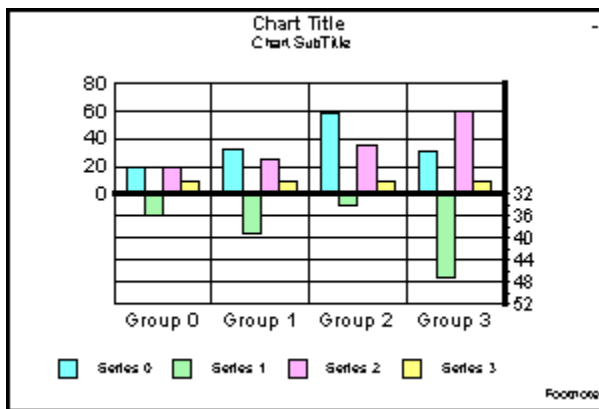
```
IdentObj getY2Axis();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY2AxisLine(),true);
setLineWidth(getY2AxisLine(),5);
setAxisDescending(getY2Axis(),true);

```



See [getY2AxisLine\(\)](#) on page 1168, [getY2Label\(\)](#) on page 1170, [getY2MajorGrid\(\)](#) on page 1171, [getY2MinorGrid\(\)](#) on page 1173, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

getY2AxisLine()

This method returns the object ID of the Y2-axis base line.

Syntax:

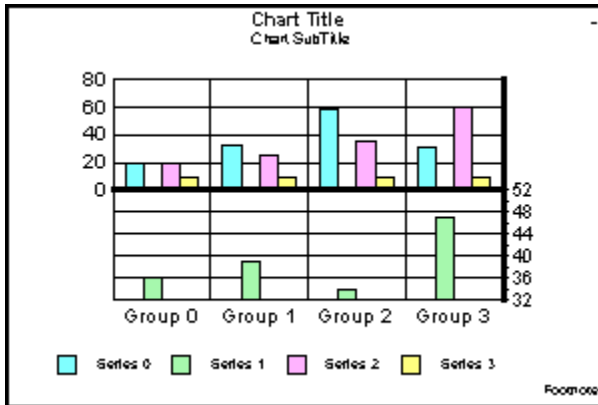
```
IdentObj getY2AxisLine();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY2AxisLine\(\),true);
setLineWidth(getY2AxisLine\(\),5);

```



See [getY2Axis\(\)](#) on page 1167, [getY2Label\(\)](#) on page 1170, [getY2MajorGrid\(\)](#) on page 1171, [getY2MinorGrid\(\)](#) on page 1173, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

getY2FrontZeroLine()

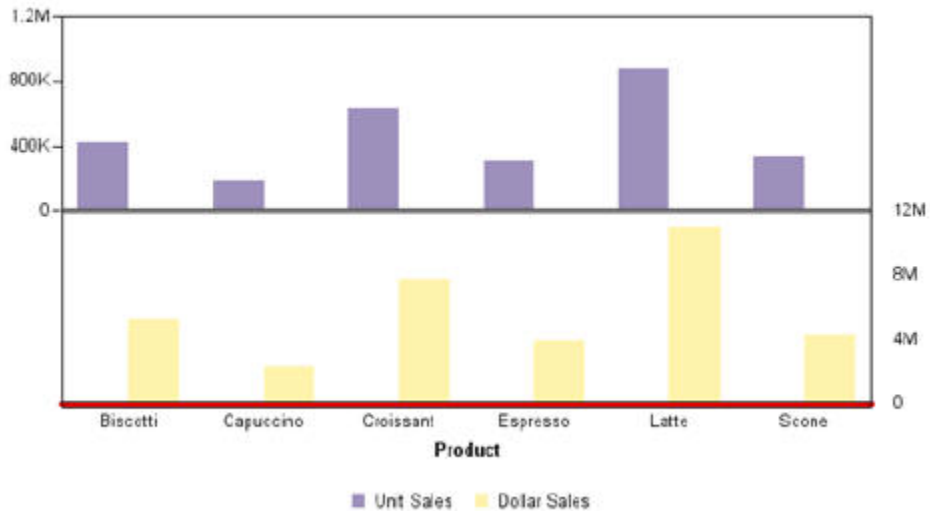
This method returns the object ID of the Y2-axis front zero line.

Syntax:

```
IdentObj getY2FrontZeroLine\(\);
```

Example:

```
setDisplay(getY2FrontZeroLine(),true);
setDisplay(getY2ZeroLine(),true);
setLineWidth(getY2ZeroLine(),5);
setFillColor(getY2ZeroLine(),new Color(255,0,0));
```



Note: Y2ZeroLine must be displayed (see [getY2ZeroLine\(\)](#) on page 1177) in order for Y2FrontZeroLine to be drawn on top (in front) of bars.

getY2Label()

This method returns the object ID of labels on the Y2-axis.

Syntax:

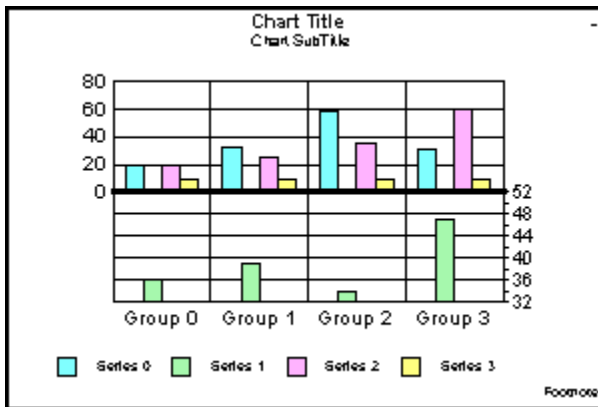
```
IdentObj getY2Label();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setAutofit(getY2Label(),false);
setFontSizeVC(getY2Label(),1500);

```



See [getY2Axis\(\)](#) on page 1167, [getY2AxisLine\(\)](#) on page 1168, [getY2MajorGrid\(\)](#) on page 1171, [getY2MinorGrid\(\)](#) on page 1173, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getY2MajorGrid()

This method returns the object ID of the Y2-axis major grid lines.

Syntax:

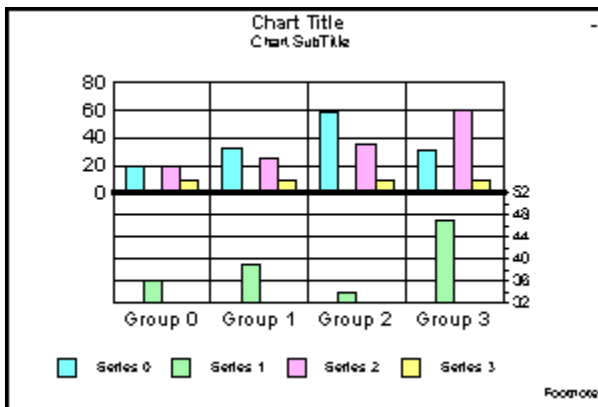
```
IdentObj getY2MajorGrid();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setY2MinorGridDisplay(true);
setY2MinorGridStyle(3);
setDisplay(getY2MajorGrid(),true);
setGridStyle(getY2MajorGrid(),1);

```



See [getY2Axis\(\)](#) on page 1167, [getY2AxisLine\(\)](#) on page 1168, [getY2Label\(\)](#) on page 1170, [getY2MinorGrid\(\)](#) on page 1173, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

getY2MajorTick()

This method returns the object ID of major tick marks that may be drawn on the Y2 axis. These tick marks are enabled/disabled with the Y2MajorTickDisplay property.

Syntax:

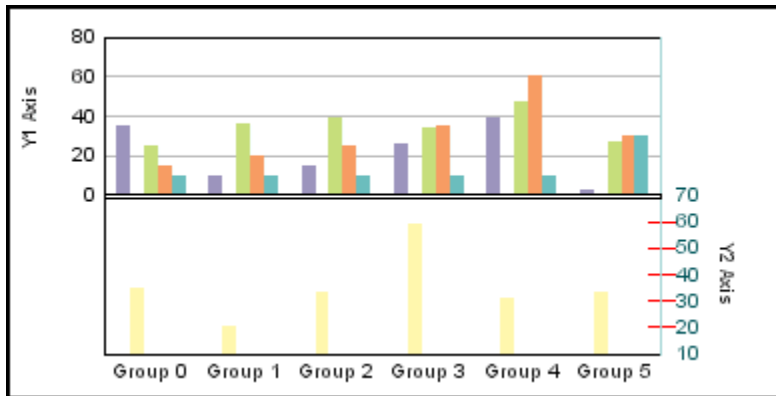
```
IdentObj getY2MajorTick();
```


Example:

```

setDepthAngle(0);
setDepthRadius(0);
setY2MajorGridDisplay(false);
setY2MajorTickDisplay(true);
setTickLength(getY2MajorTick(), 1000);
setBorderColor(getY2MajorTick(), new Color (255,0,0));
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");

```

**getY2MinorGrid()**

This method returns the object ID of the Y2-axis minor grid lines.

Syntax:

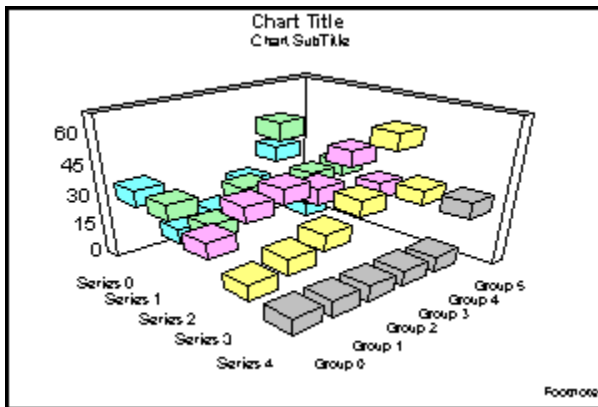
```
IdentObj getY2MinorGrid();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY2MinorGrid\(\),true);
setGridStyle(getY2MinorGrid\(\),2);

```



See [getY2Axis\(\)](#) on page 1167, [getY2AxisLine\(\)](#) on page 1168, [getY2Label\(\)](#) on page 1170, [getY2MajorGrid\(\)](#) on page 1171, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

getY2MinorTick()

This method returns the object ID of minor tick marks that may be drawn on the Y2 axis. These tick marks are enabled/disabled with the Y2MinorTickDisplay property.

Syntax:

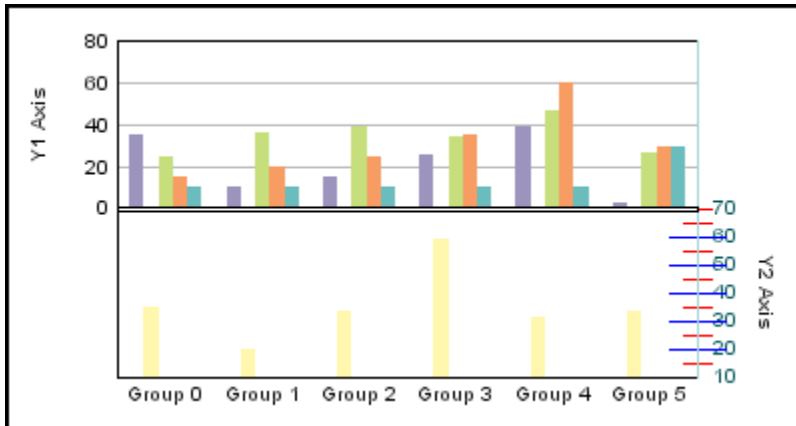
```
IdentObj getY2MinorTick\(\);
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setY2MajorGridDisplay(false);
setY2MinorTickDisplay(true);
setY2MajorTickDisplay(true);
setTickLength(getY2MinorTick(), 1000);
setBorderColor(getY2MinorTick(), new Color (255,0,0));
setBorderColor(getY2MajorTick(), new Color(0,0,255));
setTickLength(getY2MajorTick(), 2000);
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");

```

**getY2Title()**

This method returns the object ID of the Y2-axis title.

Syntax:

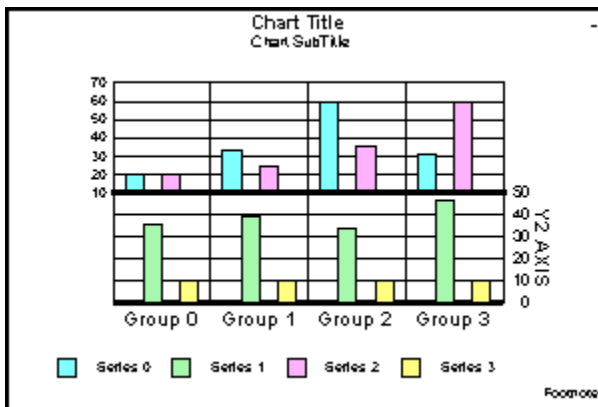
```
IdentObj getY2Title();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setDisplay(getY2Title(),true);
setDisplay(getY2ZeroLine(),true);
setDualAxisSplitPosition(50);
setFontStyle(getY2Title(),2);
setGraphType(21);
setTextJustVert(getY2Title(),0);
setTextString(getY2Title(),"Y2 AXIS");

```



See [getY2Axis\(\)](#) on page 1167, [getY2AxisLine\(\)](#) on page 1168, [getY2Label\(\)](#) on page 1170, [getY2MajorGrid\(\)](#) on page 1171, [getY2MinorGrid\(\)](#) on page 1173, [getY2TitleBox\(\)](#) on page 1176, [getY2ZeroLine\(\)](#) on page 1177.

getY2TitleBox()

This method returns the object ID of the Y2-axis title box.

Syntax:

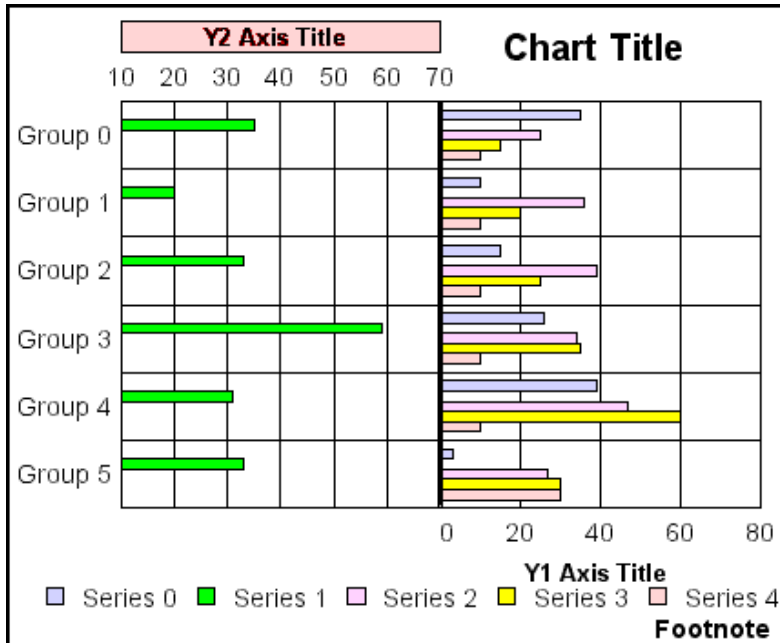
```
IdentObj getY2TitleBox()
```

Example:

```

setGraphType(28); setDepthAngle(0);
setDepthRadius(0);
setY1TitleString("Y1 Axis Title");
setY2TitleString("Y2 Axis Title");
setDisplay(getY2TitleBox(), true);
setFillColor(getY2TitleBox(), new Color (255,0,0));
setSubtitleDisplay(false);

```

**getY2ZeroLine()**

This method returns the object ID of the Y2-axis zero line.

Syntax:

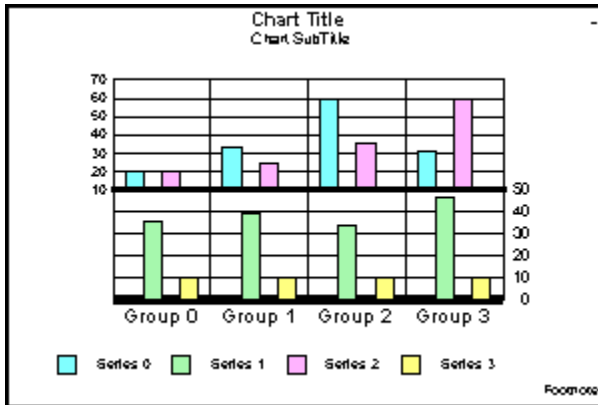
```
IdentObj getY2ZeroLine();
```

Example:

```

setDataRange(1,1,4,4);
setViewableSeries(4);
setDataScrollerPresenceSeries(0);
setViewableGroups(4);
setDataScrollerPresenceGroups(0);
setGraphType(21);
setDepthAngle(0);
setDepthRadius(0);
setDisplay(getY2ZeroLine\(\),true);
setLineWidth(getY2ZeroLine\(\),8);

```



See [getY2Axis\(\)](#) on page 1167, [getY2AxisLine\(\)](#) on page 1168, [getY2Label\(\)](#) on page 1170, [getY2MajorGrid\(\)](#) on page 1171, [getY2MinorGrid\(\)](#) on page 1173, [getY2Title\(\)](#) on page 1175, [getY2TitleBox\(\)](#) on page 1176.

getY3Axis()

This method returns the object ID of the Y3-axis.

Syntax:

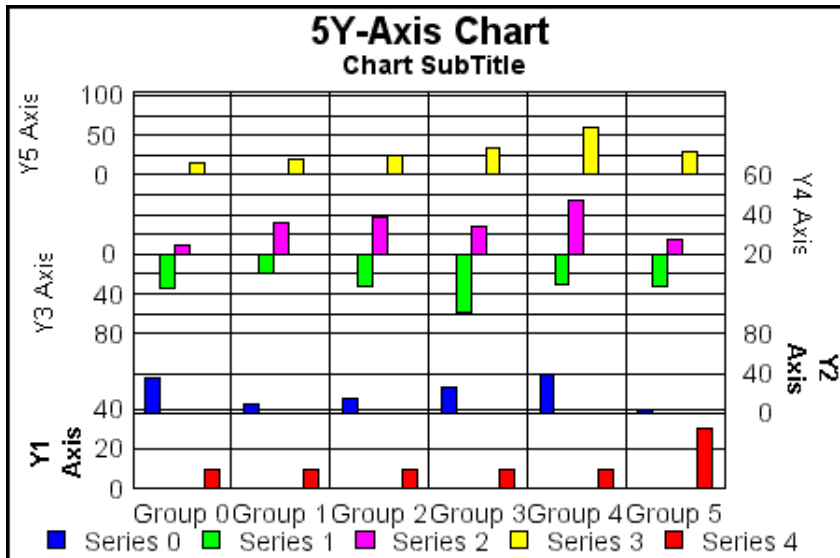
```
IdentObj getY3Axis\(\);
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setAxisDescending(getY3Axis(), true);

```



See [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

getY3AxisLine()

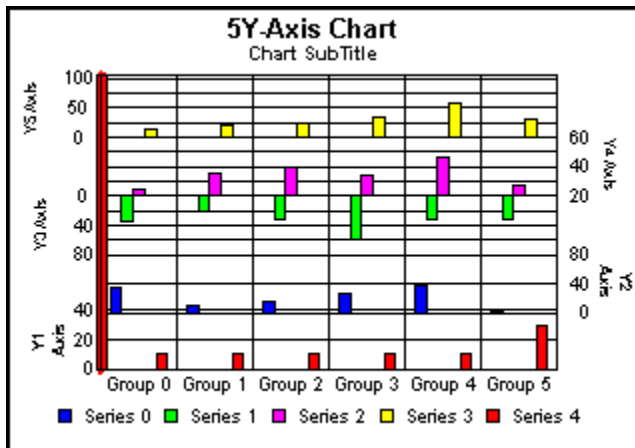
This method returns the object ID of the Y3-axis base line.

Syntax:

```
IdentObj getY3AxisLine()
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setAxisDescending(getY3Axis(), true);
setLineWidth(getY3AxisLine(), 5);
setFillColor(getY3AxisLine(), new Color(255,0,0));
```



See [getY3Axis\(\)](#) on page 1178, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

getY3FrontZeroLine()

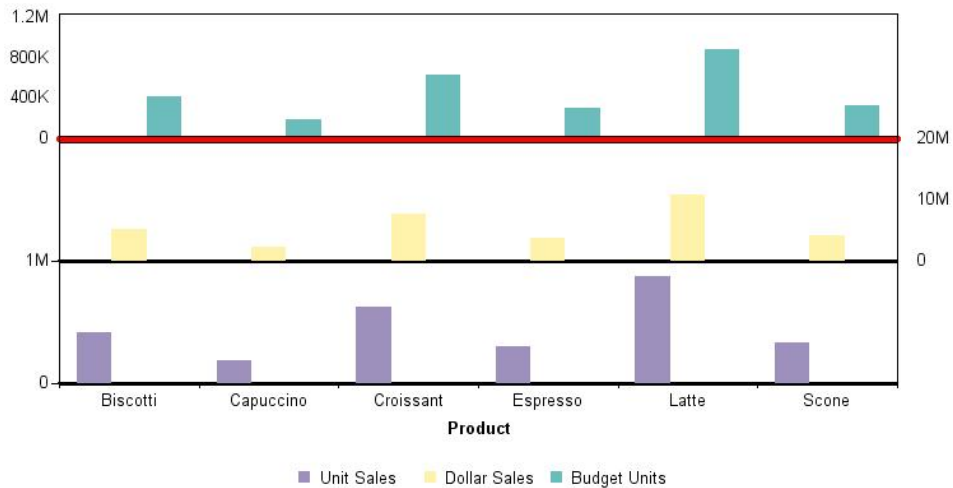
This method returns the object ID of the Y3-axis front zero line.

Syntax:

```
IdentObj getY3FrontZeroLine();
```

Example:

```
setDisplay(getY3ZeroLine(),true);
setDisplay(getY3FrontZeroLine(),true);
setLineWidth(getY3ZeroLine(),5);
setFillColor(getY3ZeroLine(),new Color(255,0,0));
```



Note: Y3ZeroLine must be displayed (see [getY3ZeroLine\(\)](#) on page 1189) in order for Y3FrontZeroLine to be drawn on top (in front) of bars.

getY3Label()

This method returns the object ID of labels on the Y3-axis.

Syntax:

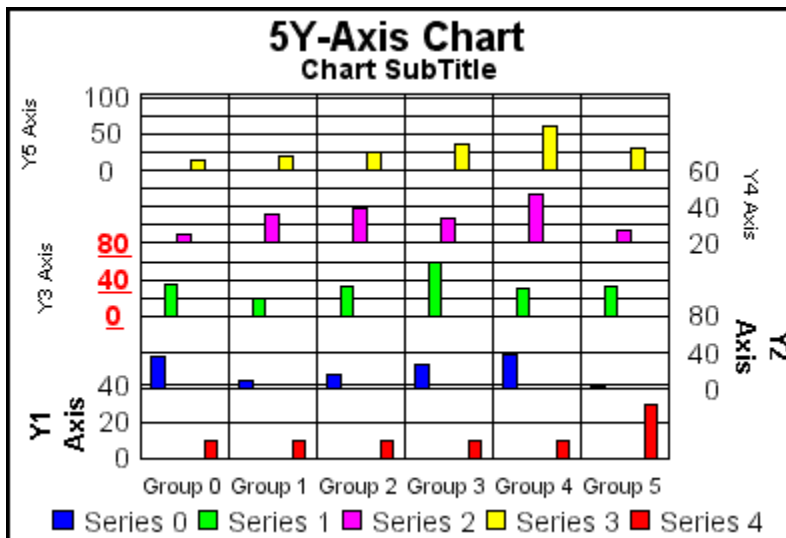
```
IdentObj getY3Label()
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY3Label(), 6);
setFillColor(getY3Label(), new Color(255,0,0));

```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getY3MajorGrid()

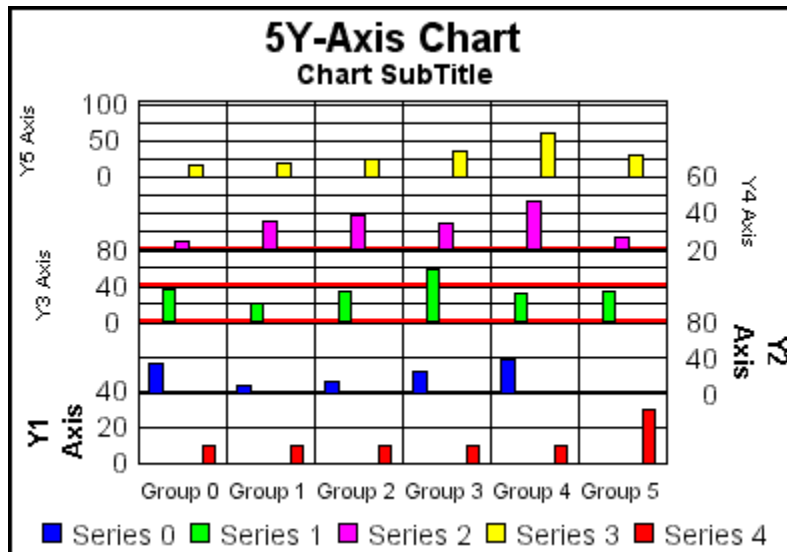
This method returns the object ID of the Y3-axis major grid lines.

Syntax:

```
IdentObj getY3MajorGrid();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY3MajorGrid(), 2);
setFillColor(getY3MajorGrid(), new Color(255,0,0));
```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

getY3MajorTick()

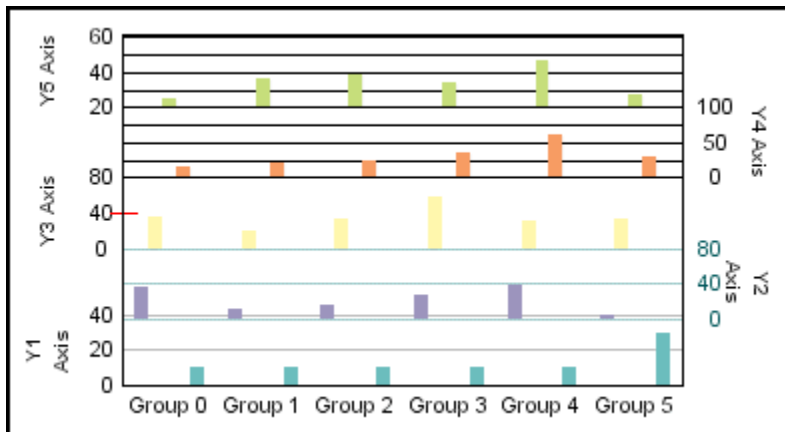
This method returns the object ID of major tick marks on the Y3-axis in a 5Y-Axis chart. The `setDisplay()` method must be used to enable and show this object.

Syntax:

```
IdentObj getY3MajorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY3MajorGrid(), false);
setDisplay(getY3MinorGrid(), false);
setDisplay(getY3MajorTick(), true);
setTickLength(getY3MajorTick(), 1000);
setBorderColor(getY3MajorTick(), new Color(255,0,0));
```



getY3MinorGrid()

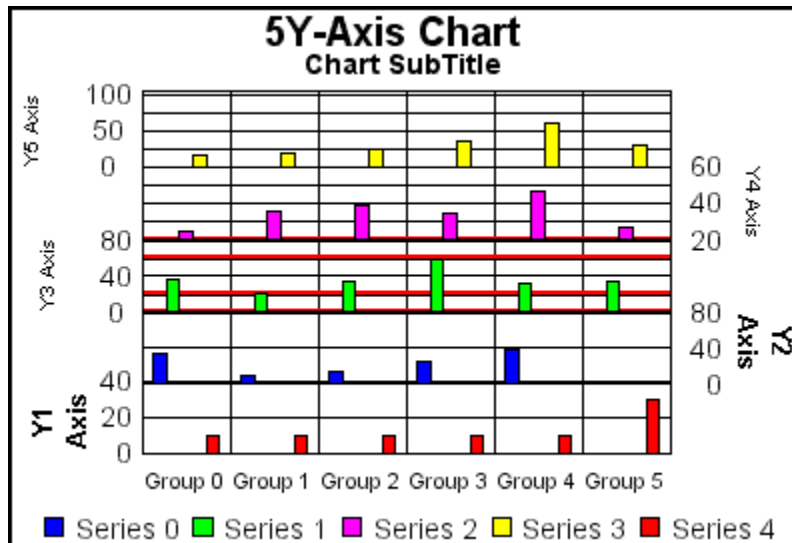
This method returns the object ID of the Y3-axis minor grid lines.

Syntax:

```
IdentObj getY3MinorGrid();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY3MinorGrid(), 2);
setFillColor(getY3MinorGrid(), new Color(255,0,0));
```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

getY3MinorTick()

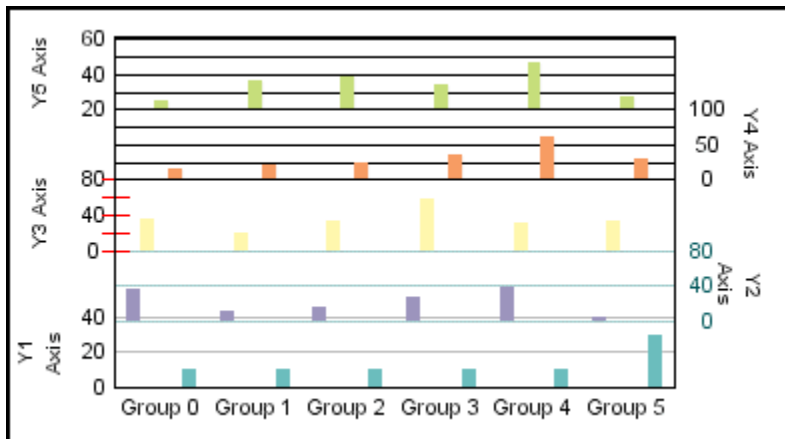
This method returns the object ID of minor tick marks on the Y3-axis in a 5Y-Axis chart. The `setDisplay()` method must be used to enable and show this object.

Syntax:

```
IdentObj getY3MinorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY3MajorGrid(), false);
setDisplay(getY3MinorGrid(), false);
setDisplay(getY3MinorTick(), true);
setTickLength(getY3MinorTick(), 1000);
setBorderColor(getY3MinorTick(), new Color(255,0,0));
```



getY3Title()

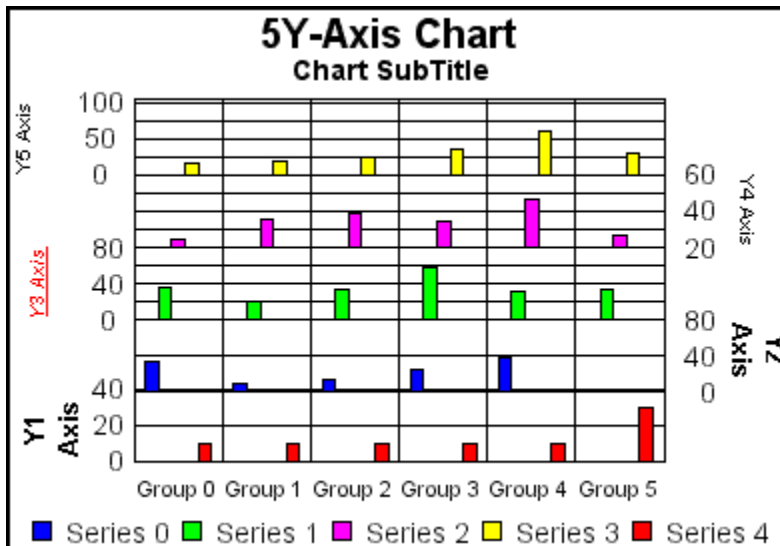
This method returns the object ID of the Y3-axis title.

Syntax:

```
IdentObj getY3Title();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY3Title(), 7);
setFillColor(getY3Title(), new Color(255,0,0));
```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3TitleBox\(\)](#) on page 1188, [getY3ZeroLine\(\)](#) on page 1189.

getY3TitleBox()

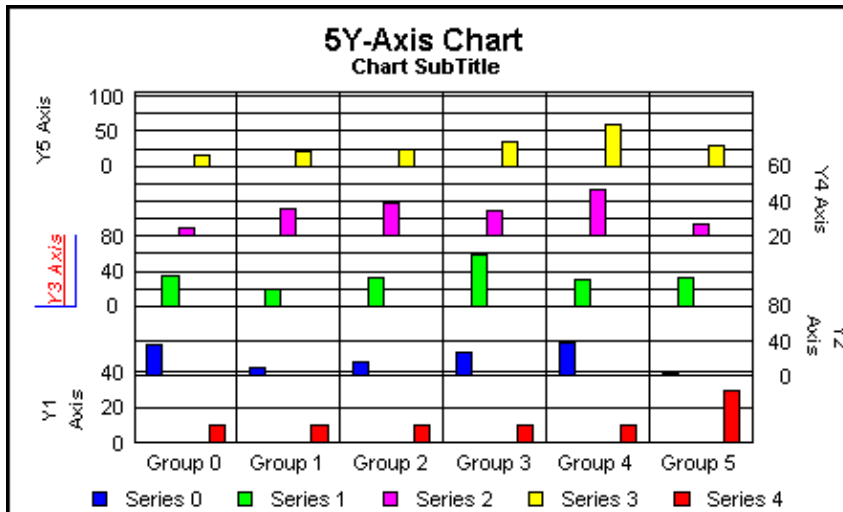
This method returns the object ID of the title box on a graph with an Y3-axis.

Syntax:

```
IdentObj getY3TitleBox();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY3Title(), 7);
setFillColor(getY3Title(), new Color(255,0,0));
setDisplay(getY3TitleBox(), true);
setBorderColor(getY3TitleBox(), new Color (0,0,255));
```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3ZeroLine\(\)](#) on page 1189.

getY3ZeroLine()

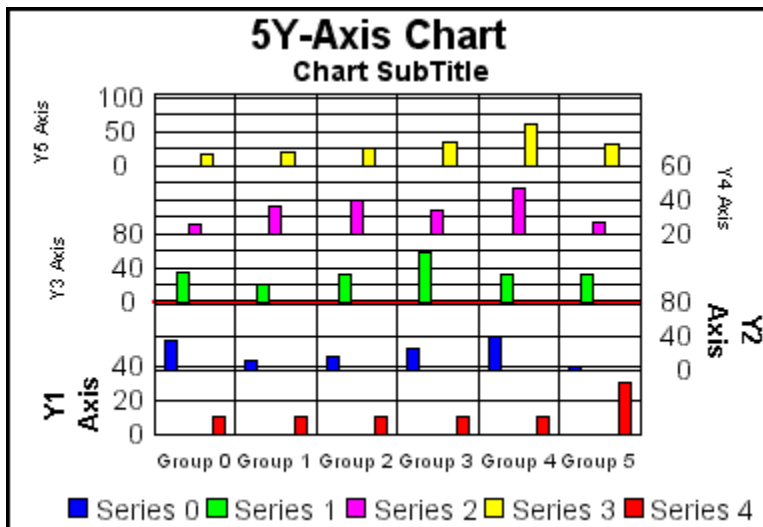
This method returns the object ID of the Y3-axis zero line.

Syntax:

```
IdentObj getY3ZeroLine();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY3ZeroLine(), 2);
setFillColor(getY3ZeroLine(), new Color (255,0,0));
setDisplay(getY3ZeroLine(), true);
```



See [getY3Axis\(\)](#) on page 1178, [getY3AxisLine\(\)](#) on page 1179, [getY3Label\(\)](#) on page 1181, [getY3MajorGrid\(\)](#) on page 1183, [getY3MinorGrid\(\)](#) on page 1184, [getY3Title\(\)](#) on page 1186, [getY3TitleBox\(\)](#) on page 1188.

getY4Axis()

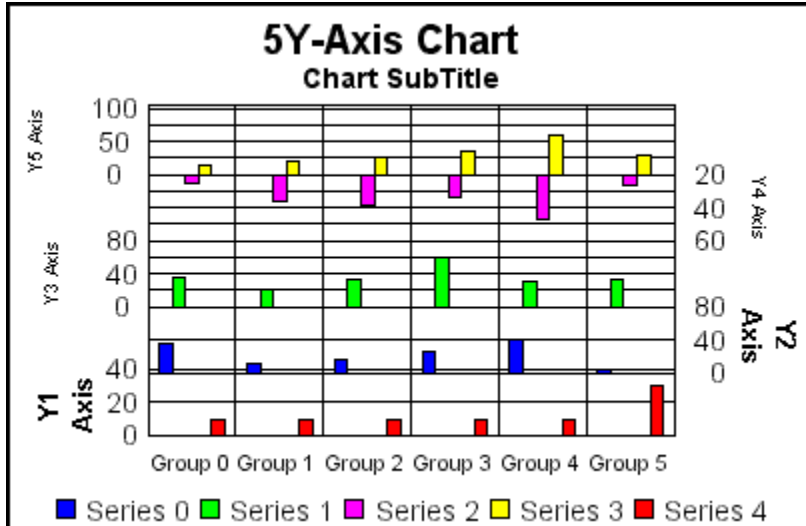
This method returns the object ID of the Y4-axis.

Syntax:

```
IdentObj getY4Axis();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setAxisDescending(getY4Axis(), true);
```



See [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4Title\(\)](#) on page 1197, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

getY4AxisLine()

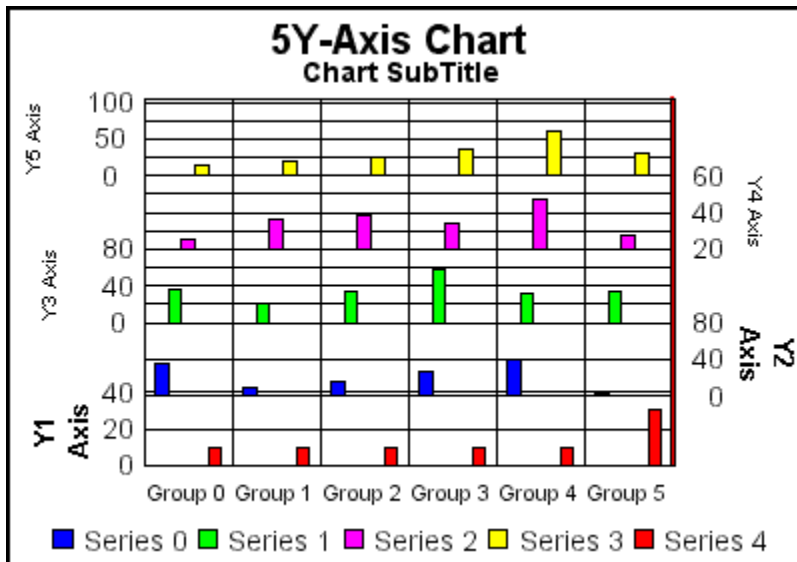
This method returns the object ID of the Y4-axis base line.

Syntax:

```
IdentObj getY4AxisLine();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY4AxisLine(), 2);
setFillColor(getY4AxisLine(), new Color (255,0,0));
```



See [getY4Axis\(\)](#) on page 1190, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4ZeroLine\(\)](#) on page 1200, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

getY4Label()

This method returns the object ID of labels on the Y4-axis.

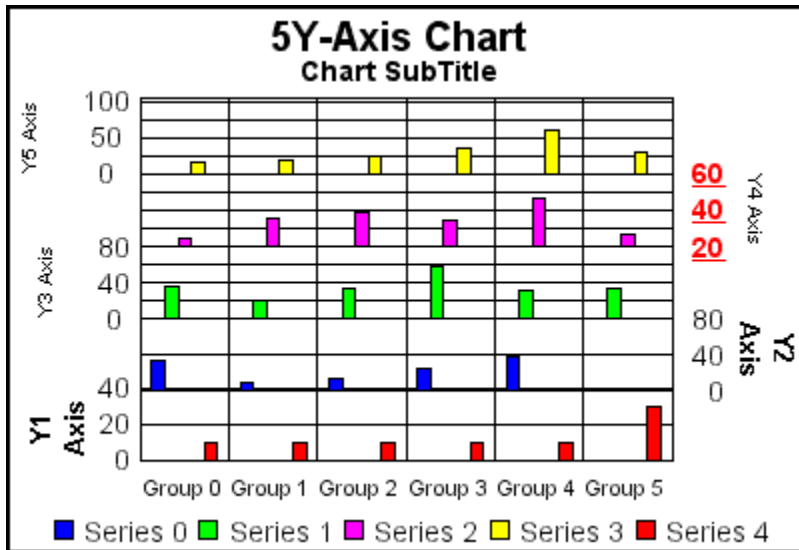
Syntax:

```
IdentObj getY4Label();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
```

```
setFontStyle(getY4Label\(\), 6);
setFillColor(getY4Label\(\), new Color(255,0,0));
```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4Title\(\)](#) on page 1197, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getY4MajorGrid()

This method returns the object ID of the Y4-axis major grid lines.

Syntax:

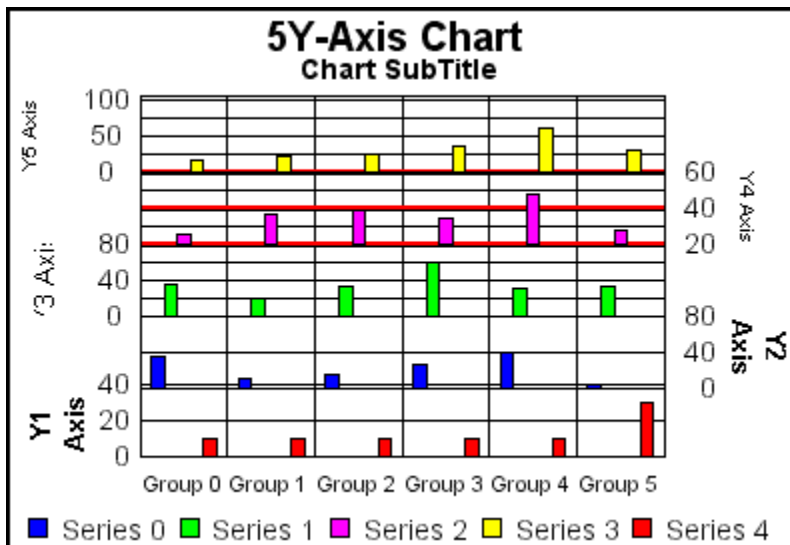
```
IdentObj getY4MajorGrid\(\);
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY4MajorGrid(), 2);
setFillColor(getY4MajorGrid(), new Color(255,0,0));

```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MinorGrid\(\)](#) on page 1195, [getY4Title\(\)](#) on page 1197, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

getY4MajorTick()

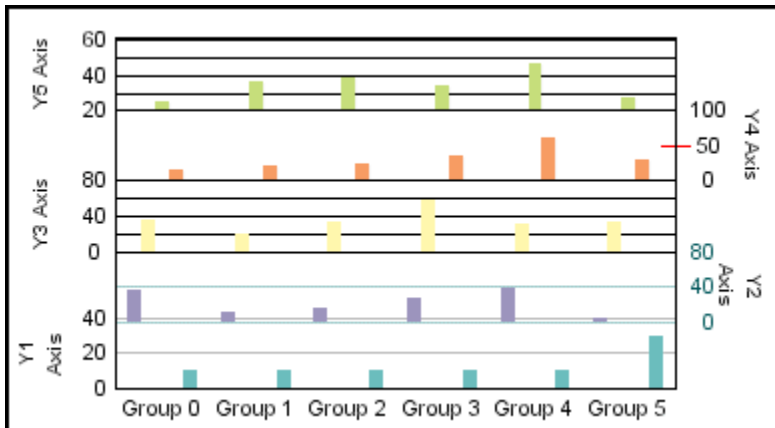
This method returns the object ID of major tick marks on the Y4-axis in a 5Y-Axis chart. The [setDisplay\(\)](#) method must be used to enable and show this object.

Syntax:

```
IdentObj getY4MajorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY4MajorGrid(), false);
setDisplay(getY4MinorGrid(), false);
setDisplay(getY4MajorTick(), true);
setTickLength(getY4MajorTick(), 1000);
setBorderColor(getY4MajorTick(), new Color(255,0,0));
```

**getY4MinorGrid()**

This method returns the object ID of the Y4-axis minor grid lines.

Syntax:

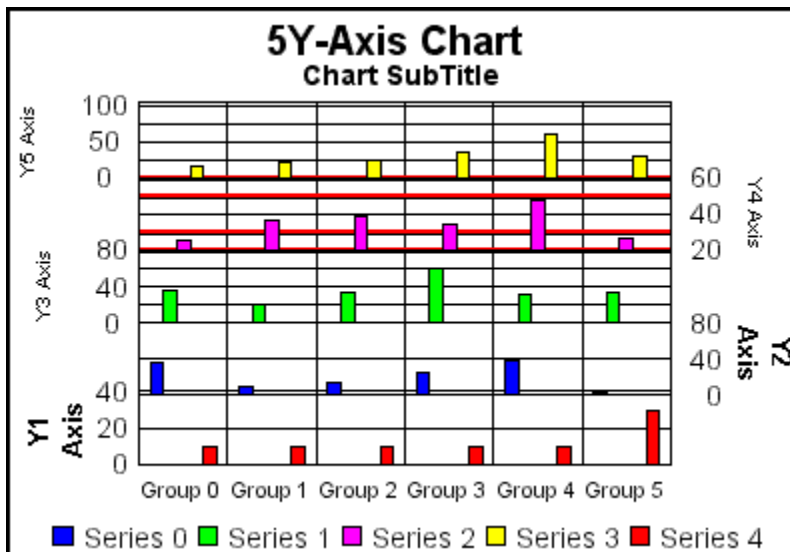
```
IdentObj getY4MinorGrid();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY4MinorGrid(), 2);
setFillColor(getY4MinorGrid(), new Color(255,0,0));

```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4Title\(\)](#) on page 1197, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

getY4MinorTick()

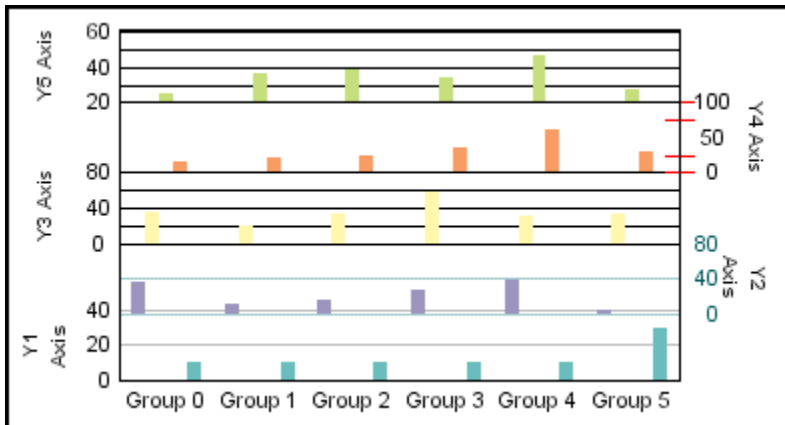
This method returns the object ID of minor tick marks on the Y4-axis in a 5Y-Axis chart. The `setDisplay()` method must be used to enable and show this object.

Syntax:

```
IdentObj getY4MinorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY4MajorGrid(), false);
setDisplay(getY4MinorGrid(), false);
setDisplay(getY4MinorTick(), true);
setTickLength(getY4MinorTick(), 1000);
setBorderColor(getY4MinorTick(), new Color(255,0,0));
```

**getY4Title()**

This method returns the object ID of the Y4-axis title.

Syntax:

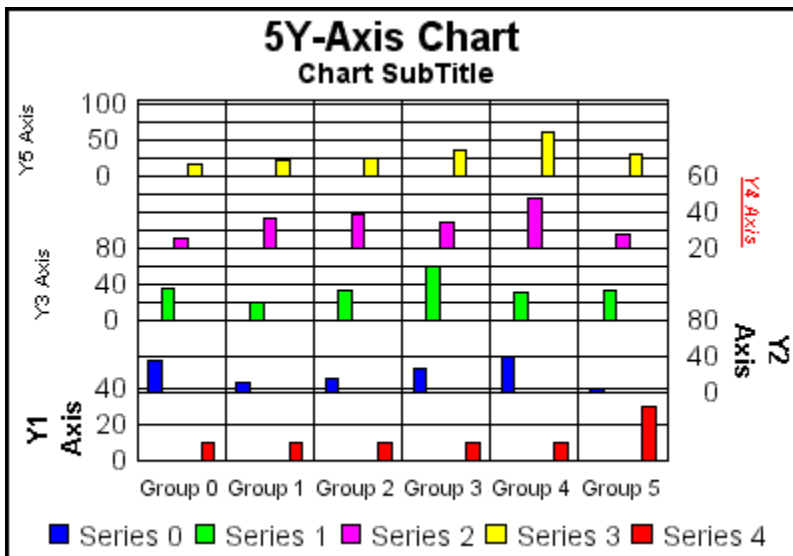
```
IdentObj getY4Title();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY4Title(), 7);
setFillColor(getY4Title(), new Color(255,0,0));

```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4TitleBox\(\)](#) on page 1198, [getY4ZeroLine\(\)](#) on page 1200.

getY4TitleBox()

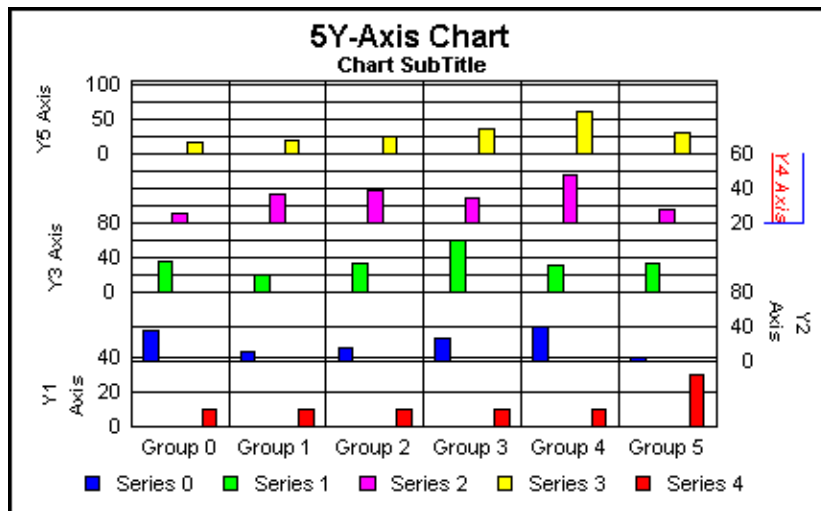
This method returns the object ID of the title box on a graph with an Y4-axis.

Syntax:

```
IdentObj getY4TitleBox();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY4Title(), 7);
setFillColor(getY4Title(), new Color(255,0,0));
setBorderColor(getY4TitleBox(), new Color(0,0,255));
setDisplay(getY4TitleBox(), true);
```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4Title\(\)](#) on page 1197, [getY4ZeroLine\(\)](#) on page 1200.

getY4ZeroLine()

This method returns the object ID of the Y4-axis zero line.

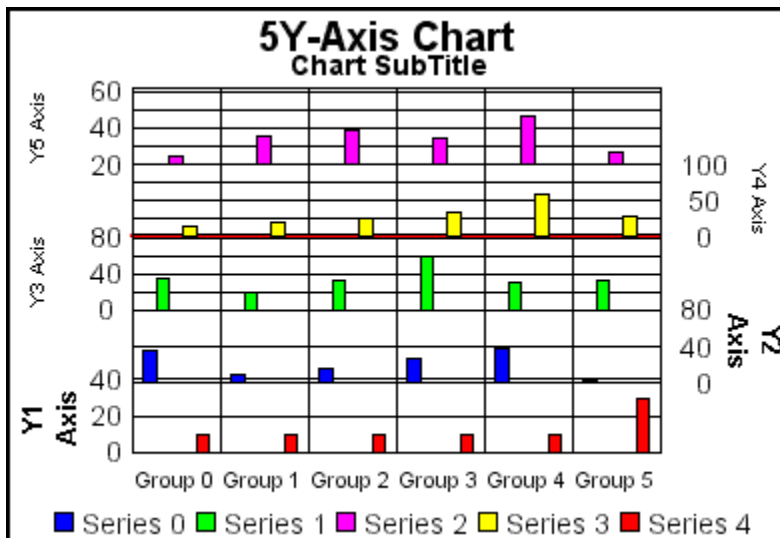
Syntax:

```
IdentObj getY4ZeroLine();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");

setLineWidth(getY4ZeroLine(), 2);
setFillColor(getY4ZeroLine(), new Color (255,0,0));
setDisplay(getY4ZeroLine(), true);
```



See [getY4Axis\(\)](#) on page 1190, [getY4AxisLine\(\)](#) on page 1191, [getY4Label\(\)](#) on page 1192, [getY4MajorGrid\(\)](#) on page 1193, [getY4MinorGrid\(\)](#) on page 1195, [getY4Title\(\)](#) on page 1197, [getY4TitleBox\(\)](#) on page 1198.

getY5Axis()

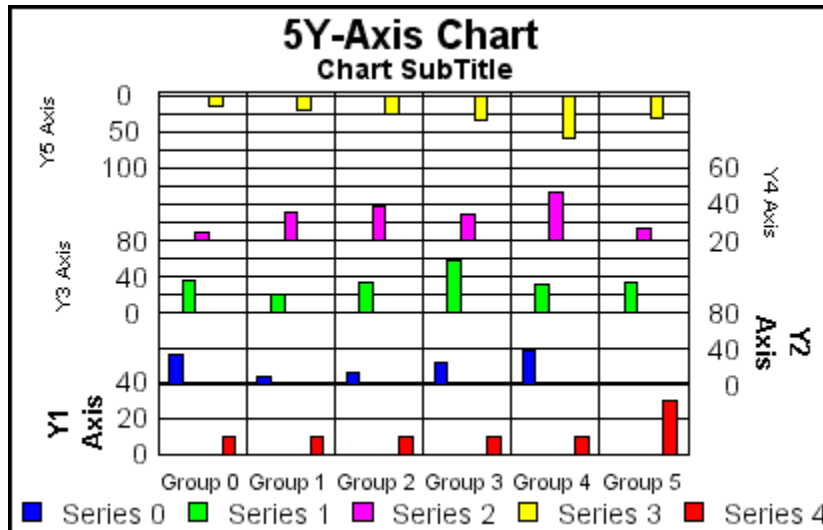
This method returns the object ID of the Y5-axis.

Syntax:

```
IdentObj getY5Axis();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setAxisDescending(getY5Axis(), true);
```



See [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

getY5AxisLine()

This method returns the object ID of the Y5-axis base line.

Syntax:

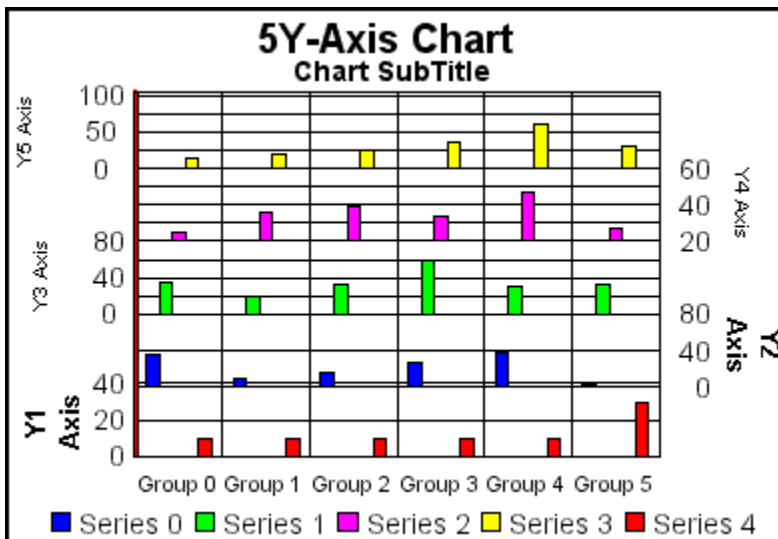
```
IdentObj getY5AxisLine();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY5AxisLine(), 2);
setFillColor(getY5AxisLine(), new Color (255,0,0));

```



See [getY5Axis\(\)](#) on page 1201, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

getY5Label()

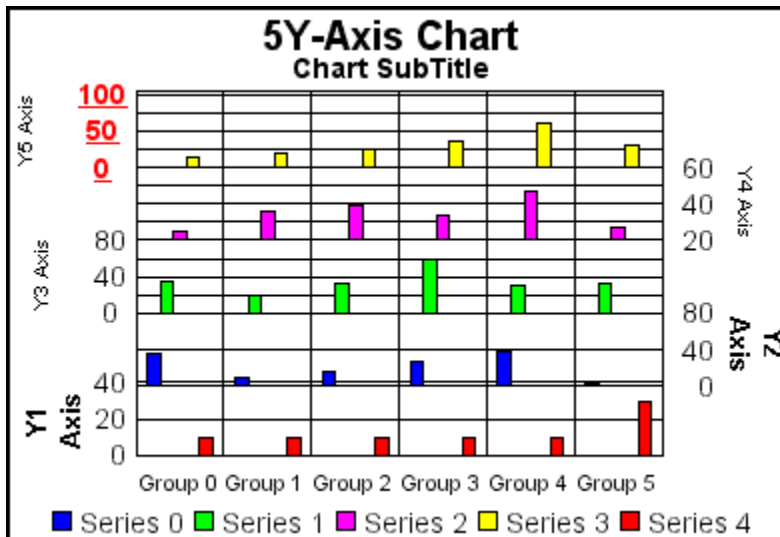
This method returns the object ID of labels on the Y5-axis.

Syntax:

```
IdentObj getY5Label();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY5Label(), 6);
setFillColor(getY5Label(), new Color(255,0,0));
```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

For information about formatting numeric labels, see [Formatting Numbers and Numeric Labels](#) on page 130.

getY5MajorGrid()

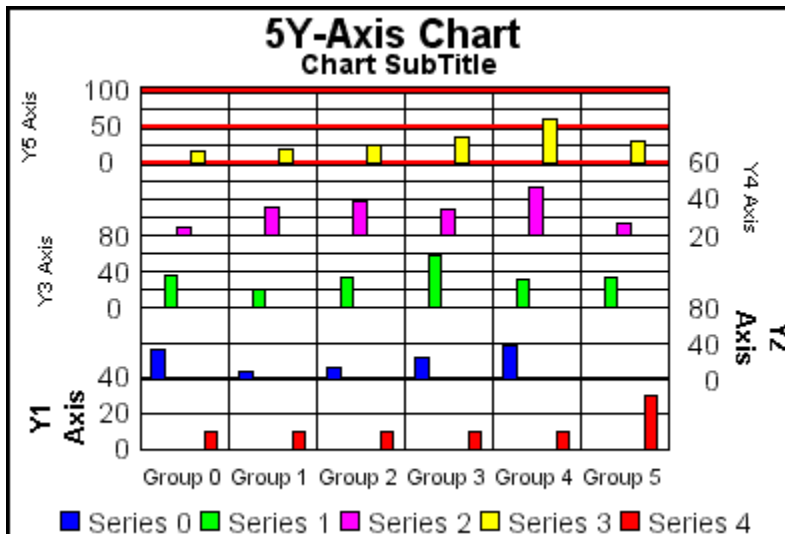
This method returns the object ID of the Y5-axis major grid lines.

Syntax:

```
IdentObj getY5MajorGrid();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY5MajorGrid(), 2);
setFillColor(getY5MajorGrid(), new Color(255,0,0));
```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

getY5MajorTick()

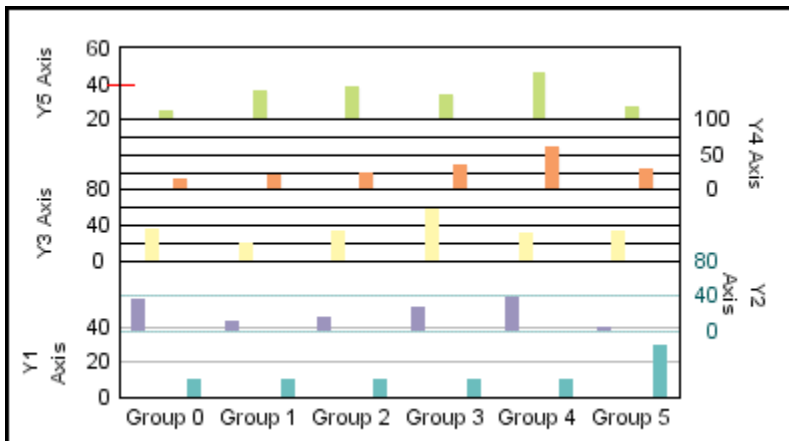
This method returns the object ID of major tick marks on the Y5-axis in a 5Y-Axis chart. The `setDisplay()` method must be used to enable and show this object.

Syntax:

```
IdentObj getY5MajorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY5MajorGrid(), false);
setDisplay(getY5MinorGrid(), false);
setDisplay(getY5MajorTick(), true);
setTickLength(getY5MajorTick(), 1000);
setBorderColor(getY5MajorTick(), new Color(255,0,0));
```



getY5MinorGrid()

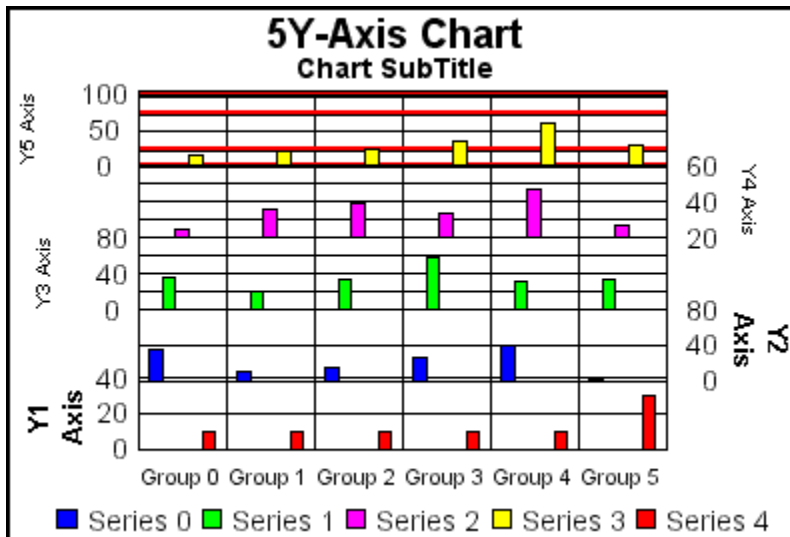
This method returns the object ID of the Y5-axis minor grid lines.

Syntax:

```
IdentObj getY5MinorGrid();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY5MinorGrid(), 2);
setFillColor(getY5MinorGrid(), new Color(255,0,0));
```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

getY5MinorTick()

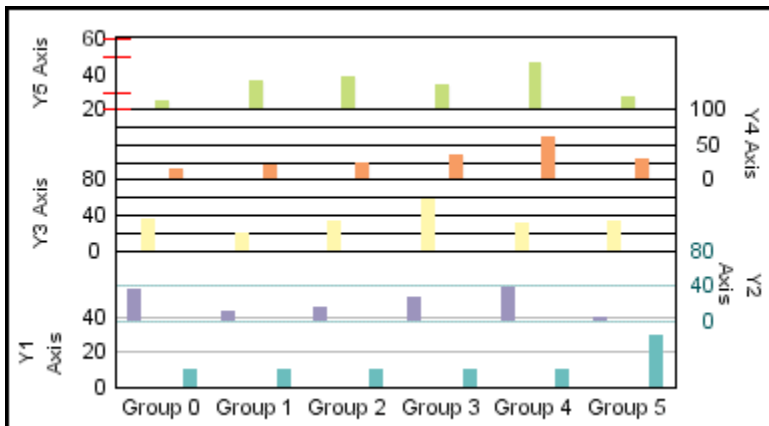
This method returns the object ID of minor tick marks on the Y5-axis in a 5Y-Axis chart. The `setDisplay()` method must be used to enable and show this object.

Syntax:

```
IdentObj getY5MinorTick();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 4);
setAxisAssignment(getSeries(3), 3);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setDisplay(getY5MajorGrid(), false);
setDisplay(getY5MinorGrid(), false);
setDisplay(getY5MinorTick(), true);
setTickLength(getY5MinorTick(), 1000);
setBorderColor(getY5MinorTick(), new Color(255,0,0));
```



getY5Title()

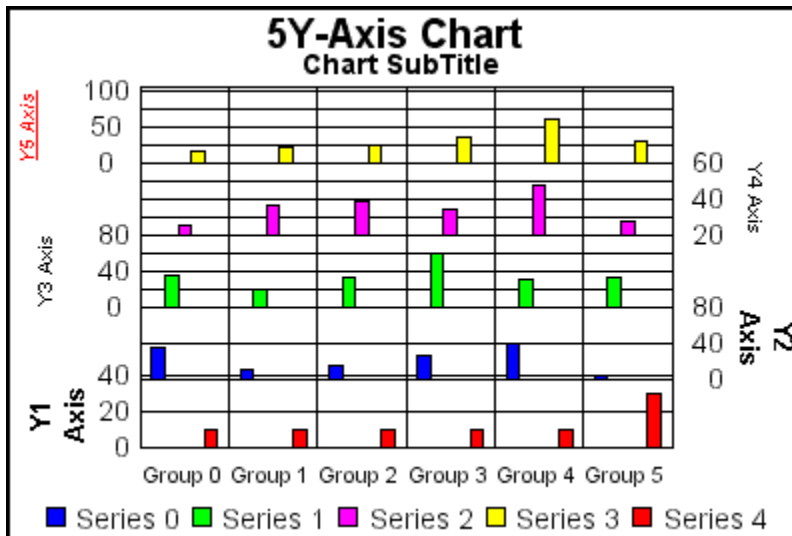
This method returns the object ID of the Y5-axis title.

Syntax:

```
IdentObj getY5Title();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY5Title(), 7);
setFillColor(getY5Title(), new Color(255,0,0));
```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5TitleBox\(\)](#) on page 1210, [getY5ZeroLine\(\)](#) on page 1211.

getY5TitleBox()

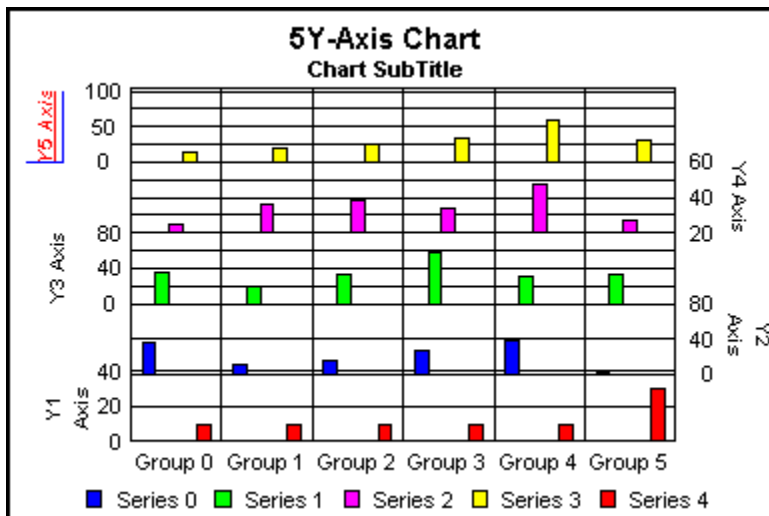
This method returns the object ID of the Y5-axis title box.

Syntax:

```
IdentObj getY5TitleBox();
```

Example:

```
setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setFontStyle(getY5Title(), 7);
setFillColor(getY5Title(), new Color(255,0,0));
setBorderColor(getY5TitleBox(), new Color(0,0,255));
setDisplay(getY5TitleBox(), true);
```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5ZeroLine\(\)](#) on page 1211.

getY5ZeroLine()

This method returns the object ID of the Y5-axis zero line.

Syntax:

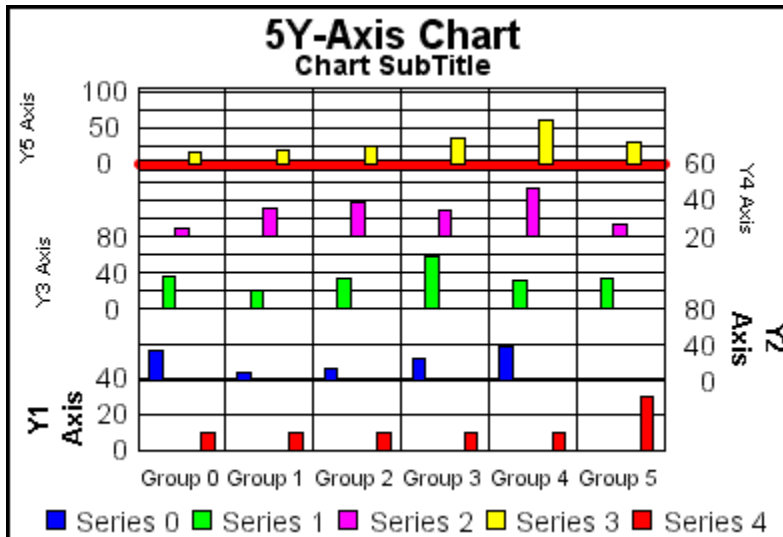
```
IdentObj getY5ZeroLine();
```

Example:

```

setDepthAngle(0);
setDepthRadius(0);
setFootnoteDisplay(false);
setAxisAssignment(getSeries(0), 1);
setAxisAssignment(getSeries(1), 2);
setAxisAssignment(getSeries(2), 3);
setAxisAssignment(getSeries(3), 4);
setGraphType(105);
setTitleString("5Y-Axis Chart");
setY1TitleString("Y1 Axis");
setY2TitleString("Y2 Axis");
setTextString(getY3Title(), "Y3 Axis");
setTextString(getY4Title(), "Y4 Axis");
setTextString(getY5Title(), "Y5 Axis");
setLineWidth(getY5ZeroLine(), 5);
setFillColor(getY5ZeroLine(), new Color(255,0,0));
setY5ZeroLineDisplay(true);

```



See [getY5Axis\(\)](#) on page 1201, [getY5AxisLine\(\)](#) on page 1202, [getY5Label\(\)](#) on page 1203, [getY5MajorGrid\(\)](#) on page 1205, [getY5MinorGrid\(\)](#) on page 1206, [getY5Title\(\)](#) on page 1208, [getY5TitleBox\(\)](#) on page 1210.

Graph Types and 3D Preset Viewing Angles

This appendix describes graph types and 3D preset viewing angles.

In this appendix:

- ❑ [Graph Types](#)
- ❑ [3D Preset Viewing Angles](#)

Graph Types

The following table lists each of the possible graph types that you can select using the GraphType property:

#	GraphType	#	GraphType
0	<i>3D Bars (GraphType=0)</i>	52	<i>Horizontal Bi-Polar Absolute Line (GraphType=52)</i>
1	<i>3D Pyramid (GraphType=1)</i>	53	<i>Horizontal Bi-Polar Stacked Line (GraphType=53)</i>
2	<i>3D Octagon (GraphType=2)</i>	54	<i>Horizontal Percent Line (GraphType=54)</i>
3	<i>3D Cylinder (GraphType=3)</i>	55	<i>Pie (GraphType=55)</i>
4	<i>3D Floating Cubes (GraphType=4)</i>	56	<i>Ring Pie (GraphType=56)</i>
5	<i>3D Floating Pyramids (GraphType=5)</i>	57	<i>Multi Pie (GraphType=57)</i>
6	<i>3D Connected Series Area (GraphType=6)</i>	58	<i>Multi Ring Pie (GraphType=58)</i>
7	<i>3D Connected Series Ribbon (GraphType=7)</i>	59	<i>Multi Proportional Pie (GraphType=59)</i>

#	GraphType	#	GraphType
8	<i>3D Cone (Graph Type=8)</i>	60	<i>Multi Proportional Ring Pie (GraphType=60)</i>
9	<i>3D Connected Group Area (GraphType=9)</i>	61	<i>XY Scatter (GraphType=61)</i>
10	<i>3D Connected Group Ribbon (GraphType=10)</i>	62	<i>XY Scatter Dual-Axis (GraphType=62)</i>
11	<i>3D Sphere (GraphType=11)</i>	63	<i>XY Scatter with Labels (GraphType=63)</i>
12	<i>3D Surface (GraphType=12)</i>	64	<i>XY Scatter with Labels Dual-Axis (GraphType=64)</i>
13	<i>3D Surface with Sides (GraphType=13)</i>	65	<i>XY Polar (GraphType=65)</i>
14	<i>3D Honeycomb Surface (GraphType=14)</i>	66	<i>XY Polar Dual-Axis (GraphType=66)</i>
15	<i>3D Smooth Surface (GraphType=15)</i>	67	<i>Radar Line (GraphType=67)</i>
16	<i>3D Smooth Surface with Sides (GraphType=16)</i>	68	<i>Radar Area (GraphType=68)</i>
17	<i>Vertical Clustered Bars (GraphType=17)</i>	69	<i>Radar Line Dual-Axis (GraphType=69)</i>
18	<i>Vertical Stacked Bars (GraphType=18)</i>	70	<i>Open-Hi-Lo-Close Candle Stock Graph (GraphType=70)</i>
19	<i>Vertical Dual-Axis Clustered Bars (GraphType=19)</i>	71	<i>Open-Hi-Lo-Close Candle Stock Graph with Volume (GraphType=71)</i>
20	<i>Vertical Dual-Axis Stacked Bars (GraphType=20)</i>	72	<i>Candle Stock Open-Close (GraphType=72)</i>
21	<i>Vertical Bi-Polar Clustered Bars (GraphType=21)</i>	73	<i>Stock Hi-Lo (GraphType=73)</i>

#	GraphType	#	GraphType
22	<i>Vertical Bi-Polar Stacked Bars (GraphType=22)</i>	74	<i>Stock Hi-Lo Dual-Axis (GraphType=74)</i>
23	<i>Vertical Percent Bars (GraphType=23)</i>	75	<i>Stock Hi-Lo Bi-Polar (GraphType=75)</i>
24	<i>Horizontal Clustered Bars (GraphType=24)</i>	76	<i>Stock Hi-Lo Close (GraphType=76)</i>
25	<i>Horizontal Stacked Bars (GraphType=25)</i>	77	<i>Stock Hi-Lo Close Dual-Axis (GraphType=77)</i>
26	<i>Horizontal Dual-Axis Clustered Bars (GraphType=26)</i>	78	<i>Stock Hi-Lo Close Bi-Polar (GraphType=78)</i>
27	<i>Horizontal Dual-Axis Stacked Bars (GraphType=27)</i>	79	<i>Stock Hi-Lo Open-Close (GraphType=79)</i>
28	<i>Horizontal Bi-Polar Clustered Bars (GraphType=28)</i>	80	<i>Stock Hi-Lo Open-Close Dual-Axis (GraphType=80)</i>
29	<i>Horizontal Bi-Polar Stacked Bars (GraphType=29)</i>	81	<i>Stock Hi-Lo Open-Close Bi-Polar (GraphType=81)</i>
30	<i>Horizontal Percent Bars (GraphType=30)</i>	82	<i>Stock Hi-Lo with Volume (GraphType=82)</i>
31	<i>Vertical Absolute Area (GraphType=31)</i>	83	<i>Stock Hi-Lo Open-Close with Volume (GraphType=83)</i>
32	<i>Vertical Stacked Area (GraphType=32)</i>	84	<i>Candle Stock Open-Close with Volume (GraphType=84)</i>
33	<i>Vertical Bi-Polar Absolute Area (GraphType=33)</i>	85	<i>Vertical Histogram (GraphType=85)</i>
34	<i>Vertical Bi-Polar Stacked Area (GraphType=34)</i>	86	<i>Horizontal Histogram (GraphType=86)</i>

#	GraphType	#	GraphType
35	<i>Vertical Percent Area (GraphType=35)</i>	87	<i>Spectral Map (GraphType=87)</i>
36	<i>Horizontal Absolute Area (GraphType=36)</i>	88	<i>Stock Hi-Lo Close with Volume (GraphType=88)</i>
37	<i>Horizontal Stacked Area (GraphType=37)</i>	89	<i>Bubble Graph (GraphType=89)</i>
38	<i>Horizontal Bi-Polar Absolute Area (GraphType=38)</i>	90	<i>Bubble Graph with Labels (GraphType=90)</i>
39	<i>Horizontal Bi-Polar Stacked Area (GraphType=39)</i>	91	<i>Bubble Graph Dual-Axis (GraphType=91)</i>
40	<i>Horizontal Percent Area (GraphType=40)</i>	92	<i>Bubble Graph with Labels Dual-Axis (GraphType=92)</i>
41	<i>Vertical Absolute Line (GraphType=41)</i>	93	<i>Pie-Bar Graph (GraphType=93)</i>
42	<i>Vertical Stacked Line (GraphType=42)</i>	94	<i>Ring Pie-Bar Graph (GraphType=94)</i>
43	<i>Vertical Dual-Axis Absolute Line (GraphType=43)</i>	99	Reserved for Future Use
44	<i>Vertical Dual-Axis Stacked Line (GraphType=44)</i>	100	<i>Vertical Waterfall Graph (GraphType=100)</i>
45	<i>Vertical Bi-Polar Absolute Line (GraphType=45)</i>	101	<i>Horizontal Waterfall Graph (GraphType=101)</i>
46	<i>Vertical Bi-Polar Stacked Line (GraphType=46)</i>	102	<i>Pareto Graph (GraphType=102)</i>
47	<i>Vertical Percent Line (GraphType=47)</i>	103	<i>Multi-Y Y1/Y2/Y3-Axes Graph (GraphType=103)</i>

#	GraphType	#	GraphType
48	Horizontal Absolute Line (GraphType=48)	104	Multi-Y Y1/Y2/Y3/Y4-Axes Graph (GraphType=104)
49	Horizontal Stacked Line (GraphType=49)	105	Multi-Y Y1/Y2/Y3/Y4/Y5-Axes Graph (GraphType=105)
50	Horizontal Dual-Axis Absolute Line (GraphType=50)	106	Funnel Graph (GraphType=106)
51	Horizontal Dual-Axis Stacked Line (GraphType=51)	111	Pyramid Graph (GraphType=111) on page 1269
		124	Box Plot Graph (GraphType=124) on page 1270
		130	Horizontal Box Plot Graph (GraphType=130) on page 1270
		138	Thermometer Gauge Graph (GraphType=138) on page 1269

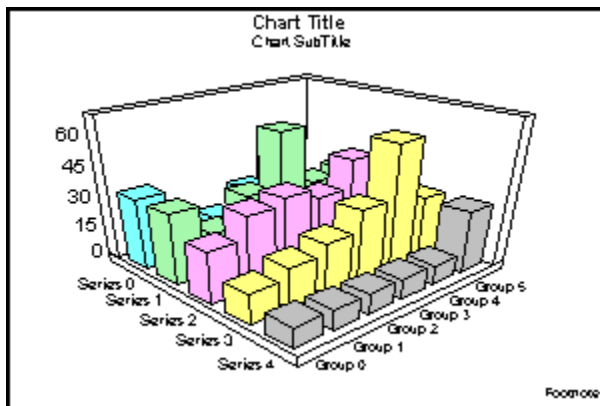
See [Using WebFOCUS Graphics](#) on page 63 for information about selecting and using the GraphType property.

3D Graphs

This topic contains illustrations that show the different types of 3D graphs.

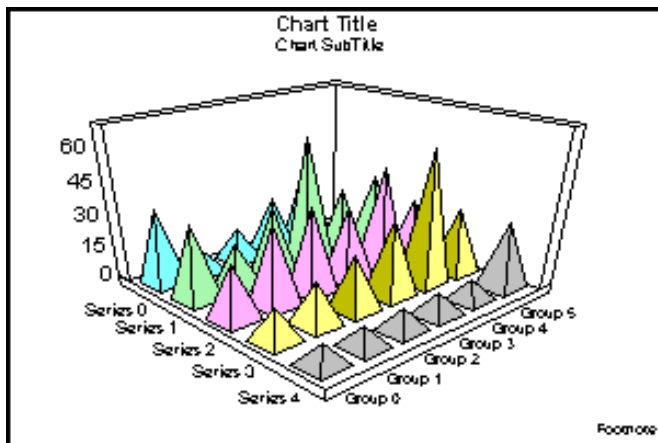
3D Bars (GraphType=0)

This is a standard 3D graph. It displays a bar for each value in the data set.



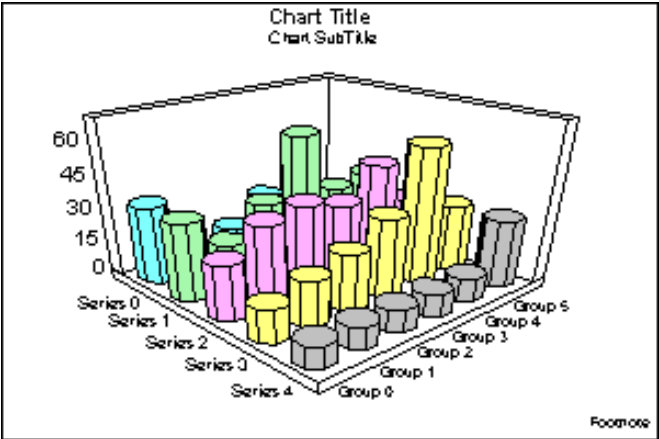
3D Pyramid (GraphType=1)

This shows pyramids to represent volume information, such as an amount of some item.



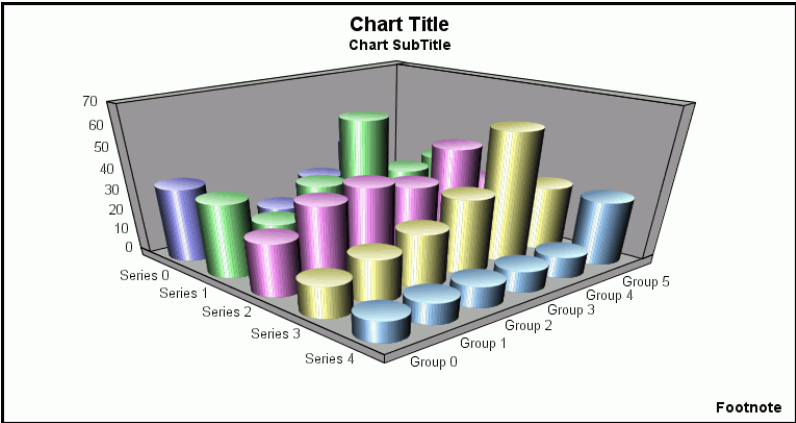
3D Octagon (GraphType=2)

This shows octagons drawn in 3D. You can make them more elliptical or more columnar.



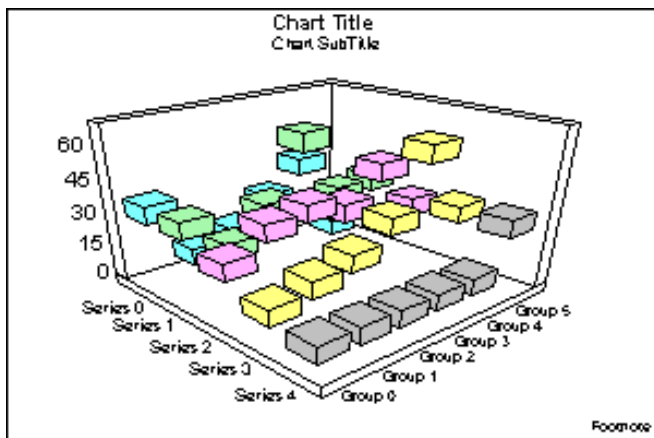
3D Cylinder (GraphType=3)

This shows cylinders drawn in 3D.



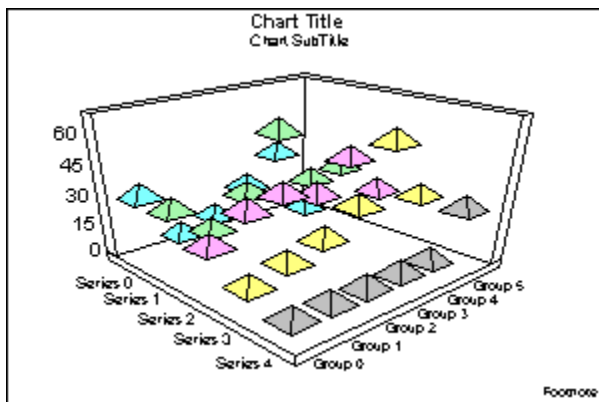
3D Floating Cubes (GraphType=4)

This shows a 3D graph type for data values that are close to each other. You can see under and around the cubes.



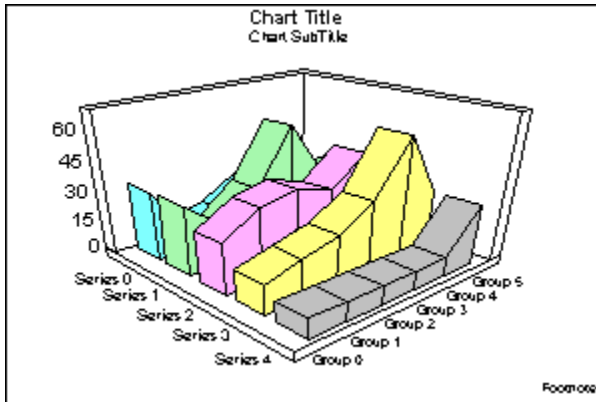
3D Floating Pyramids (GraphType=5)

This shows how the floating, diamond-shaped pyramids can trace out your data points.



3D Connected Series Area (GraphType=6)

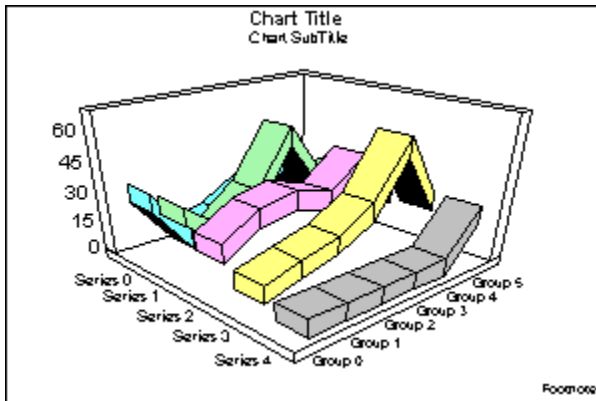
This shows trend information along the series dimension.



Note: This type of graph does not support drilldowns or conditional styling.

3D Connected Series Ribbon (GraphType=7)

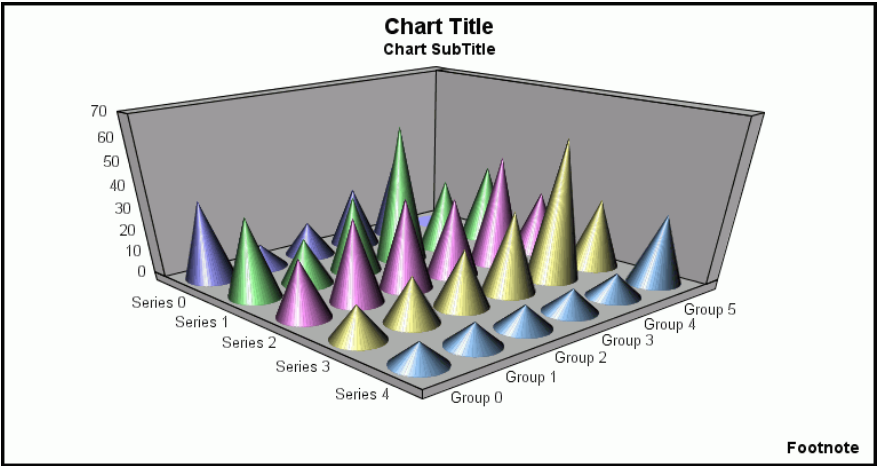
This shows trend information (represented in ribbon-shaped bars) along the series dimension.



Note: This type of graph does not support drilldowns or conditional styling.

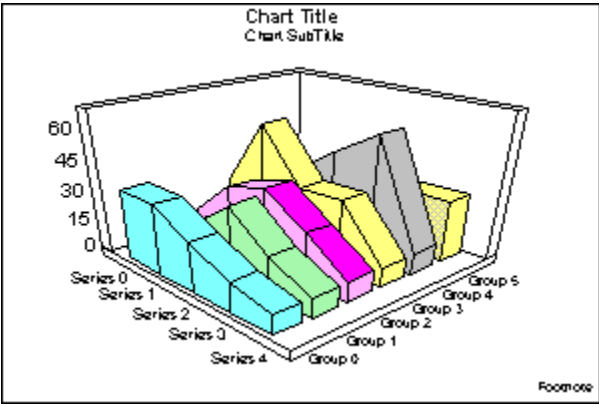
3D Cone (Graph Type=8)

This shows how cones can highlight your data points.



3D Connected Group Area (GraphType=9)

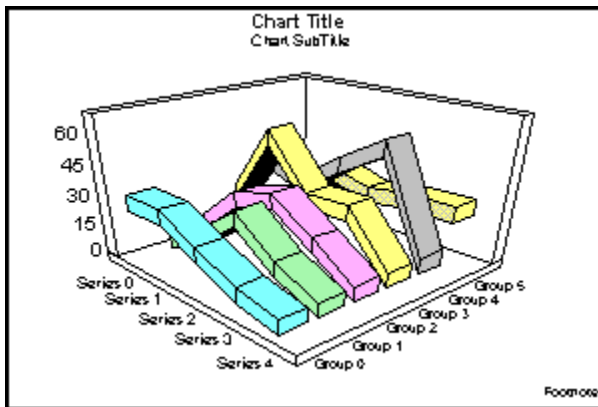
This shows trend information along the group dimension.



Note: This type of graph does not support drilldowns or conditional styling.

3D Connected Group Ribbon (GraphType=10)

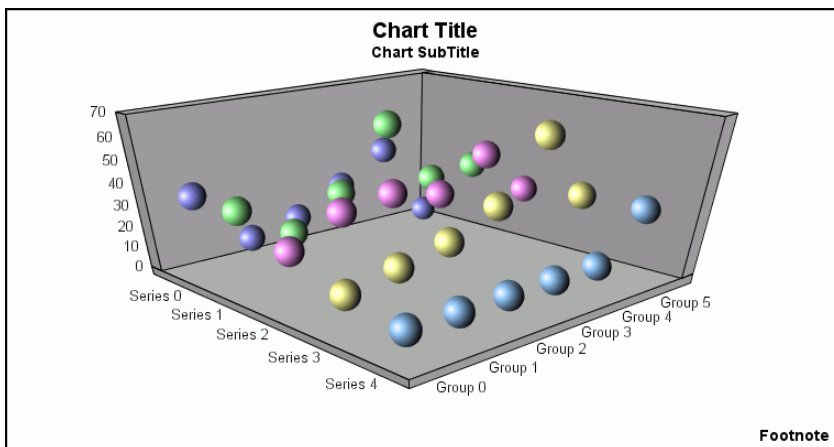
This shows trend information (represented in ribbon-shaped bars) along the group dimension.



Note: This type of graph does not support drilldowns or conditional styling.

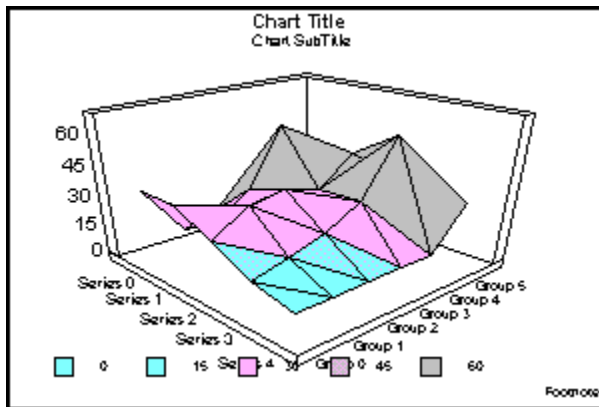
3D Sphere (GraphType=11)

This shows how spheres can highlight your data points.



3D Surface (GraphType=12)

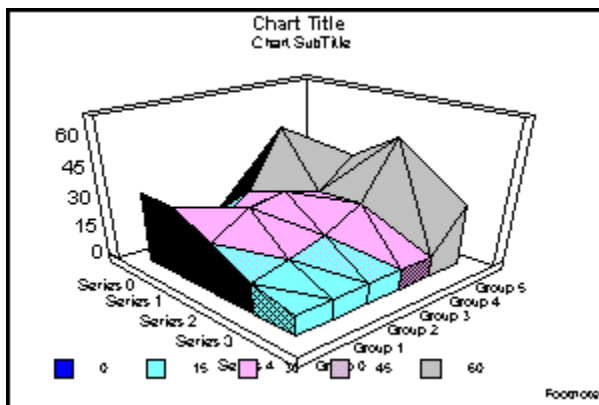
This graphs all data points as a 3D surface, using a rolling wave-like effect.



Note: This type of graph does not support drilldowns or conditional styling.

3D Surface with Sides (GraphType=13)

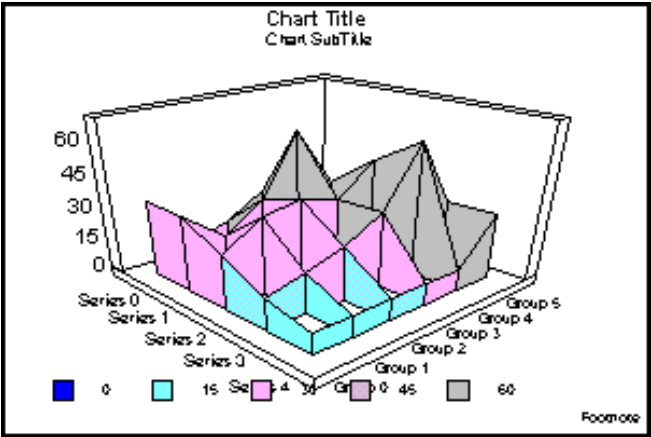
This graphs all data points as a 3D surface with solid sides.



Note: This type of graph does not support drilldowns or conditional styling.

3D Honeycomb Surface (GraphType=14)

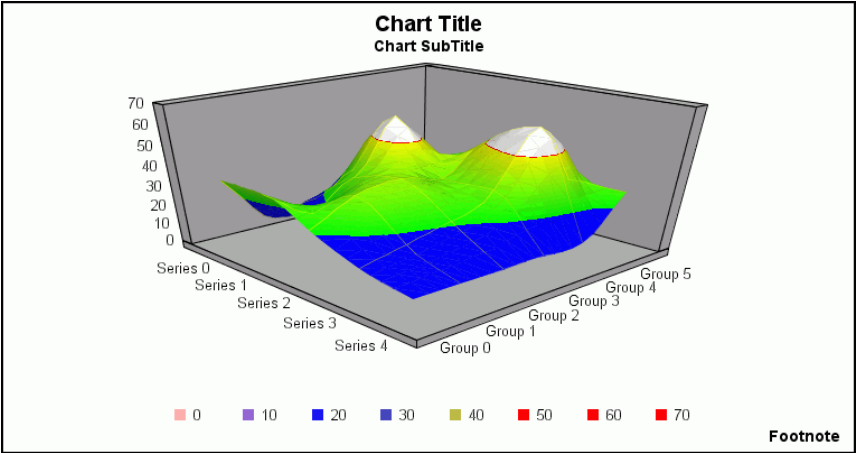
This graphs all data points as a 3D surface using a honeycomb effect.



Note: This type of graph does not support drilldowns or conditional styling.

3D Smooth Surface (GraphType=15)

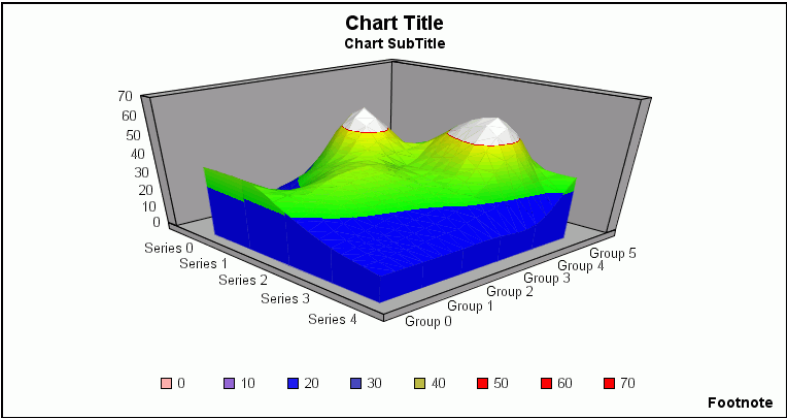
This graphs all data points as a 3D smooth surface.



Note: This type of graph does not support drilldowns or conditional styling.

3D Smooth Surface with Sides (GraphType=16)

This graphs all data points as a 3D smooth surface with sides.



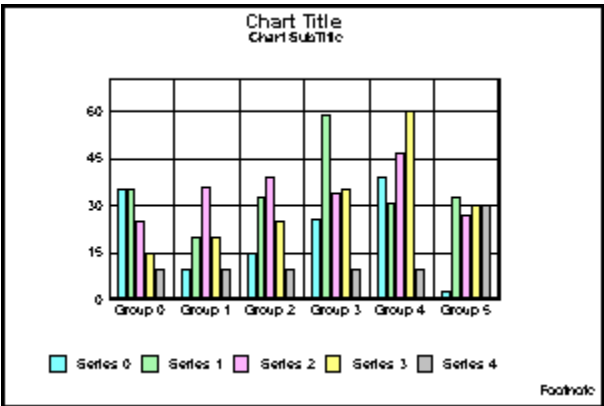
Note: This type of graph does not support drilldowns or conditional styling.

Vertical Bar Graphs

The following illustrations show the vertical bar graphs. All of the illustrations were created without 2.5D depth applied to them (for example, DepthRadius = 0).

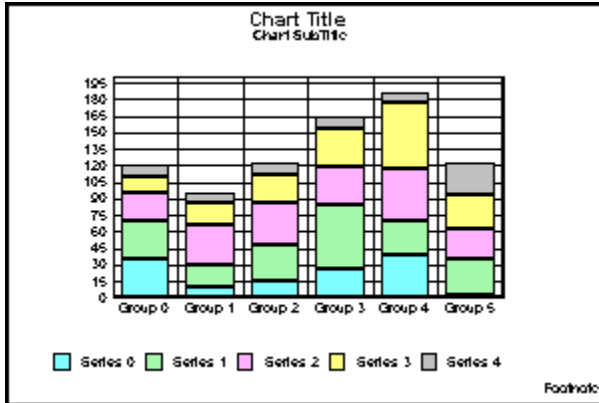
Vertical Clustered Bars (GraphType=17)

This shows bars grouped side by side. This is the standard type of two-dimensional bar graph.



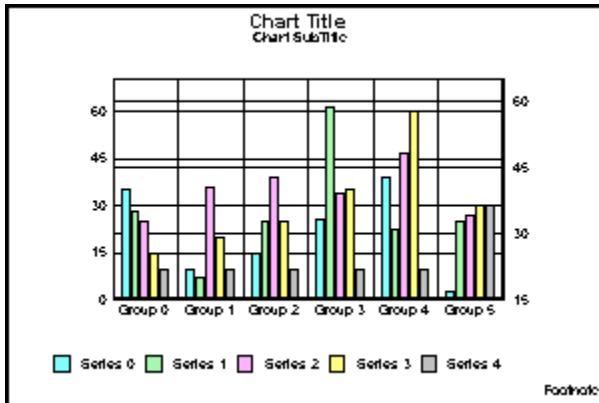
Vertical Stacked Bars (GraphType=18)

This shows stacked groups of bars. Each stack is comprised of all series in a group. The series are added together and totaled. The axis is the total value of the cumulative points.



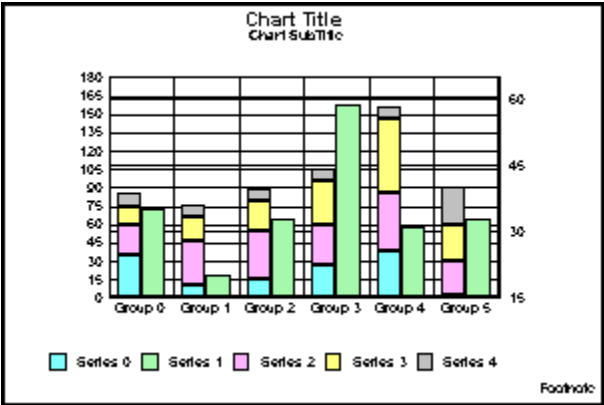
Vertical Dual-Axis Clustered Bars (GraphType=19)

This is also called a Dual-Y graph. You can assign any series to either of the two axes.



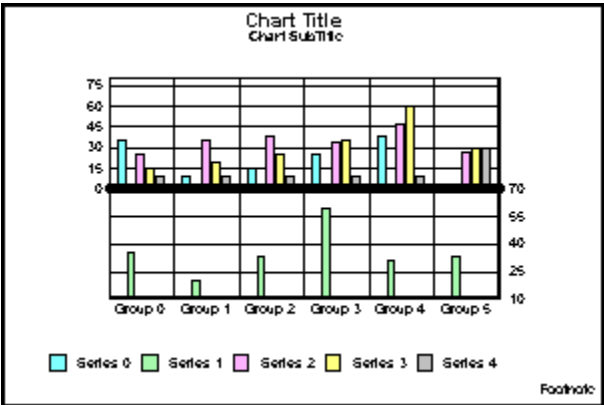
Vertical Dual-Axis Stacked Bars (GraphType=20)

This is also called a Dual-Y stacked graph. Separate stacks will be created for the data on each axis.



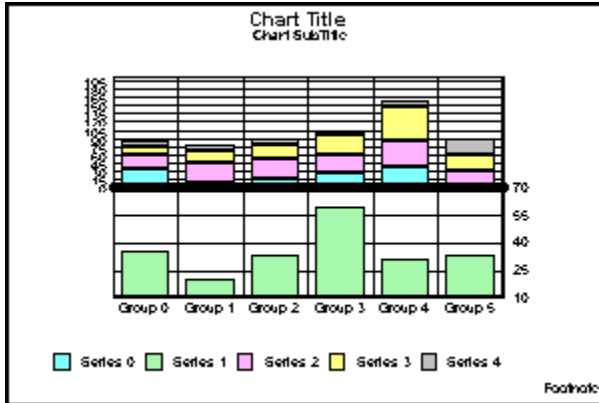
Vertical Bi-Polar Clustered Bars (GraphType=21)

This is also called a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



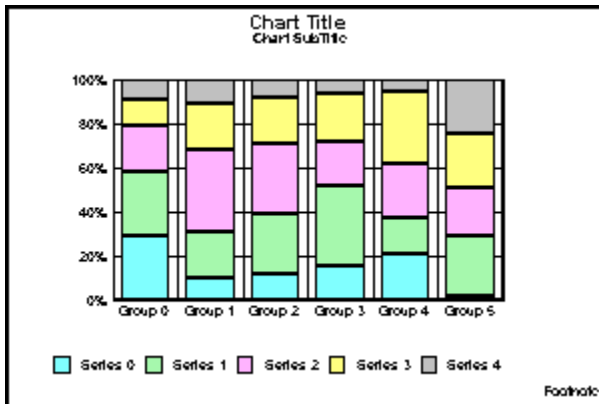
Vertical Bi-Polar Stacked Bars (GraphType=22)

This is a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Vertical Percent Bars (GraphType=23)

This is a bar version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.

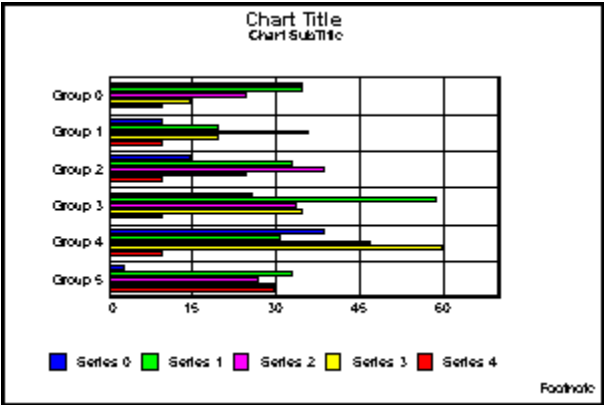


Horizontal Bar Graphs

The following illustrations show the horizontal bar graphs. All of the illustrations were created without 2.5D depth applied (for example, DepthRadius = 0).

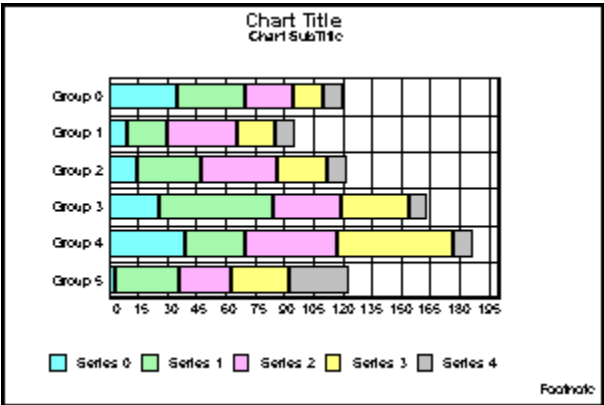
Horizontal Clustered Bars (GraphType=24)

This shows bars grouped side by side. This is the standard type of two-dimensional bar graph.



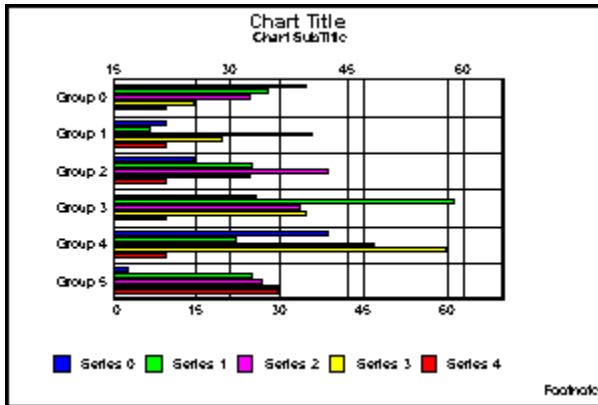
Horizontal Stacked Bars (GraphType=25)

This shows stacked groups of bars. Each stack is comprised of all series in a group. The series are added together and totaled. The axis is the total value of the cumulative points.



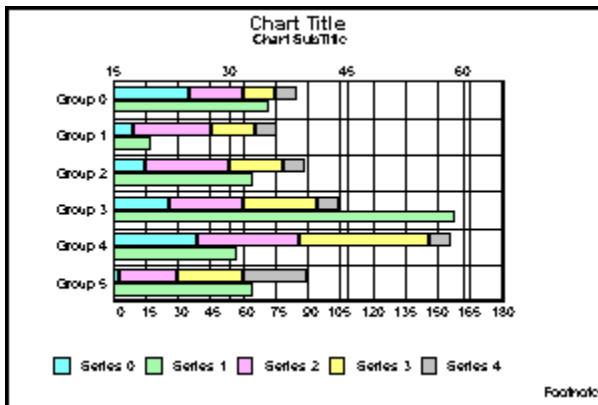
Horizontal Dual-Axis Clustered Bars (GraphType=26)

This is also called a Dual-Y graph. You can assign any series to either of the two axes.



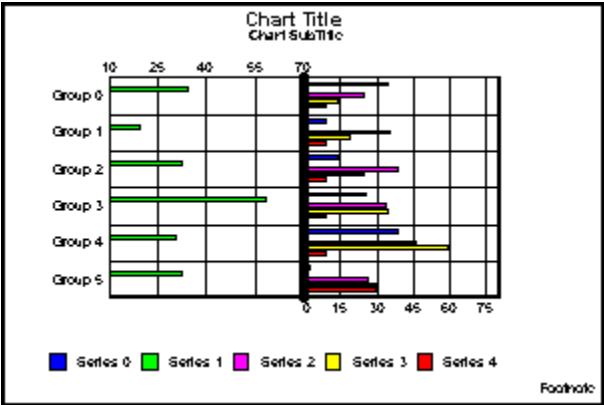
Horizontal Dual-Axis Stacked Bars (GraphType=27)

This is also called a Dual-Y stacked graph. Separate stacks will be created for the data on each axis.



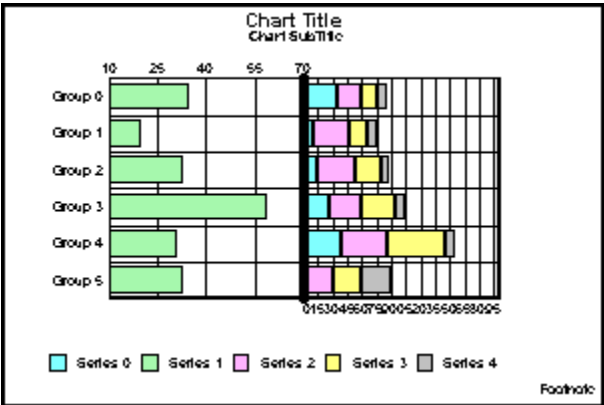
Horizontal Bi-Polar Clustered Bars (GraphType=28)

This is a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



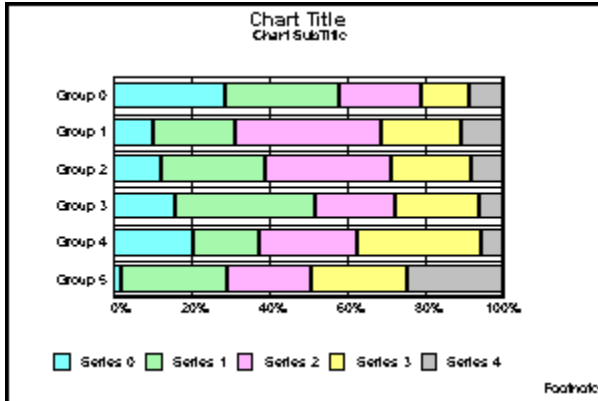
Horizontal Bi-Polar Stacked Bars (GraphType=29)

This shows a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Horizontal Percent Bars (GraphType=30)

This is a bar version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.

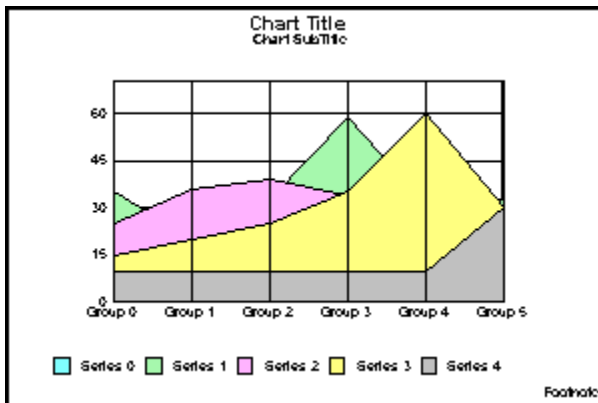


Vertical Area Graphs

The following illustrations show the vertical area graphs. All of the illustrations were done without 2.5D depth applied (for example, DepthRadius = 0).

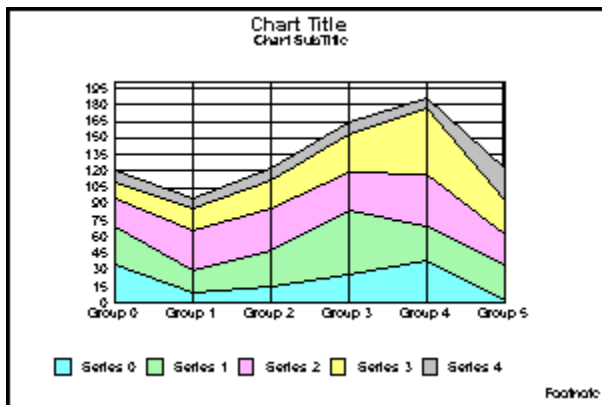
Vertical Absolute Area (GraphType=31)

This shows areas drawn on top of each other to represent the absolute relationships between data series. You can use this graph type if certain data overlaps.



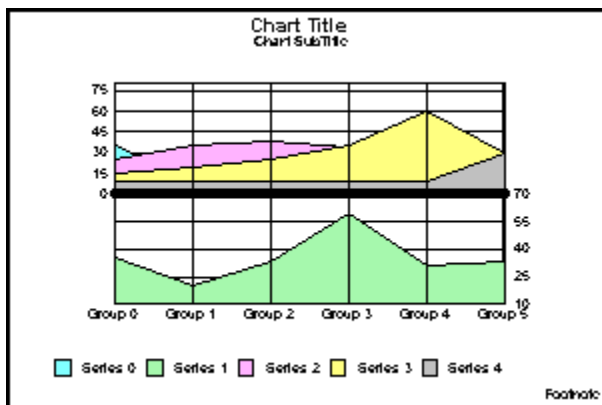
Vertical Stacked Area (GraphType=32)

This shows areas stacked on top of each other. The axis is the cumulative total of all the groups.



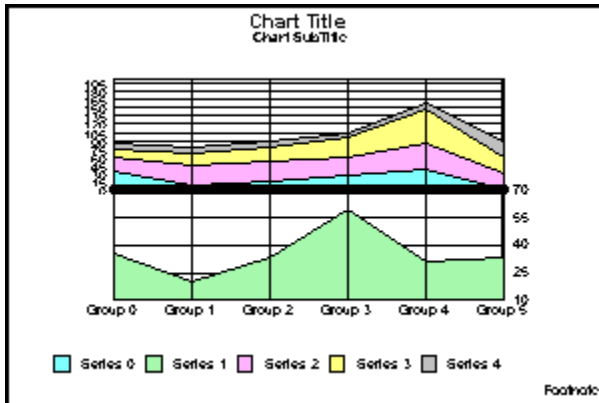
Vertical Bi-Polar Absolute Area (GraphType=33)

This shows a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



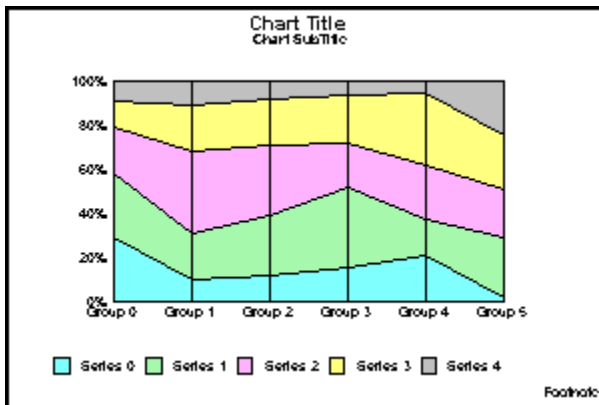
Vertical Bi-Polar Stacked Area (GraphType=34)

This shows a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Vertical Percent Area (GraphType=35)

This shows an area version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.

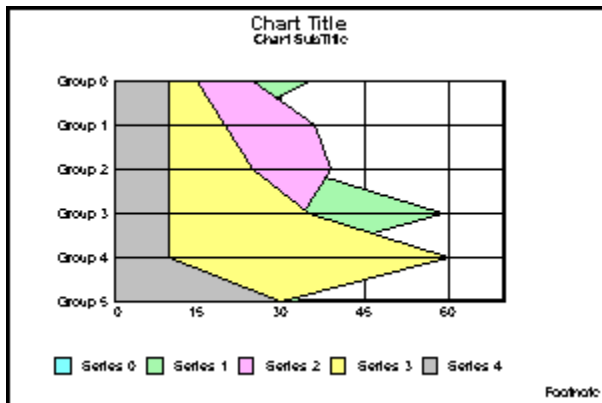


Horizontal Area Graphs

The following illustrations show the horizontal area graphs. All of the illustrations were done without 2.5D depth applied (for example, DepthRadius = 0).

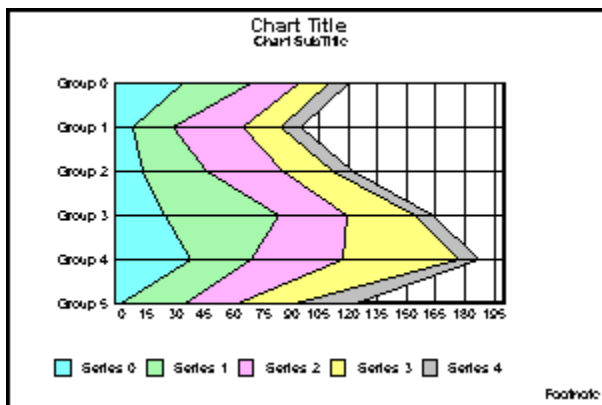
Horizontal Absolute Area (GraphType=36)

This shows areas drawn on top of each other to represent the absolute relationships between data series. You can use this graph type if certain data overlaps.



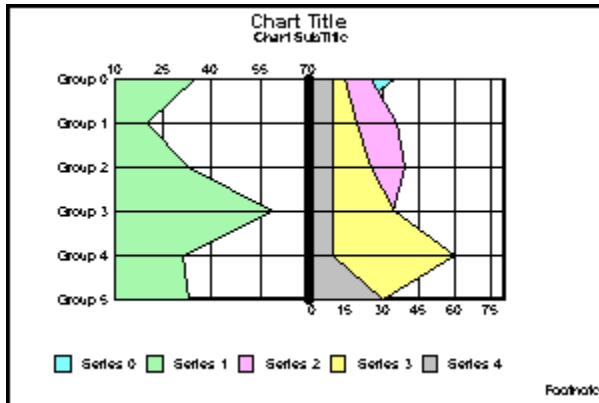
Horizontal Stacked Area (GraphType=37)

This shows areas stacked on top of each other. The axis is the cumulative total of all the groups.



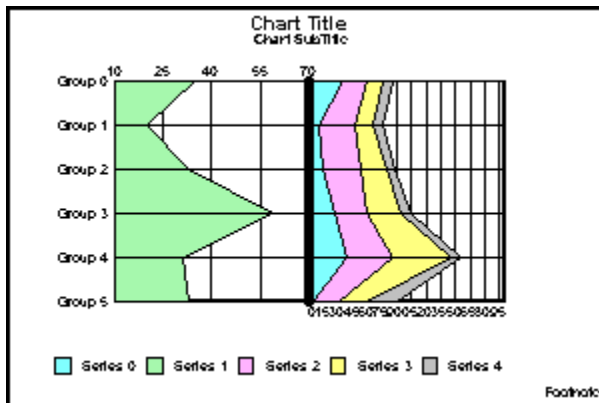
Horizontal Bi-Polar Absolute Area (GraphType=38)

This shows a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



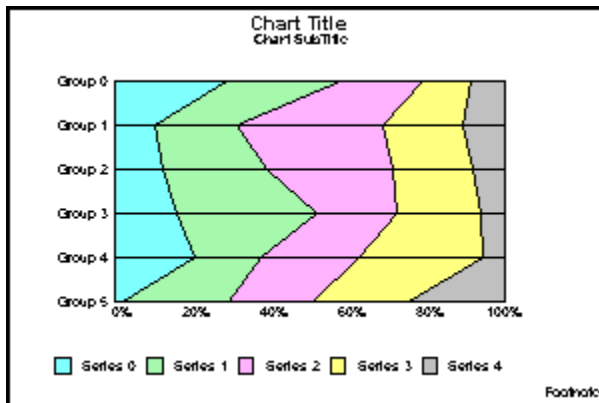
Horizontal Bi-Polar Stacked Area (GraphType=39)

This shows a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Horizontal Percent Area (GraphType=40)

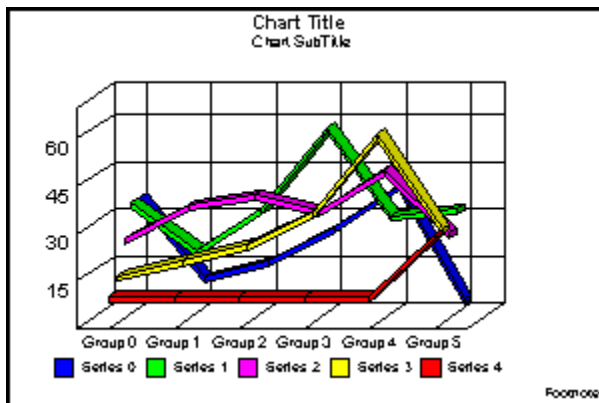
This shows an area version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.



Vertical Line Graphs

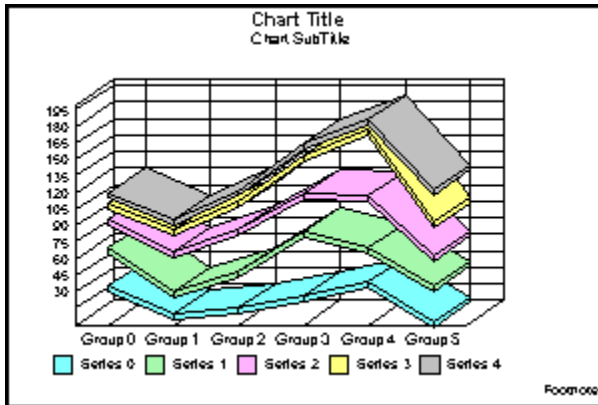
Vertical Absolute Line (GraphType=41)

This shows lines drawn on top and under each other to represent the absolute relationships between data series.



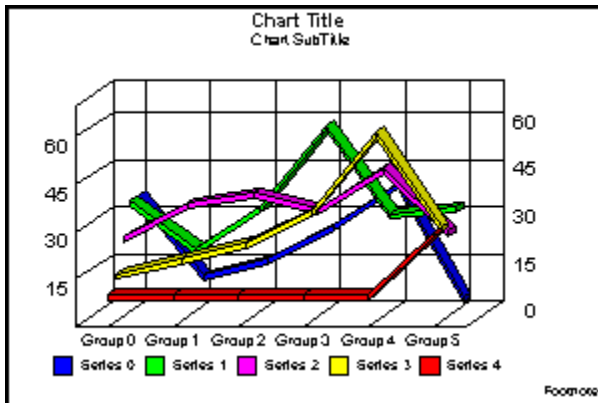
Vertical Stacked Line (GraphType=42)

This shows lines stacked on top of each other. The axis is the cumulative total of all the groups.



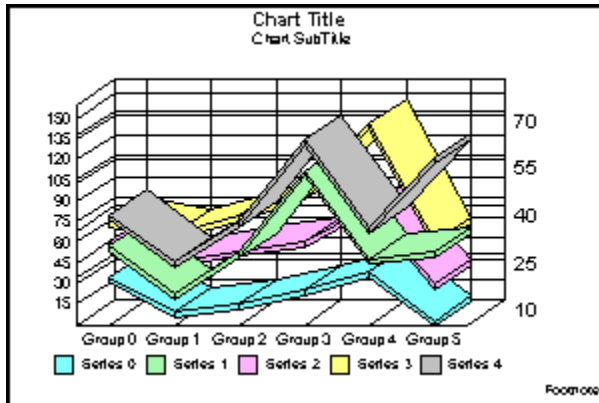
Vertical Dual-Axis Absolute Line (GraphType=43)

This is also called a Dual-Y line graph. You can assign any series to either of the two axes.



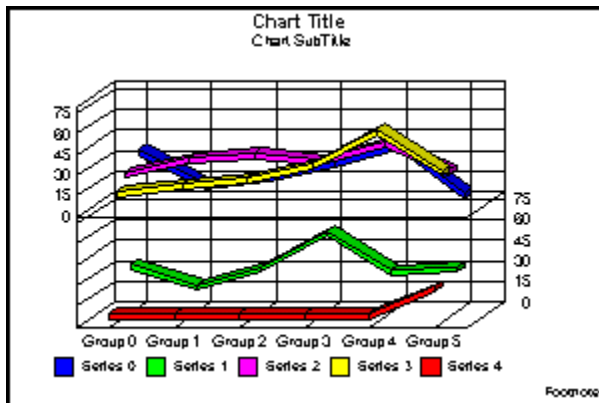
Vertical Dual-Axis Stacked Line (GraphType=44)

This is also called a Dual-Y stacked line graph. Separate stacks will be created for the data on each axis.



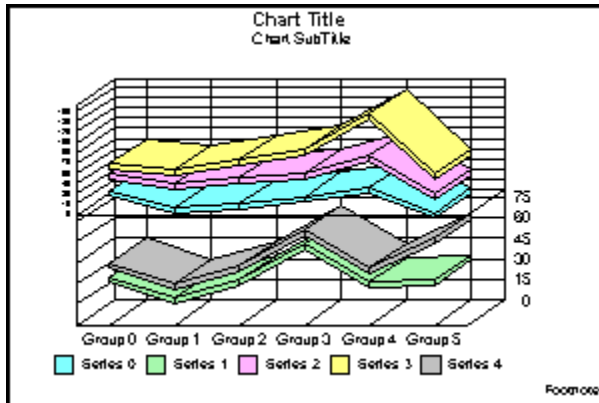
Vertical Bi-Polar Absolute Line (GraphType=45)

This is a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



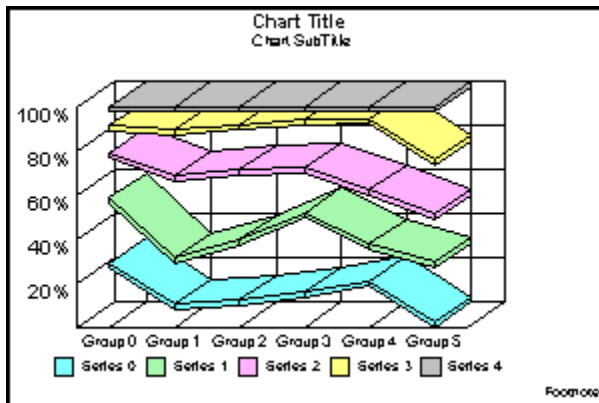
Vertical Bi-Polar Stacked Line (GraphType=46)

This shows a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Vertical Percent Line (GraphType=47)

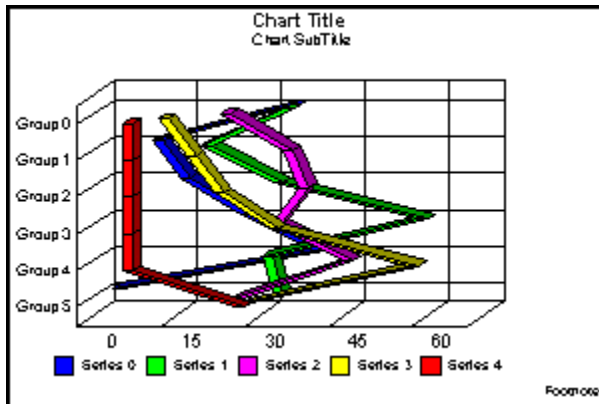
This shows a line version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.



Horizontal Line Graphs

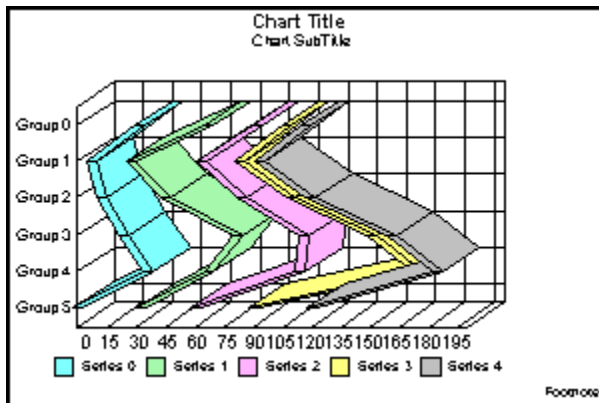
Horizontal Absolute Line (GraphType=48)

This shows lines drawn on top and under each other to represent the absolute relationships between data series.



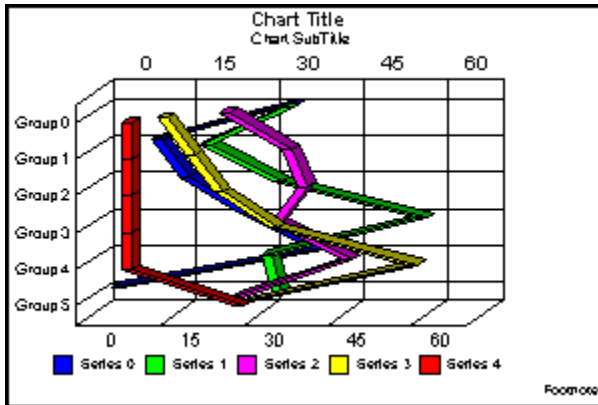
Horizontal Stacked Line (GraphType=49)

This shows lines stacked on top of each other. The axis is the cumulative total of all the groups.



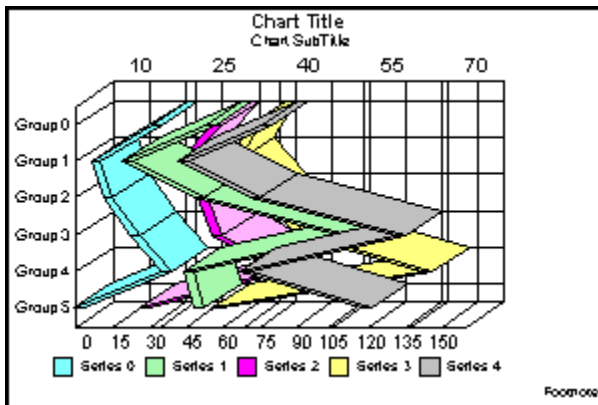
Horizontal Dual-Axis Absolute Line (GraphType=50)

This is also called a Dual-Y line graph. You can assign any series to either of the two axes.



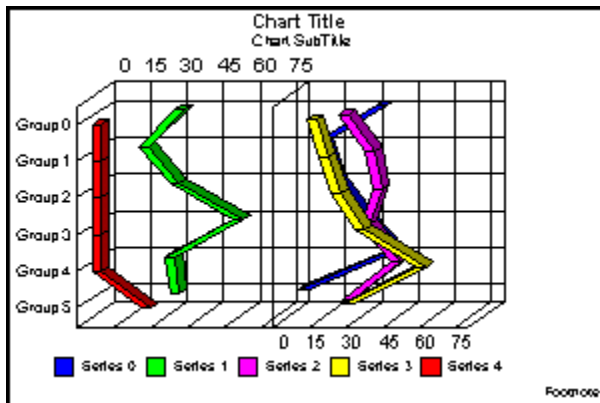
Horizontal Dual-Axis Stacked Line (GraphType=51)

This is also called a Dual-Y stacked line graph. Separate stacks will be created for the data on each axis.

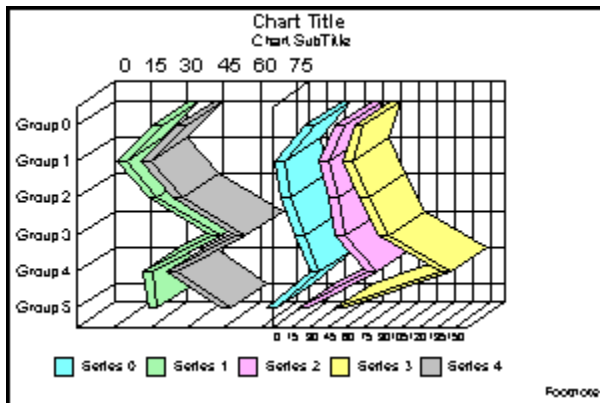


Horizontal Bi-Polar Absolute Line (GraphType=52)

This shows a Dual-Y graph with the two axes split into different sections so that each can be viewed separately.

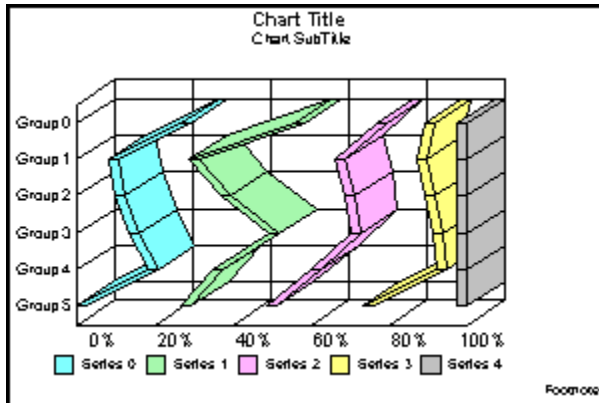
**Horizontal Bi-Polar Stacked Line (GraphType=53)**

This shows a stacked Dual-Y graph with the two axes split into different sections so that each can be viewed separately.



Horizontal Percent Line (GraphType=54)

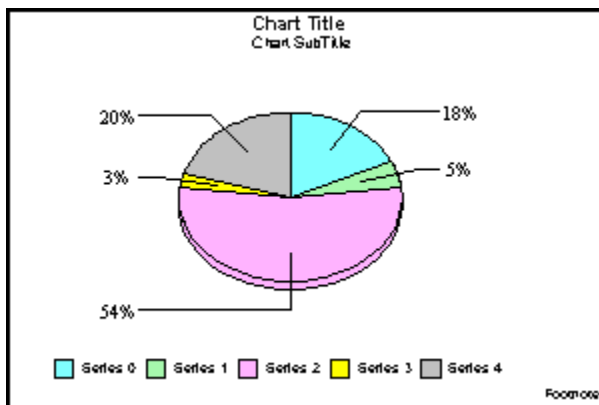
This shows a line version of a pie graph. Each group calculates the percentage of the total required for each series. The axis ranges from 0 to 100%.



Pie Graphs

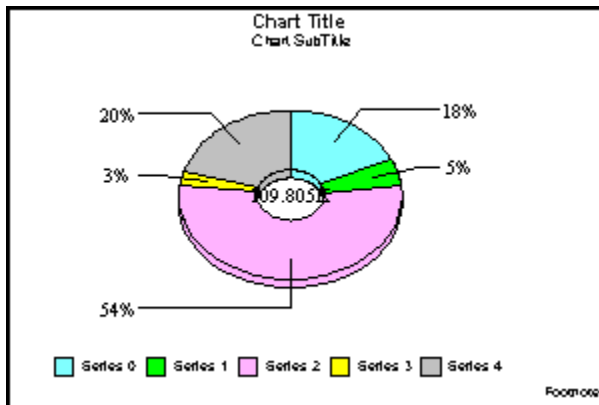
Pie (GraphType=55)

This shows the most widely used graph for displaying percentages of a total.

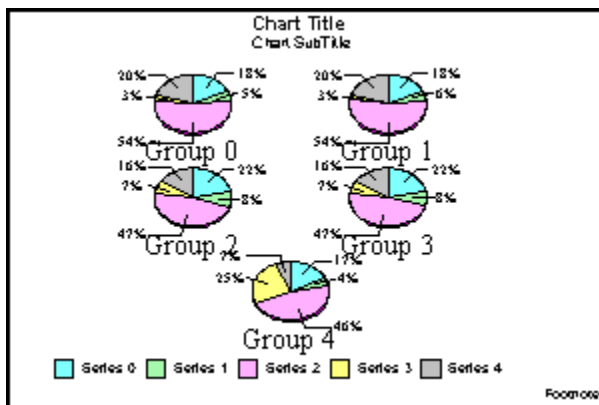


Ring Pie (GraphType=56)

This shows a ring variant of a pie graph. The total of all slices is placed in the center.

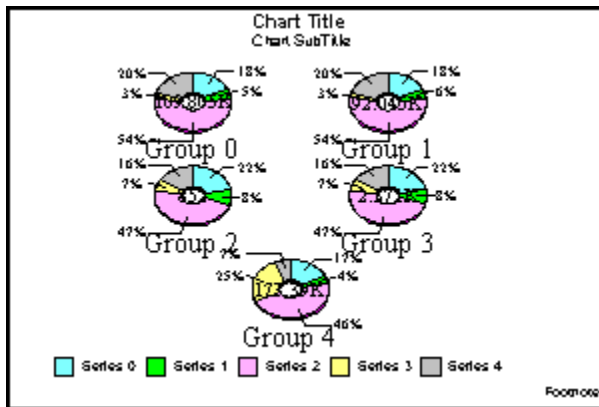
**Multi Pie (GraphType=57)**

This shows separate pies to represent each group in the data set. This is a pie variation on percentage bars.



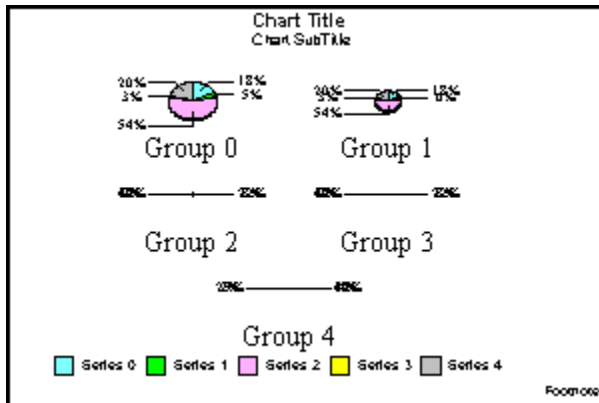
Multi Ring Pie (GraphType=58)

This shows separate ring pies to represent each group in the data set.



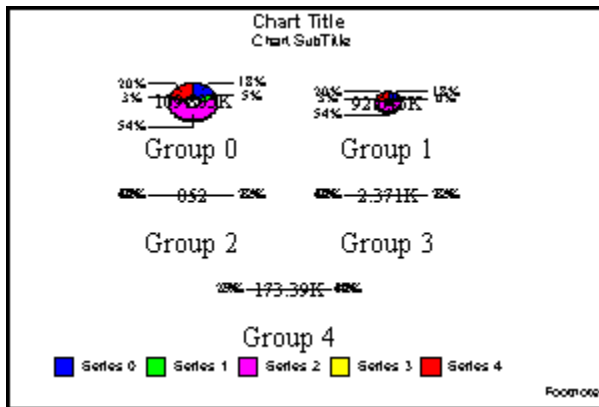
Multi Proportional Pie (GraphType=59)

This shows each pie sized in proportion to its total across the entire data set.



Multi Proportional Ring Pie (GraphType=60)

This shows each ring pie sized in proportion to its total across the entire data set.



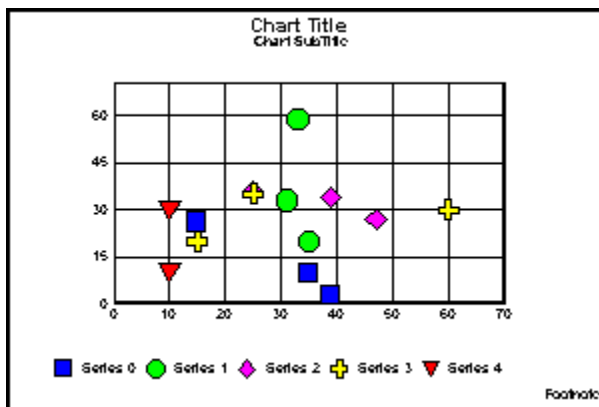
Scatter Graphs

The following illustrations show the scatter graphs. All of the illustrations were created with the MarkerSizeDefault set to 90 and UseSeriesShapes set to *True*.

Note: The X-axis for Scatter graphs is expected to be numeric. While you are allowed to substitute with an alphanumeric X-axis, not all features will work correctly.

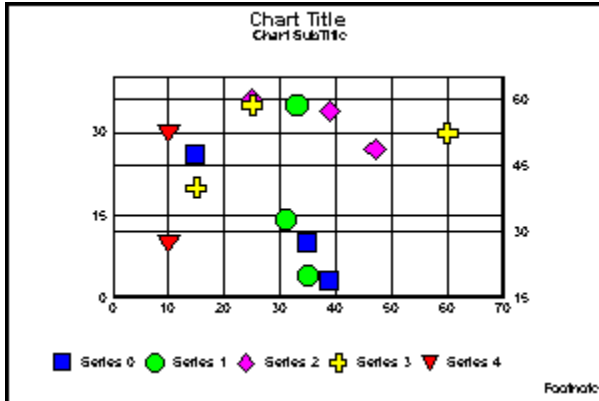
XY Scatter (GraphType=61)

This shows two values assigned to each marker, X and Y, in that order. This is a standard X-Y plot.



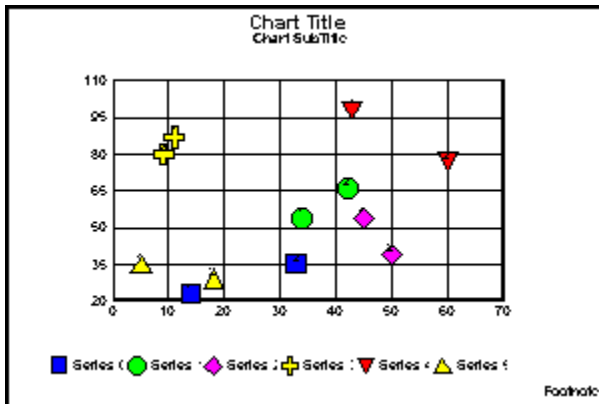
XY Scatter Dual-Axis (GraphType=62)

This is a Dual-Y scatter graph. There are two values assigned to each marker, X and Y, in that order.



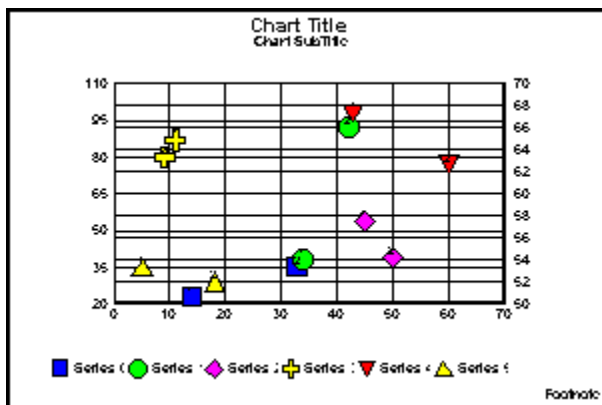
XY Scatter with Labels (GraphType=63)

This shows three values assigned to each marker—X, Y, and text label—in that order. Each XY point is labeled.



XY Scatter with Labels Dual-Axis (GraphType=64)

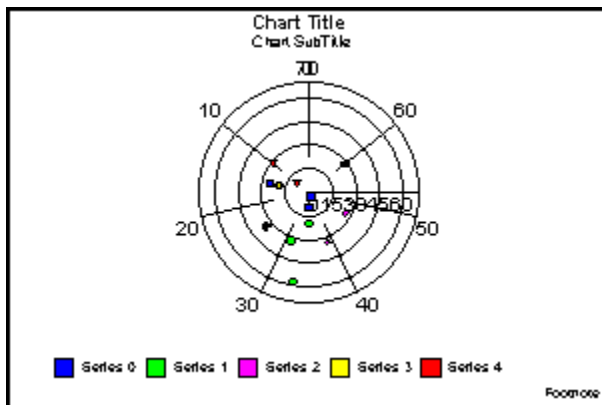
This shows a Dual-Y scatter graph with labeled markers. It requires that three values are assigned to each marker—X, Y, and text label—in that order.



Polar Graphs

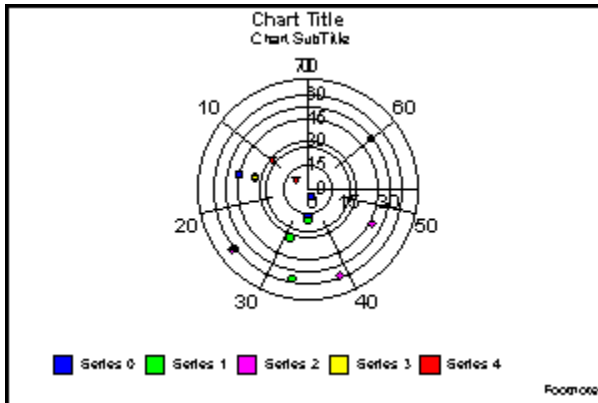
XY Polar (GraphType=65)

This shows a polar coordinate scatter graph. It requires that two values are assigned to each marker in the following order: X (degree) and Y (distance from the center).

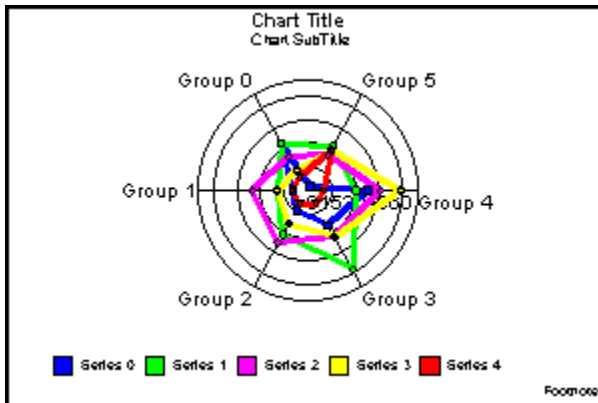


XY Polar Dual-Axis (GraphType=66)

This shows a Dual-Y polar coordinate scatter graph. It requires that two values are assigned to each marker in the following order: X (degree) and Y (distance from the center).

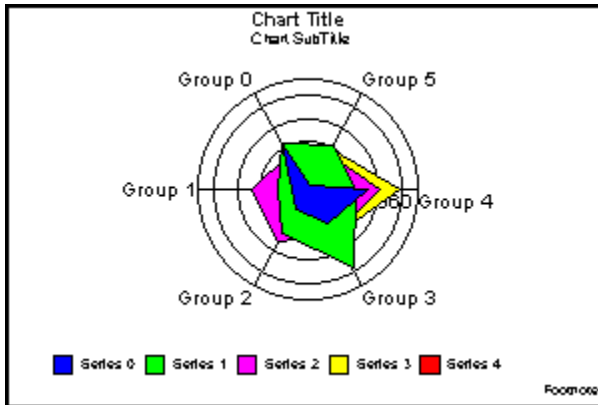
**Radar Graphs****Radar Line (GraphType=67)**

This shows a circular line graph. You can use it to represent cyclical data, such as hourly or monthly figures.

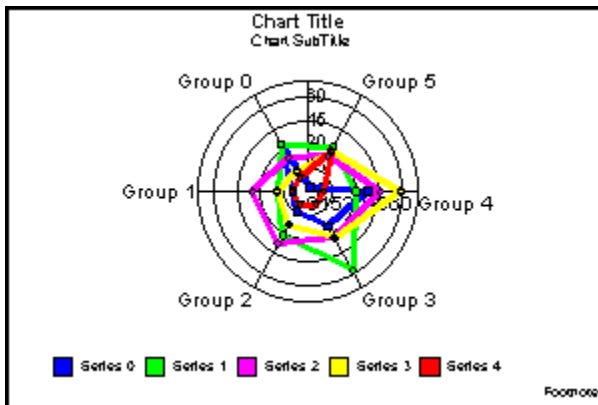


Radar Area (GraphType=68)

This shows a circular area graph. You can use it for comparisons or cyclical data sets.

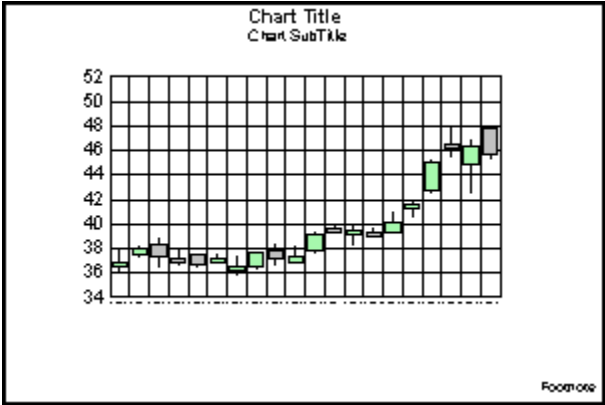
**Radar Line Dual-Axis (GraphType=69)**

This shows a Dual-Y variation on a Radar Line graph type. You can use it to show two sets of cyclical data.

**Stock Graphs****Open-Hi-Lo-Close Candle Stock Graph (GraphType=70)**

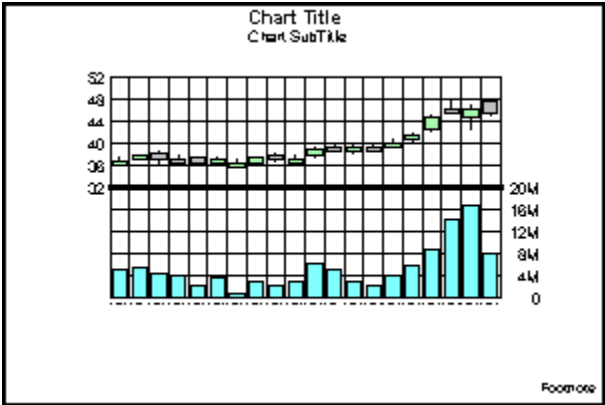
This graph type requires that four values are assigned to each marker: Open, High, Low, and Close, in that order.

The following is an example of Open-Hi-Lo-Close graph output.



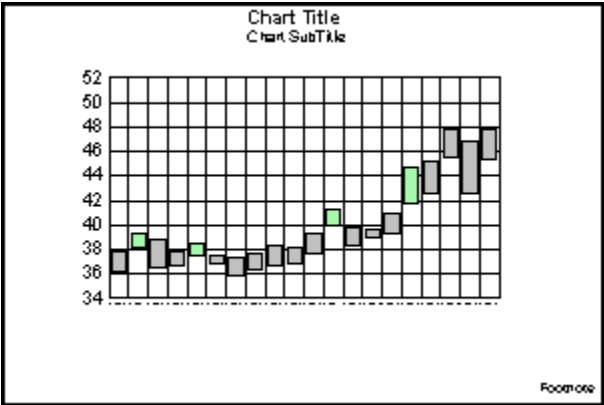
Open-Hi-Lo-Close Candle Stock Graph with Volume (GraphType=71)

This graph type requires that five values are assigned to each marker: Open, High, Low, Close, and Volume.



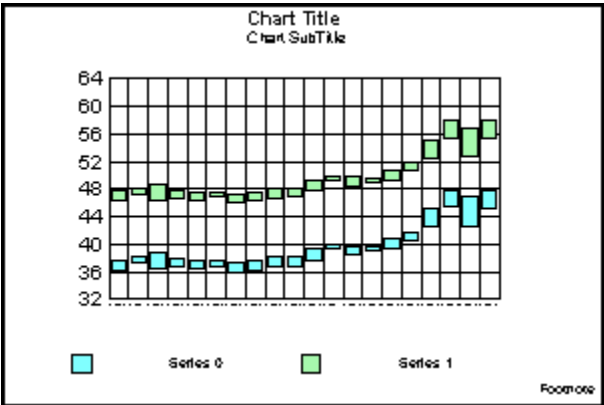
Candle Stock Open-Close (GraphType=72)

This graph type requires that two values are assigned to each marker: Open and Close.



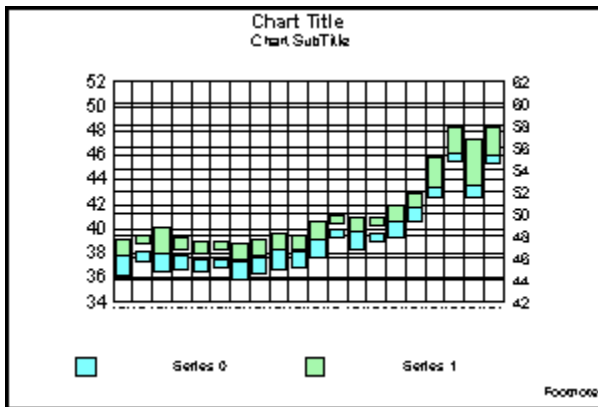
Stock Hi-Lo (GraphType=73)

This graph type requires that two values are assigned to each marker: High and Low, in that order. This is a standard financial equity graph.



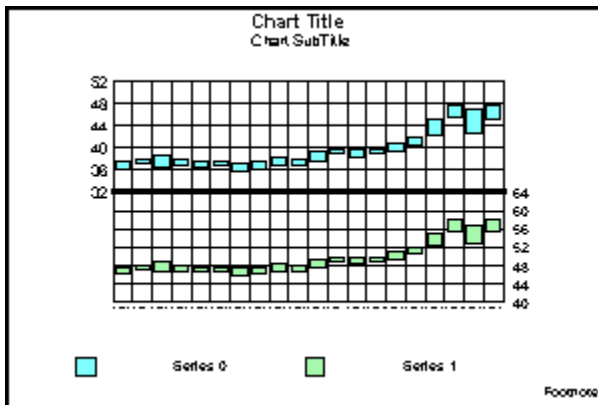
Stock Hi-Lo Dual-Axis (GraphType=74)

This shows a Dual-Y HiLo graph. It requires two values per marker: High and Low.



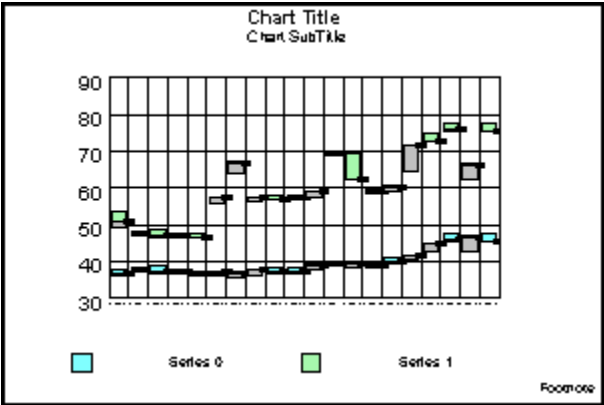
Stock Hi-Lo Bi-Polar (GraphType=75)

This shows a Dual-Y graph with the axis split into separate sections. It requires that two values are assigned to each marker: High and Low.



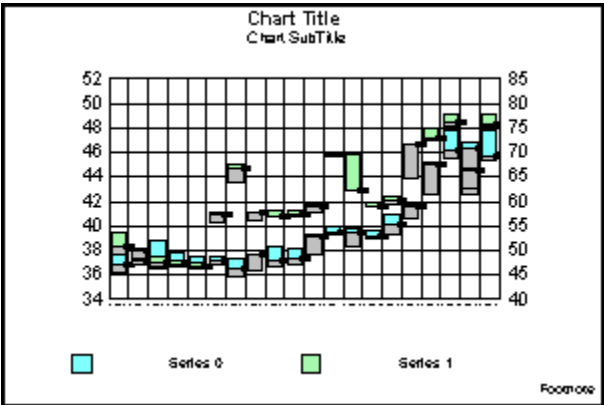
Stock Hi-Lo Close (GraphType=76)

This graph type requires that three values are assigned to each marker: High, Low, and Close, in that order. This is a standard financial equity graph.



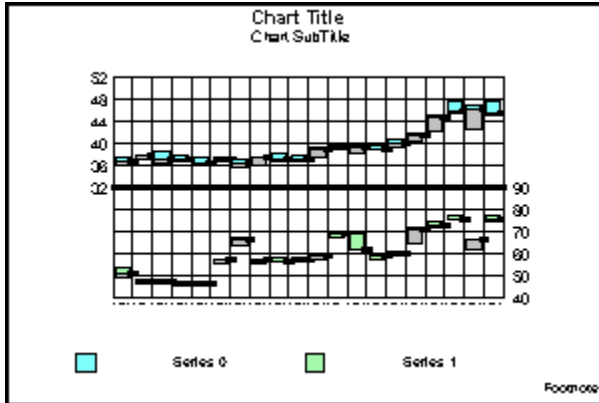
Stock Hi-Lo Close Dual-Axis (GraphType=77)

This shows a Dual-Y version of a Hi-Lo Close graph. It requires that three values are assigned to each marker: High, Low, and Close.



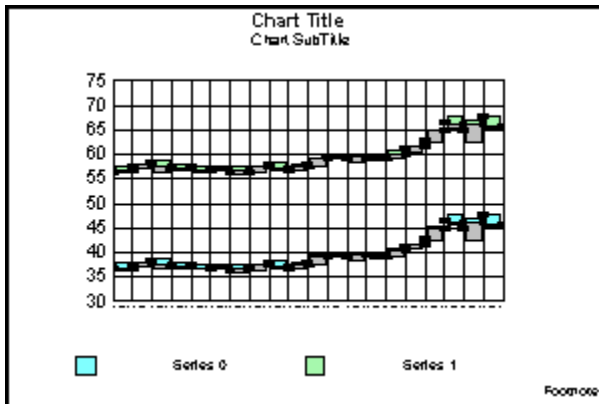
Stock Hi-Lo Close Bi-Polar (GraphType=78)

This shows a Dual-Y graph with the axis split into separate sections. It requires that three values are assigned to each marker: High, Low, and Close.



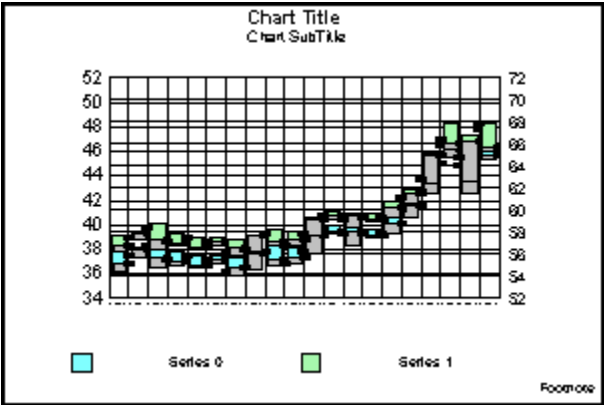
Stock Hi-Lo Open-Close (GraphType=79)

This graph type requires that four values are assigned to each marker: Open, High, Low, and Close. This is a standard financial equity graph.



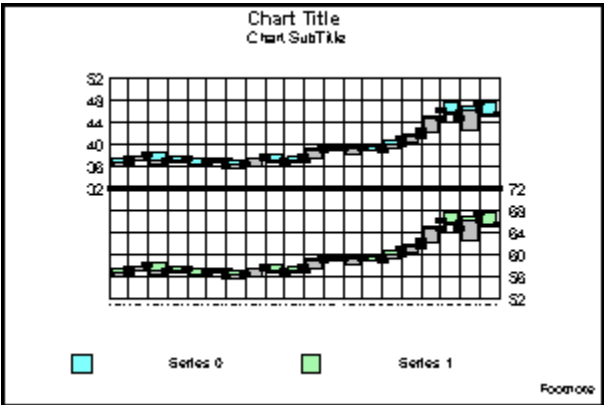
Stock Hi-Lo Open-Close Dual-Axis (GraphType=80)

This shows a Dual-Y version of GraphType 79. It requires that four values are assigned to each marker: Open, High, Low, and Close.



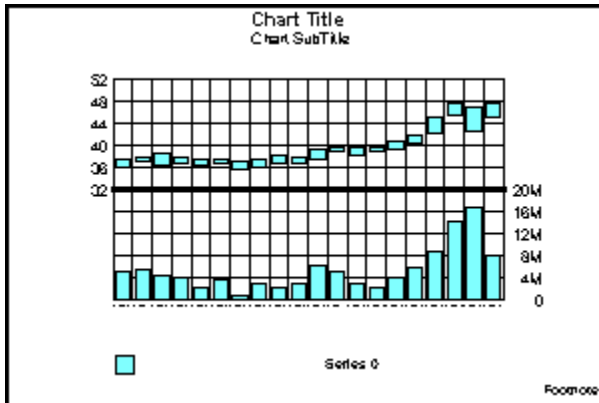
Stock Hi-Lo Open-Close Bi-Polar (GraphType=81)

This shows a Dual-Y graph with two axes split into separate sections. It requires that four values are assigned to each marker: Open, High, Low, and Close.



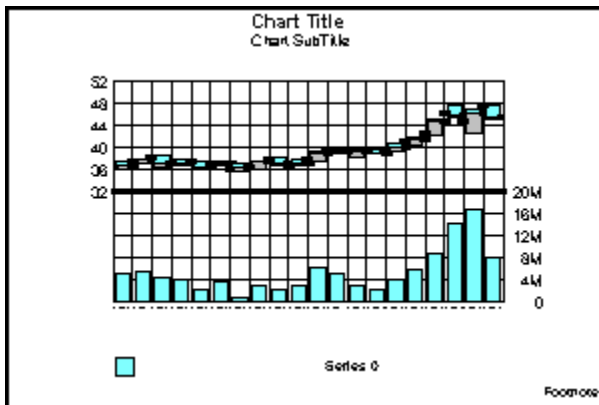
Stock Hi-Lo with Volume (GraphType=82)

This shows stock performance along with volume. It requires that three values are assigned to each marker: High, Low, and Volume.



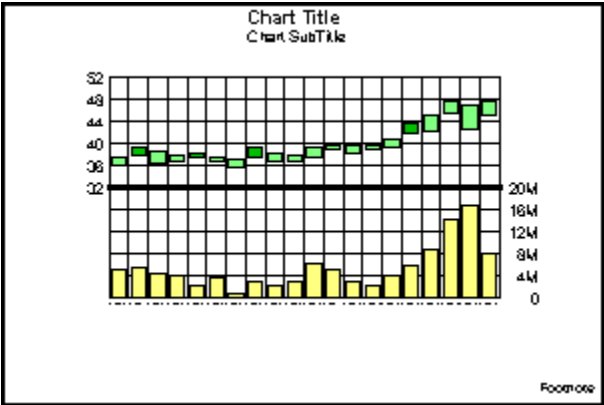
Stock Hi-Lo Open-Close with Volume (GraphType=83)

This shows stock performance along with volume. It requires that five values are assigned to each marker: Open, High, Low, Close, and Volume.



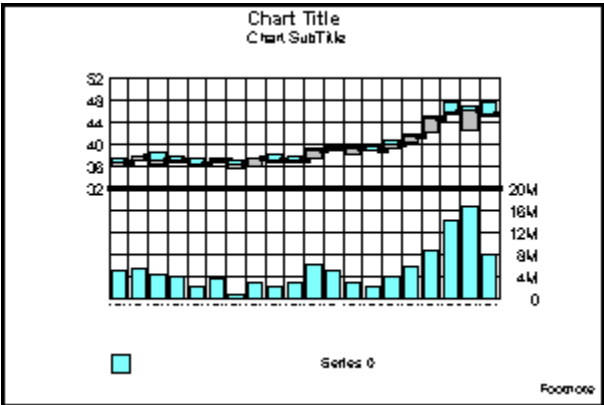
Candle Stock Open-Close with Volume (GraphType=84)

This shows a stock performance along with volume. It requires that three values are assigned to each marker: Open, Close, and Volume.



Stock Hi-Lo Close with Volume (GraphType=88)

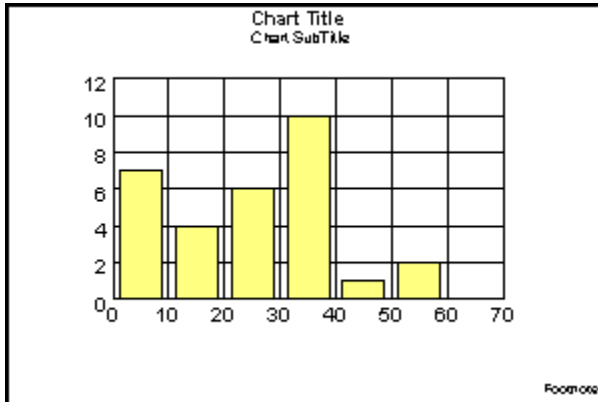
This shows stock performance along with volume. It requires that four values are assigned to each marker: High, Low, Close, and Volume.



Histogram Graphs

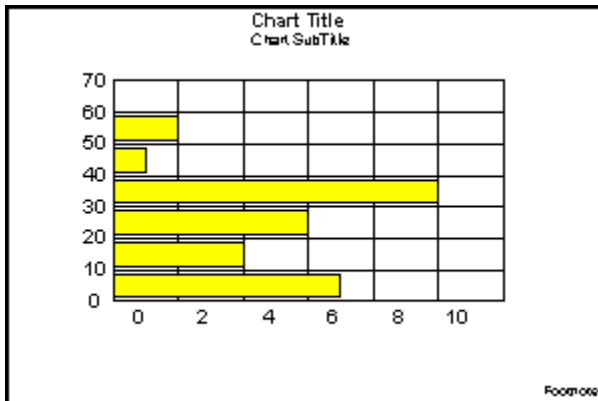
Vertical Histogram (GraphType=85)

This shows a standard histogram. It groups all of the data together and assigns it to "buckets" based on value. There are no series or groups in this graph type.



Horizontal Histogram (GraphType=86)

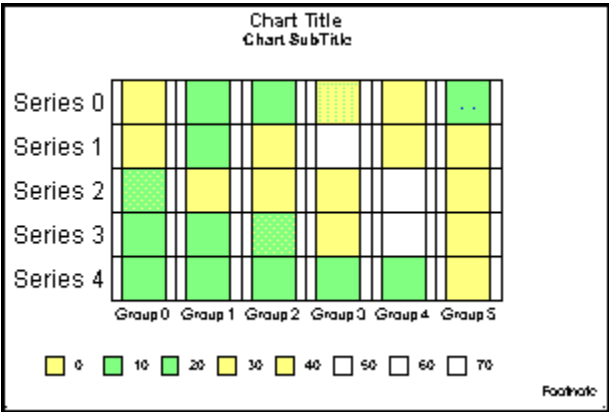
This groups all of the data together and assigns it to "buckets" based on value. There are no series or groups in this graph type.



Spectral Map

Spectral Map (GraphType=87)

This shows a row or column matrix of markers that are colored according to data values.

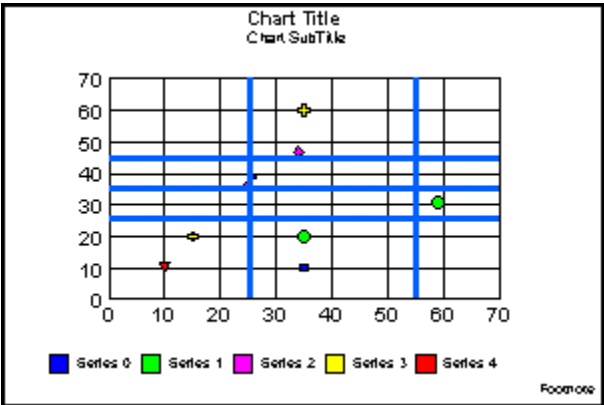


Bubble Graphs

Note: The X-axis for Bubble graphs is expected to be numeric. While you are allowed to substitute with an alphanumeric X-axis, not all features will work correctly.

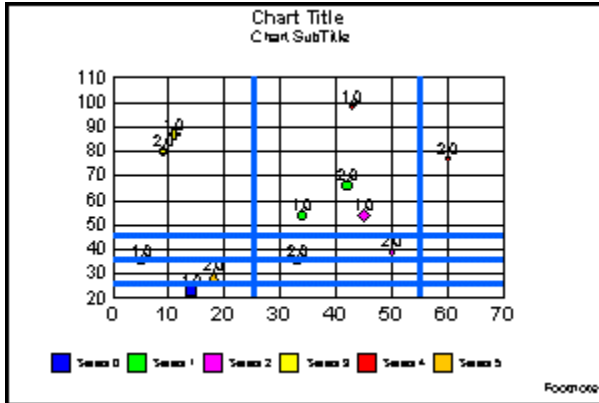
Bubble Graph (GraphType=89)

This shows three values assigned to each marker: X, Y, and Z, in that order. This has an X-Y plot in which the marker size depends on Z.



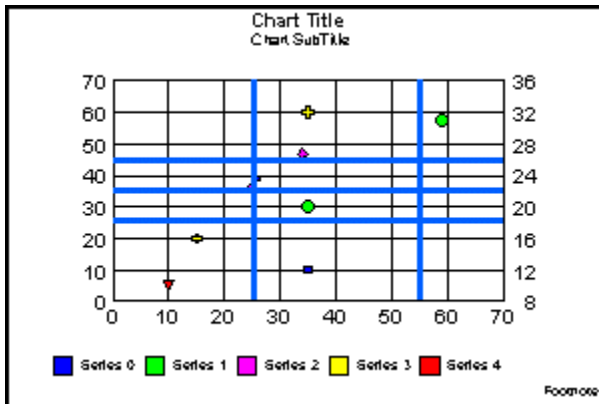
Bubble Graph with Labels (GraphType=90)

This shows that four values are assigned to each marker: X, Y, Z, and text label, in that order. This has an X-Y plot in which the marker size depends on Z, shown with labels.



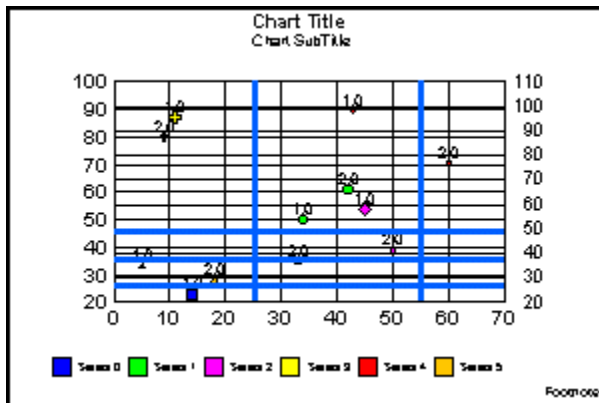
Bubble Graph Dual-Axis (GraphType=91)

This shows that three values are assigned to each marker: X, Y, and Z, in that order. This has an X-Y plot in which the marker size depends on Z.



Bubble Graph with Labels Dual-Axis (GraphType=92)

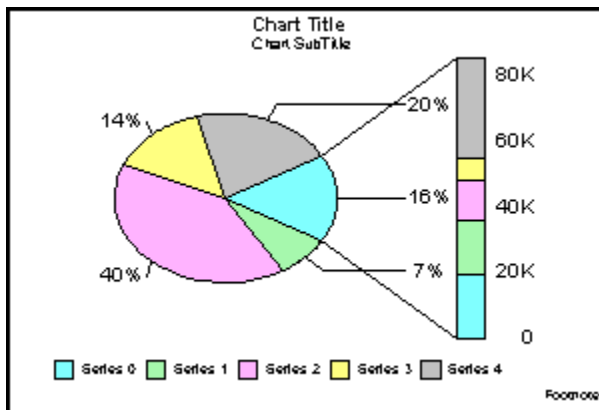
This shows four values assigned to each marker: X, Y, Z, and text label, in that order. This has an X-Y plot in which the marker size depends on Z, shown with labels.

**Pie-Bar Graphs**

The following illustrations show different types of pie-bar graphs.

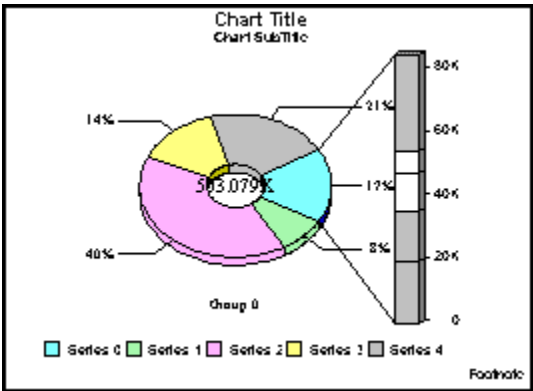
Pie-Bar Graph (GraphType=93)

This is a standard pie-bar graph.



Ring Pie-Bar Graph (GraphType=94)

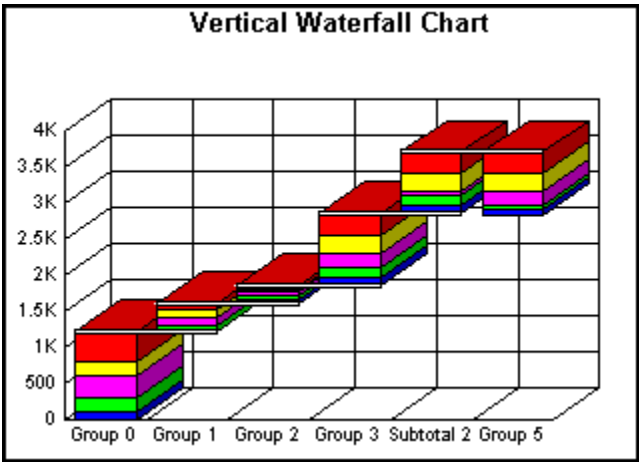
This is a ring pie-bar graph.



Special Graphs

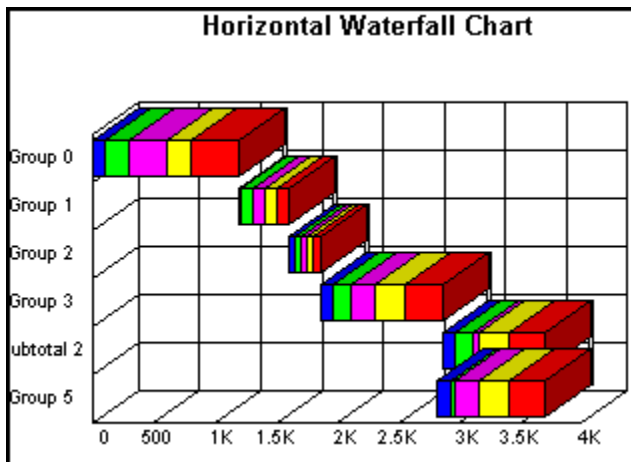
Vertical Waterfall Graph (GraphType=100)

The following is a vertical waterfall graph.



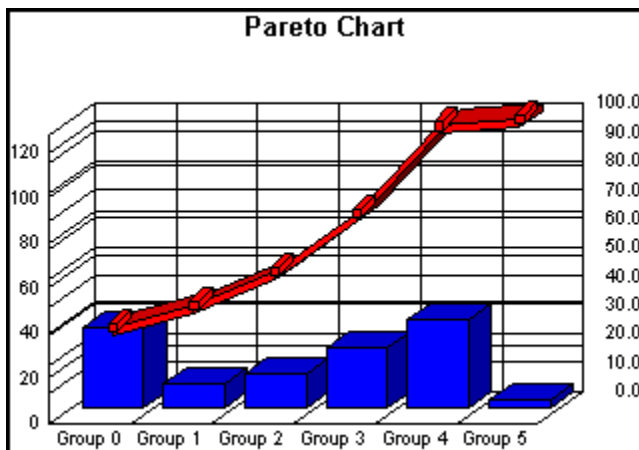
Horizontal Waterfall Graph (GraphType=101)

The following is a horizontal waterfall graph.



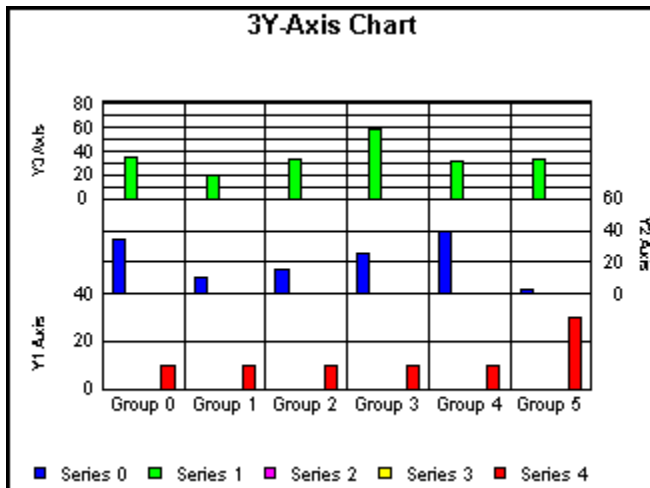
Pareto Graph (GraphType=102)

The following is a pareto graph.

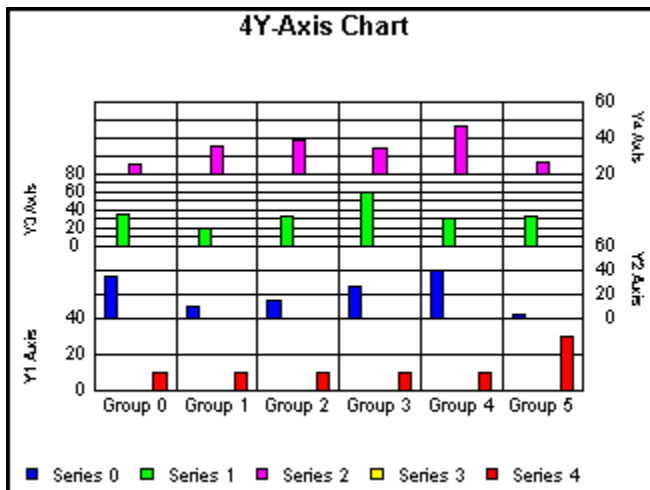


Multi-Y Y1/Y2/Y3-Axes Graph (GraphType=103)

This shows a vertical bar graph with three Y-axes.

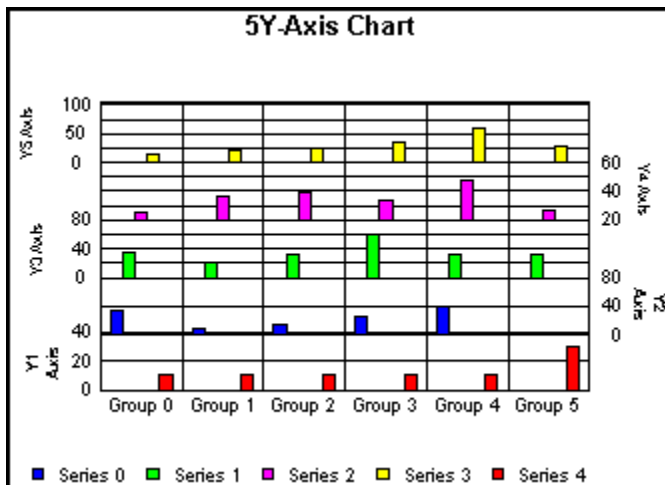
**Multi-Y Y1/Y2/Y3/Y4-Axes Graph (GraphType=104)**

This shows a vertical bar graph with four Y-axes.



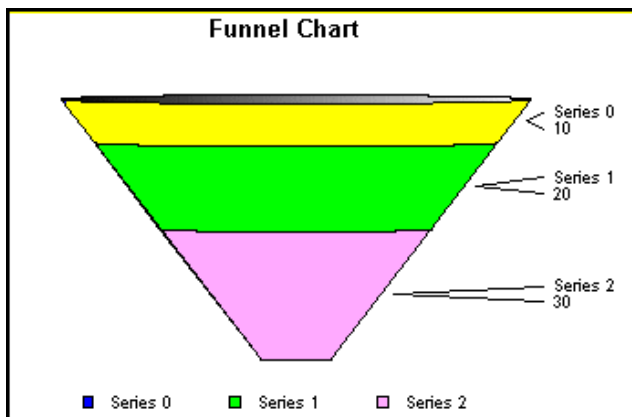
Multi-Y Y1/Y2/Y3/Y4/Y5-Axes Graph (GraphType=105)

This shows a vertical bar graph with five Y-axes.



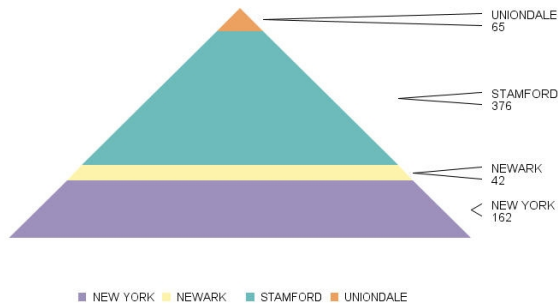
Funnel Graph (GraphType=106)

This shows a funnel graph that draws only one group of data at a time.



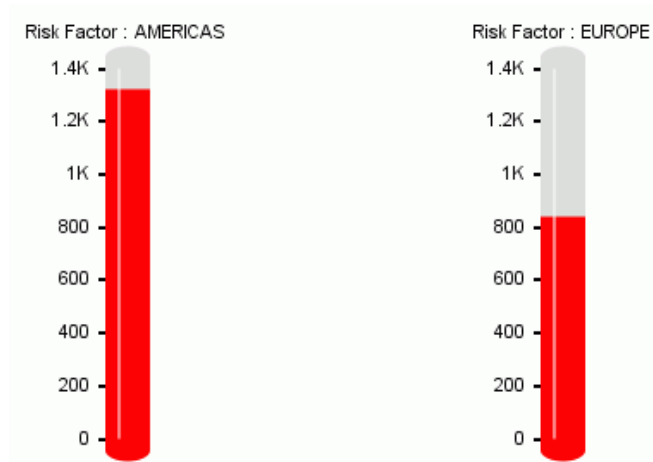
Pyramid Graph (GraphType=111)

This shows a pyramid graph that draws only one group of data at a time.



Thermometer Gauge Graph (GraphType=138)

This shows a thermometer gauge graph that compares two types of data.

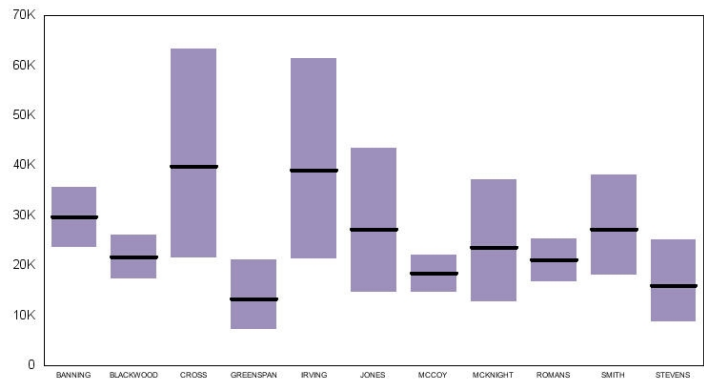


Box Plot Graphs

The following illustrations show the two types of box plot graphs. Box plot graphs can be displayed as either a box graph, or a whisker graph. For more information, see [BoxPlotType](#) on page 256.

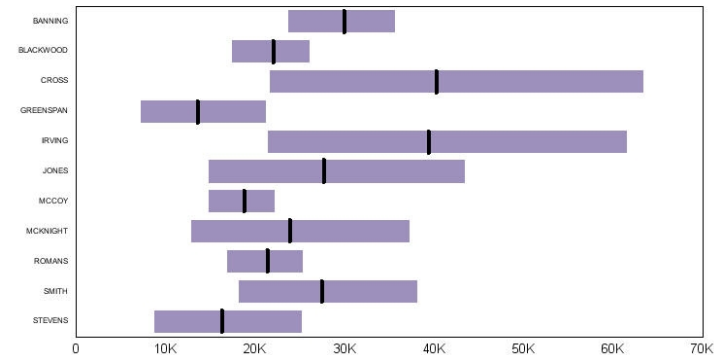
Box Plot Graph (GraphType=124)

This is a standard box plot graph.



Horizontal Box Plot Graph (GraphType=130)

This is a horizontal box plot graph.



3D Preset Viewing Angles

The [Viewing3DAnglePreset](#) on page 571 property selects an entry in a table of preset viewing angles for 3D graphs.

It can be set to one of the following:

-1 = Custom (as set by CubeViewerX/Y/Z, CubeWallThickX/Y/Z, and so on.)

0 = *Standard* on page 1271

1 = *Tall and Skinny* on page 1272

2 = *From the Top* on page 1272

3 = *Distorted* on page 1273

4 = *Short and Fat* on page 1273

5 = *Group's Eye* on page 1274

6 = *Group Emphasis* on page 1274

7 = *Few Series* on page 1275

8 = *Few Groups* on page 1275

9 = *Distorted Standard* on page 1276

10 = *Shorter and Fatter* on page 1276

11 = *Thick Wall for Series* on page 1277

12 = *Thick Wall Standard* on page 1277

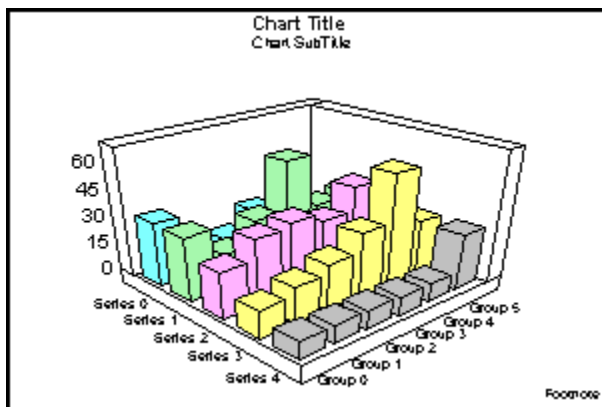
13 = *California Special* on page 1278

14 = *Blast-O-Vision* on page 1278

15 = *Special* on page 1279

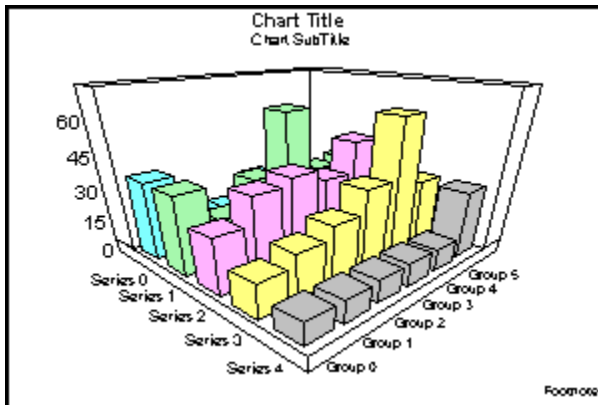
Standard

```
setViewing3DAnglePreset(0);
```



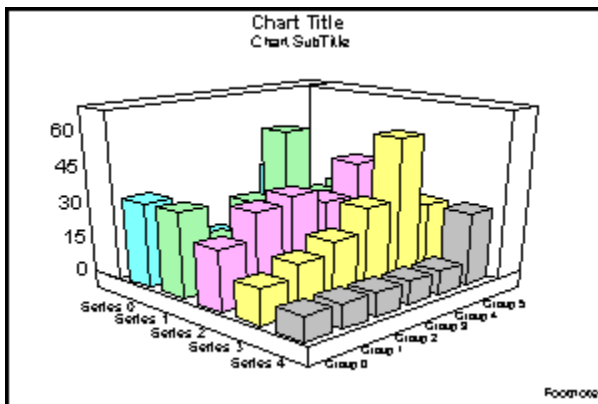
Tall and Skinny

```
setViewing3DAnglePreset(1);
```



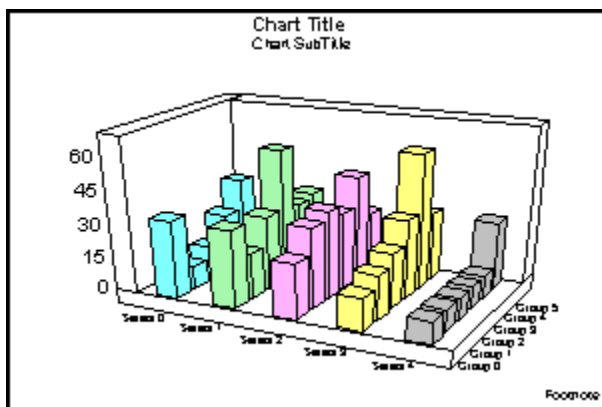
From the Top

```
setViewing3DAnglePreset(2);
```

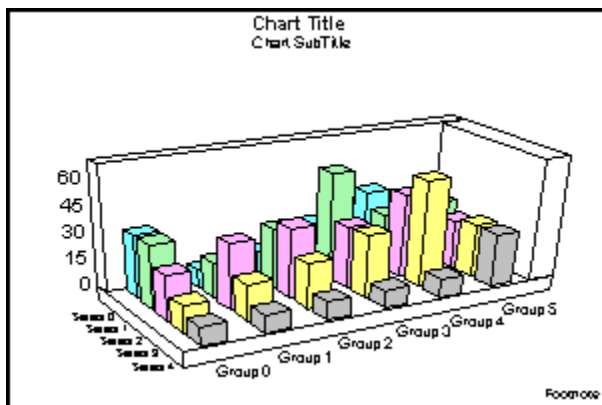


Distorted

```
setViewing3DAnglePreset(3);
```

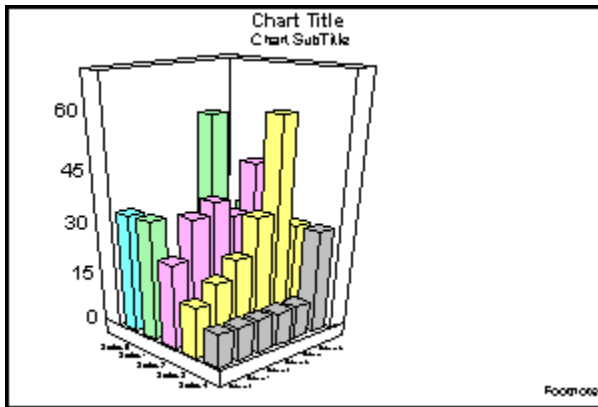
**Short and Fat**

```
setViewing3DAnglePreset(4);
```



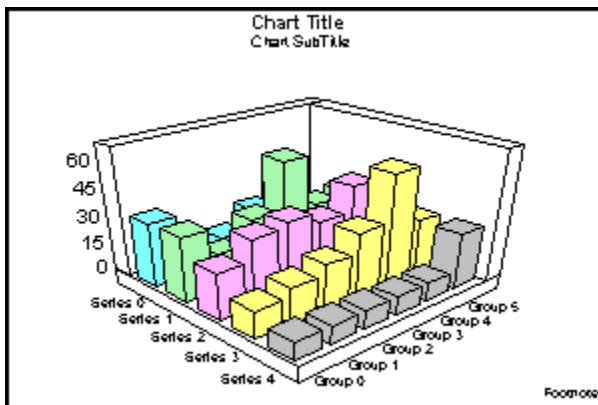
Group's Eye

```
setViewing3DAnglePreset(5);
```



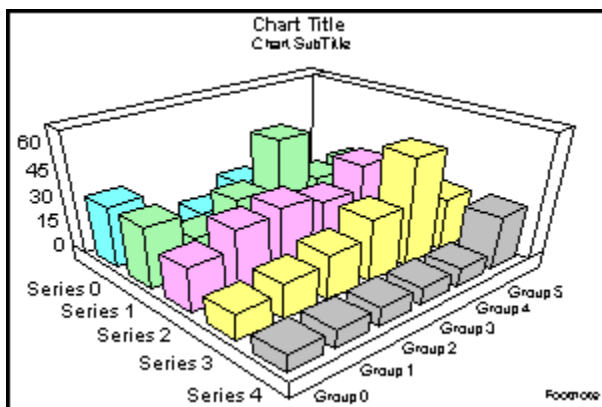
Group Emphasis

```
setViewing3DAnglePreset(6);
```

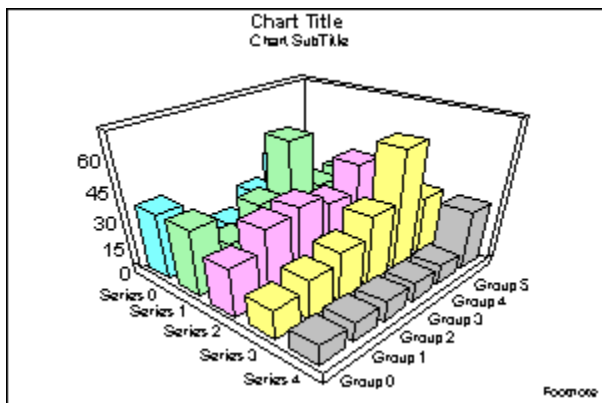


Few Series

```
setViewing3DAnglePreset(7);
```

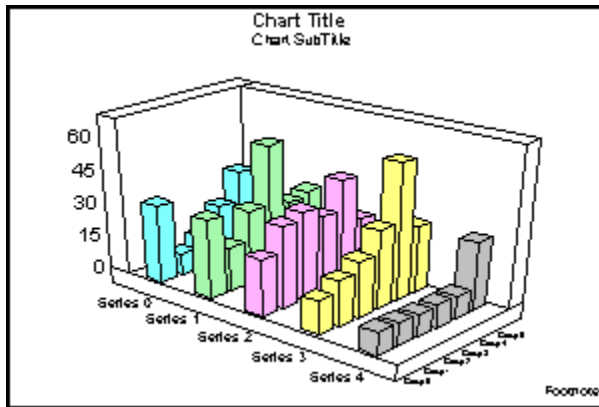
**Few Groups**

```
setViewing3DAnglePreset(8);
```



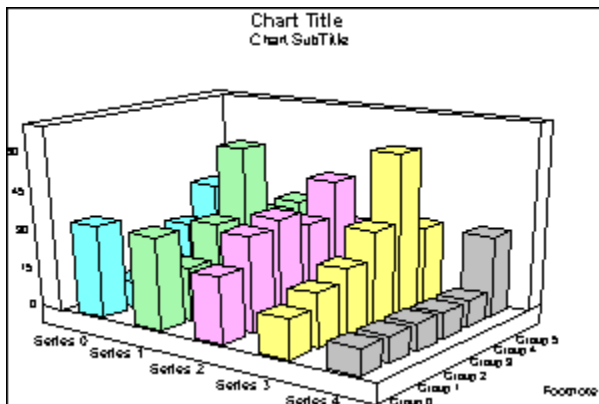
Distorted Standard

```
setViewing3DAnglePreset(9);
```



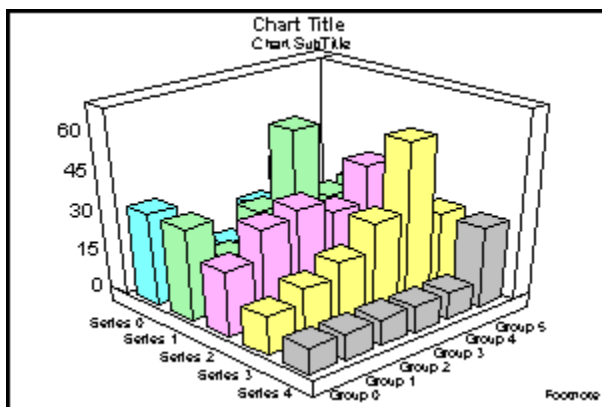
Shorter and Fatter

```
setViewing3DAnglePreset(10);
```

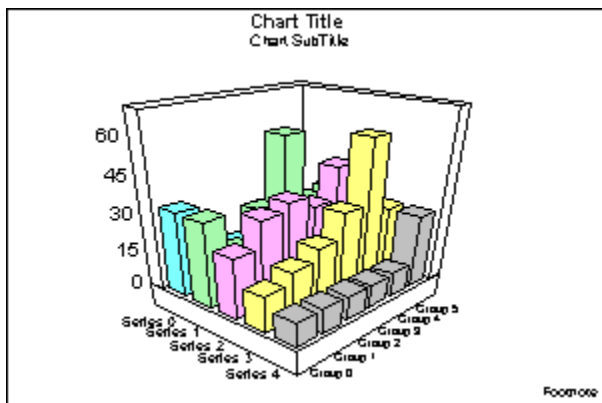


Thick Wall for Series

```
setViewing3DAnglePreset(11);
```

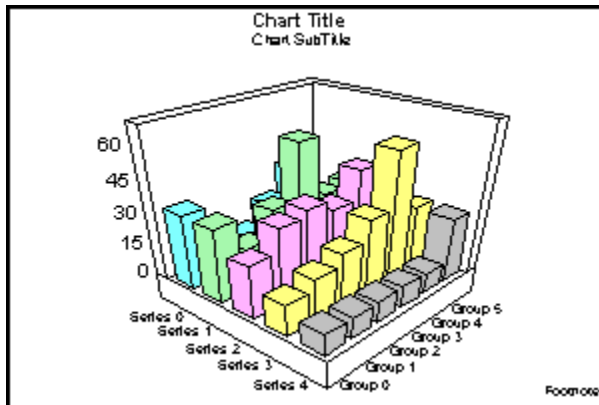
**Thick Wall Standard**

```
setViewing3DAnglePreset(12);
```



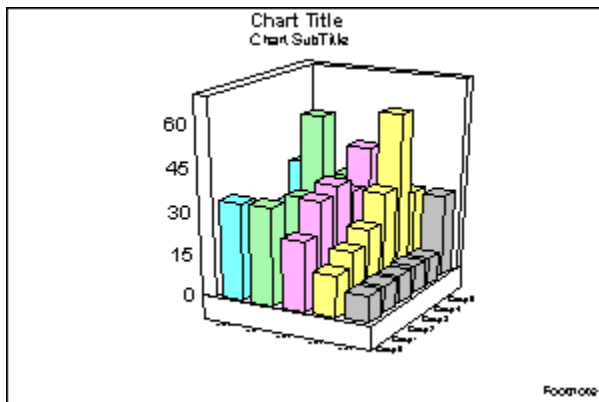
California Special

```
setViewing3DAnglePreset(13);
```



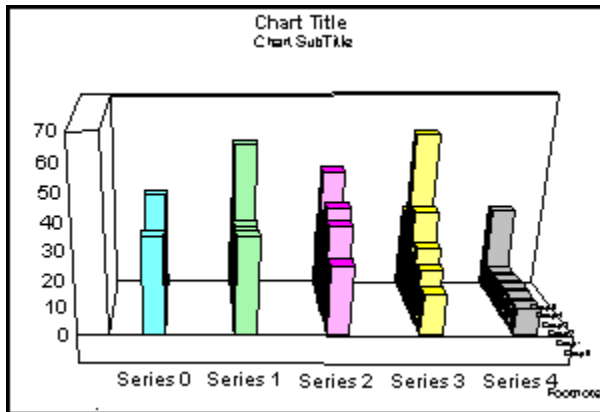
Blast-O-Vision

```
setViewing3DAnglePreset(14);
```



Special

```
setViewing3DAnglePreset(15);
```



Graph Properties Index

This appendix describes properties of graph types.

In this appendix:

- ❑ [Current Graph Properties](#)
- ❑ [Graph Properties Reserved For Future Use](#)
- ❑ [Graph Properties Reserved for Internal Use](#)

Current Graph Properties

The graph properties listed in this table are described in [Properties](#).

Properties	Data Type	Range	Default Value	Description
AdjustAutoScaleForRefLines	boolean	true/false	false	Enables/disables automatic adjustment of the scale if necessary to make reference lines display on the graph.
AlignMultiYBars	boolean	true/false	false	Aligns bars in a Multi-Y graph.
AltFmtMatchAxisLabelColor	boolean	true/false	false	Enables/disables whether Y-axis labels inherit the color of alternate frame regions.
AutoHideDataTextOnSmallRisers	boolean	true/false	false	Suppresses data text in risers when the height of the riser is smaller than the height of the text.
Autoshading	boolean	true/false	true	Enables/disables automatic shading.
BiDirectional	boolean	true/false	false	Reverses horizontal order of legends.

Properties	Data Type	Range	Default Value	Description
<i>Cascade</i>	boolean	true/false	true	Determines if properties are interrelated.
<i>ColorMode</i>	integer	0...5	0	Defines the graph color mode or riser color/styling mode.
<i>ConnectLineMarkers</i>	boolean	true/false	true	Connects markers in a line graph.
<i>ConnectParetoBars</i>	boolean	true/false	true	Connects bars in a Pareto graph.
<i>ConnectScatterMarkers</i>	boolean	true/false	false	Connects markers in a scatter graph.
<i>CubeFocusFactor</i>	double	0...100	50.0	Defines the 3D-cube focus factor.
<i>CubeIsometricProjection</i>	boolean	true/false	false	Enables/disables 3D-cube isometric projection.
<i>CubeLightSourceX</i>	double	1...100	0.0	Defines the 3D-cube light source for the Xcoordinate.
<i>CubeLightSourceY</i>	double	1...100	40.0	Defines the 3D-cube light source for the Ycoordinate.
<i>CubeLightSourceZ</i>	double	1...100	100.0	Defines the 3D-cube light source for the Z coordinate.
<i>CubePanX</i>	double	0...100	39.3	Pans a 3D-cube in the X direction.
<i>CubePanY</i>	double	0...100	60.7	Pans a 3D-cube in the Y direction.
<i>CubeRiserInterpolation</i>	integer	0...2	0	Controls whether risers in 3D graphs are interpolated (faster) or explicitly calculated (more accurate).

Properties	Data Type	Range	Default Value	Description
<i>CubeSizeX</i>	double	0...100	30.0	Defines the 3D-cube size in the X direction.
<i>CubeSizeY</i>	double	0...100	25.0	Defines the 3D-cube size in the Y direction.
<i>CubeSizeZ</i>	double	0...100	30.0	Defines the 3D-cube size in the Z direction.
<i>CubeSquareRisers</i>	boolean	true/false	true	Enables/disables square risers in a 3D-cube.
<i>CubeTranslationX</i>	double	0...100	65.0	Defines the 3D-cube translation in the X direction.
<i>CubeTranslationY</i>	double	0...100	42.5	Defines the 3D-cube translation in the Y direction.
<i>CubeTranslationZ</i>	double	0...100	50.0	Defines the 3D-cube translation in the X direction.
<i>CubeViewerX</i>	double	0...100	50.0	Defines the 3D-cube viewer in the X direction.
<i>CubeViewerY</i>	double	0...100	60.0	Defines the 3D-cube viewer in the Y direction.
<i>CubeViewerZ</i>	double	0...100	36.8	Defines the 3D-cube viewer in the Z direction.
<i>CubeWallThickX</i>	double	0...100	20.0	Defines the 3D-cube wall thickness in the X direction.
<i>CubeWallThickY</i>	double	0...100	20.0	Defines the 3D-cube wall thickness in the Y direction.
<i>CubeWallThickZ</i>	double	0...100	20.0	Defines the 3D-cube wall thickness in the Z direction.

Properties	Data Type	Range	Default Value	Description
<i>CubeZoomFactor</i>	double	0...100	50.0	Defines the 3D-cube zoom factor.
<i>CurveFitEquationDisplay</i>	boolean	true/false	true	Shows/hides equations with a curve-fit line.
<i>CurveFitHighOrderFirst</i>	boolean	true/false	true	Defines the order of curved fit equation text.
<i>CurveFitNumSegments</i>	integer	Any positive integer	50	Sets the number of interpolated points to draw a curved fit line.
<i>DataItemsAlongSeries</i>	boolean	true/false	true	Enables/disables data items along a series.
<i>DataLineThickness</i>	integer	1...100	30	Defines the thickness of a data line.
<i>DataScrollerPresenceGroups</i>	integer	0...2	0	Enables a data scroller bar for groups.
<i>DataScrollerPresenceSeries</i>	integer	0...2	0	Enables a data scroller bar for a series.
<i>DataSortType</i>	integer	0/1	0	Sorts data from high to low in Pareto charts only.
<i>DataTextAngleDefault</i>	integer	0...360	90	Defines the default data text angle.
<i>DataTextDisplay</i>	boolean	true/false	false	Shows/hides data text.
<i>DataTextFormat</i>	integer	-1...21	0	Selects a preset data text format.
<i>DataTextFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a standard Java format pattern for displaying data text.

Properties	Data Type	Range	Default Value	Description
<i>DataTextPosition</i>	integer	0...5	1	Defines the data text position.
<i>DataTextRadiusDefault</i>	integer	0...360	20	Defines the default data text radius.
<i>DataTextStackedTotalOnTop()</i>	String	true/false	false	Enables/disables the display of cumulative values on top of the stacked riser.
<i>DataTextTemplateBubble</i>	String	Defined macros and any printable characters	[size]	Defines a special format for data text in a bubble graph.
<i>DataTextTemplateHiLo</i>	String	Defined macros and any printable characters	Null String	Defines a special format for data text in a stock graph.
<i>DataTextTemplateScatter</i>	String	Defined macros and any printable characters	[X],[Y]	Defines a special format for data text in a scatter graph.
<i>DepthAngle</i>	integer	0...180	45	Defines the depth angle of the graph.
<i>DepthRadius</i>	integer	0...100	40	Defines the depth radius of the graph.
<i>Display3DFloor</i>	boolean	true/false	true	Shows/hides a 3D-cube floor.
<i>Display3DLeftWall</i>	boolean	true/false	true	Shows/hides a 3D-cube left wall.
<i>Display3DRightWall</i>	boolean	true/false	true	Shows/hides a 3D-cube right wall.

Properties	Data Type	Range	Default Value	Description
<i>DisplayBarAsPictograph</i>	boolean	true/false	false	Shows/hides images on bars.
<i>DisplayFunnelGroup</i>	integer	0...# of groups	0	Selects groups to show in a funnel graph.
<i>DisplayParetoSeries</i>	integer	0...# of groups	0	Selects series to show in a Pareto graph.
<i>DrawAreaMarkers</i>	boolean	true/false	false	Enables/disables display of markers on an area chart.
<i>DrawCompleteAreaRiser</i>	boolean	true/false	true	Enables/disables display of divider lines that serve as visual separators between each group in area charts.
<i>DualAxisLineDisplay</i>	boolean	true/false	true	Shows/hides a dual axis line.
<i>DualAxisSplitPosition</i>	integer	10...90	50	Defines the location of the dual axis line.
<i>ExactColorByHeight</i>	boolean	true/false	true	Enables/disables exact color by height.
<i>ExtendToFrameEdge</i>	boolean	true/false	true	Extends combo graph to the frame edge.
<i>FillMissingData</i>	integer	0...2	0	Specifies interpolation of missing data.
<i>FootnoteDisplay</i>	boolean	true/false	true	Shows/hides the footnote of the graph.
<i>FootnoteString</i>	String	Any characters	Null String	Defines the footnote of the graph.
<i>ForceSeriesAbsolute</i>	boolean	true/false	false	Enables/disables stacked and non-stacked charts on the same graph.

Properties	Data Type	Range	Default Value	Description
<i>FrameDisplay</i>	boolean	true/false	true	Shows/hides the frame of the graph.
<i>FunnelLabelValueLocation</i>	integer	0...2	0	Shows the location of labels in a funnel graph.
<i>GaugeColorCenterByQuality</i> on page 367	boolean	true/false	false	Controls whether the center of a gauge is the same color as the quality band to which the needle points.
<i>GraphType</i>	integer	0...106	17	Selects a graph type.
<i>Grid3DFloorDisplayX</i>	boolean	true/false	false	Shows/hides the X-axis grid lines on the 3D-cube floor.
<i>Grid3DFloorDisplayZ</i>	boolean	true/false	false	Shows/hides the Z-axis grid lines on the 3D-cube floor.
<i>Grid3DLeftWallDisplayY</i>	boolean	true/false	false	Shows/hides the Y-axis grid lines on the 3D-cube left wall.
<i>Grid3DLeftWallDisplayZ</i>	boolean	true/false	false	Shows/hides the Z-axis grid lines on the 3D-cube left wall.
<i>Grid3DRightWallDisplayX</i>	boolean	true/false	false	Shows/hides the X-axis grid lines on the 3D-cube right wall.
<i>Grid3DRightWallDisplayY</i>	boolean	true/false	false	Shows/hides the Y-axis grid lines on the 3D-cube right wall.
<i>Grid3DRiserDisplayX</i>	boolean	true/false	false	Shows/hides the X-axis grid lines on the 3D-cube risers.
<i>Grid3DRiserDisplayY</i>	boolean	true/false	false	Shows/hides the Y-axis grid lines on the 3D-cube risers.

Properties	Data Type	Range	Default Value	Description
<i>Grid3DRiserDisplayZ</i>	boolean	true/false	false	Shows/hides the Z-axis grid lines on the 3D-cube risers.
<i>InterpretAsHLOC</i>	boolean	true/false	false	Interprets a PGSDK data set as <i>High</i> , <i>Low</i> , <i>Open</i> , or <i>Close</i> .
<i>LegendAutomatic</i>	boolean	true/false	false	Enables/disables the automatic legends.
<i>LegendDisplay</i>	boolean	true/false	true	Shows/hides legends.
<i>LegendMarkerPosition</i>	integer	0...4	0	Defines the legend marker position.
<i>LegendMarkersPerRow</i>	integer	0...# series	0	Defines the number of legend markers to draw in a row.
<i>LegendOrient</i>	integer	0...2	2	Sets the legend's orientation.
<i>LegendReverse</i>	boolean	true/false	false	Enables/disables legends in reverse order.
<i>LegendScroll</i>	integer	0...2	0	Sets the legend's scrolling mode.
<i>LegendSeriesCount</i>	integer	0...# series	0	Sets the number of series to scroll within the legend.
<i>LegendSeriesStart</i>	integer	0...# series	0	Sets the first series to scroll within the legend.
<i>MarkerDisplay</i>	boolean	true/false	true	Shows/hides markers.
<i>MarkerSizeDefault</i>	integer	0...100	30	Defines the default marker size.
<i>MoveAxisTitles</i>	boolean	true/false	false	Enables/disables the movement of axis titles.
<i>NestedLabels</i>	boolean	true/false	false	Enables/disables nested labels on the O1 axis.

Properties	Data Type	Range	Default Value	Description
<i>NumMissingDataSegments</i>	integer	1...25	3	Sets the number of missing data segments to use with the FillMissingData property.
<i>NumPieArcSegmentsPerPixel</i>	double	0...1	.15	Sets the number of pie arc segments per pixel when drawing pies.
<i>O1AxisLineDisplay</i>	boolean	true/false	true	Shows/hides O1 axis line.
<i>O1AxisSide</i>	integer	0...2	0	Selects the side where the O1 axis will be imaged.
<i>O1ExcludeMaxLabel</i>	boolean	true/false	false	Excludes/includes the maximum label on the O1 axis.
<i>O1ExcludeMinLabel</i>	boolean	true/false	false	Excludes/includes the minimum label on the O1 axis.
<i>O1LabelAutoAngle</i>	integer	0...2	0	Controls the automatic rotation of O1 axis labels when they overlap.
<i>O1LabelAutoSkip</i>	integer	0...2	0	Selects no, automatic, or manual label skipping.
<i>O1LabelDisplay</i>	boolean	true/false	true	Shows/hides the O1 axis labels.
<i>O1LabelMargin</i>	integer	0...100	0	Defines margins for O1 axis labels.
<i>O1LabelRotate</i>	integer	0...2	0	Defines the rotation of labels on the O1 axis.
<i>O1LabelSkipBegin</i>	integer	0...# Groups	0	Selects the first label to skip.

Properties	Data Type	Range	Default Value	Description
<i>O1LabelSkipCount</i>	integer	0...# Groups	0	Selects the label interval to skip.
<i>O1LabelStagger</i>	boolean	true/false	false	Enables/disables the staggered labels on the O1 axis.
<i>O1LabelWrap</i>	boolean	true/false	false	Enables/disables the wrapped labels on the O1 axis.
<i>O1MajorGridDisplay</i>	boolean	true/false	true	Shows/hides major grid lines on the O1 axis.
<i>O1MajorGridStyle</i>	integer	0...4	0	Selects the style of major grid lines on the O1 axis.
<i>O1MinorGridCount</i>	integer	1...50	1	Defines the number of minor grid lines on the O1 axis.
<i>O1MinorGridDisplay</i>	boolean	true/false	false	Shows/hides minor grid lines on the O1 axis.
<i>O1MinorGridStyle</i>	integer	0...4	0	Selects the style of minor grid lines on the O1 axis.
<i>O1TitleDisplay</i>	boolean	true/false	false	Shows/hides the O1 axis title.
<i>O1TitleString</i>	String	Any characters	Null String	Defines the O1 axis title string.
<i>O2ExcludeMaxLabel</i>	boolean	true/false	false	Excludes/includes the maximum label on the O2 axis.
<i>O2ExcludeMinLabel</i>	boolean	true/false	false	Excludes/includes the minimum label on the O2 axis.
<i>O2LabelAutoSkip</i>	integer	0...2	0	Selects no, automatic, or manual label skipping.

Properties	Data Type	Range	Default Value	Description
<i>O2LabelDisplay</i>	boolean	true/false	true	Shows/hides the O2 axis labels.
<i>O2LabelRotate</i>	integer	0...3	0	Selects the rotation of labels on the O2 axis.
<i>O2LabelSkipBegin</i>	integer	0...# groups	0	Sets the starting group to skip on the O2 axis.
<i>O2LabelSkipCount</i>	integer	0...# groups	0	Labels skip intervals on the O2 axis.
<i>O2LabelWrap</i>	boolean	true/false	false	Enables/disables wrapped labels on the O2 axis.
<i>O2TitleDisplay</i>	boolean	true/false	false	Shows/hides the O2 axis title.
<i>O2TitleString</i>	String	Any characters	Null String	Defines the O2 axis title string.
<i>OtherPercentage</i>	double	0...100	.5	Defines the percentage for pie slices to be grouped into the Other series.
<i>OtherSeries</i>	boolean	true/false	true	Enables/disables an Other series in a pie graph for slices less than the OtherPercentage.
<i>OverDrawSkip</i>	boolean	true/false	true	Enables/disables overdrawing in very large data sets.
<i>ParetoDisplayThreshold</i>	integer	0...100	1	Splits frame and accumulation line in Pareto graphs.
<i>ParetoY2MajorGridStep</i>	double	1.0...500.0	10.0	Sets Pareto graph percentage scale increments.
<i>PieBarLabelDisplay</i>	boolean	true/false	true	Enables/disables label display in a pie-bar graph.

Properties	Data Type	Range	Default Value	Description
<i>PieBarLabelString</i>	string	Any characters	Null String	Defines the label string in a pie-bar graph.
<i>PieBarSeries</i>	integer	0...# series in graph	0	Selects the series that is used to graph the bar in a pie-bar graph.
<i>PieDepth</i>	integer	0...100	30	Defines the depth of a pie graph.
<i>PieFeelerTextDisplay</i>	integer	0...3	1	Defines the pie feeler text display mode.
<i>PieFeelerTextFormat</i>	integer	-1...21	2	Selects a preset format for the pie feeler data text.
<i>PieFeelerTextFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a standard Java format pattern for data text next to pie feelers.
<i>PieLabelDisplay</i>	integer	0...3	1	Defines the pie label display mode.
<i>PieRingSize</i>	integer	0...100	30	Defines the pie ring size.
<i>PieRingTotalDisplay</i>	boolean	true/false	true	Shows/hides the pie ring total.
<i>PieRingTotalFormat</i>	integer	-1...21	0	Selects a preset format for the pie ring total.
<i>PieRingTotalFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a standard Java format pattern for the total value in the center of the pie ring.
<i>PieRotate</i>	integer	0...359	0	Rotates a pie.
<i>PiesPerRow</i>	integer	1...32 Groups	2	Defines the number of pies per row.

Properties	Data Type	Range	Default Value	Description
<i>PieTilt</i>	integer	0...100	10	Tilts a pie.
<i>Place</i>	boolean	true/false	false	Enables/disables the automatic graph layout manager.
<i>PrimitiveAntialiasing</i>	boolean	true/false	false	Enables/disables smoother drawing of graph objects using Java 2D.
<i>QuadrantLineCountX</i>	integer	Any integer value	1	Sets the number of vertical quadrant lines.
<i>QuadrantLineCountY</i>	integer	Any integer value	1	Sets the number of horizontal quadrant lines.
<i>RemoveDuplicateDataText</i>	Boolean	true/false	false	Enables/disables display of duplicate text values when two or more series have the same text value.
<i>RepositionDataTextCollision</i>	integer	1-10	4	Sets the number of times colliding data values will be repositioned.
<i>RepositionStackedDataTextCollision</i>	boolean	boolean	false	Enables/disables staggering of colliding data text values in stacked bars.
<i>ReshapeEnable</i>	boolean	true/false	true	Enables/disables the reshaping of risers.
<i>ResizeBarMode</i>	boolean	true/false	true	Enables/disables the resizing of risers.
<i>ReverseGroups</i>	boolean	true/false	false	Enables/disables reverse groups.
<i>ReverseSeries</i>	boolean	true/false	false	Enables/disables reverse series.

Properties	Data Type	Range	Default Value	Description
<i>Riser3DThicknessY</i>	integer	1...100	75	Defines the thickness of 3D risers in the Y direction.
<i>RiserBarGroupSpacing</i>	integer	-1...100	0	Defines the riser bar group spacing.
<i>RiserBorderMode</i>	integer	0...2	2	Defines when riser borders are drawn.
<i>RiserWidth</i>	integer	0...100	75	Defines the riser width.
<i>ScaleFromZero</i>	boolean	true/false	true	Enables/disables the scale from zero.
<i>ScrollOffsetGroup</i>	integer	0...#Groups in Graph	0	Defines the first group in a graph.
<i>ScrollOffsetSeries</i>	integer	0...#Series in Graph	0	Defines the first series in a graph.
<i>SelectionEnable</i>	integer	0...5	3	Defines the user selection mode.
<i>SelectionEnableMove</i>	boolean	true/false	true	Enables/disables the moving of objects in a graph.
<i>SeriesAreRows</i>	boolean	true/false	true	Enables/disables the series as rows.
<i>SeriesDefaultBorderColor</i>	color	Any R,G,B color	black	Sets the color to use for riser borders when UseSeriesBorderDefaults is true.
<i>SeriesDefaultTransparentBorderColor</i>	boolean	true/false	true	Uses transparent series borders when UseSeriesBorderDefaults is true.

Properties	Data Type	Range	Default Value	Description
<i>SeriesLineWidthDefault</i>	integer	1...100	3	Sets the default line width for all the series.
<i>SeriesLooping</i>	integer	0...#Series in data the defines the graph	32	Defines the series at which series colors are repeated.
<i>ShadowXOffsetDefault</i>	integer	-20...20	2	Defines the default value for the shadow X-offset.
<i>ShadowYOffsetDefault</i>	integer	-20...20	2	Defines the default value for the shadow Y-offset.
<i>SquareMarkers</i>	boolean	true/false	true	Forms legend markers in square shapes.
<i>StackedDataValueSum</i>	boolean	true/false	true	Enables/disables the sum values in stacked graphs.
<i>Stock52WeekHighDisplay</i>	boolean	true/false	false	Shows/hides the 52-week high line in a stock graph.
<i>Stock52WeekHighValue</i>	double	Any value	40.0	Defines the 52-week high value in a stock graph.
<i>Stock52WeekLowDisplay</i>	boolean	true/false	false	Shows/hides 52-week low line in a stock graph.
<i>Stock52WeekLowValue</i>	double	Any value	10.0	Defines the 52-week low value in a stock graph.
<i>StockCloseSplitDisplay</i>	boolean	true/false	true	Shows/hides the close split display in a stock graph.
<i>StockCloseTicksDisplay</i>	boolean	true/false	true	Shows/hides close ticks in a stock graph.
<i>StockMovingAverageDisplay</i>	boolean	true/false	false	Shows/hides a moving average line in a stock graph.

Properties	Data Type	Range	Default Value	Description
<i>StockOpenTicksDisplay</i>	boolean	true/false	true	Shows/hides open ticks in a stock graph.
<i>StockTickLength</i>	integer	0...100	50	Defines the stock tick length.
<i>SubtitleDisplay</i>	boolean	true/false	true	Shows/hides the subtitle of the graph.
<i>SubtitleString</i>	String	Any characters	"Chart Subtitle"	Defines the subtitle of the graph.
<i>TextAntialiasing</i>	boolean	true/false	false	Enables/disables smoother drawing of graph text objects using Java 2D.
<i>TitleDisplay</i>	boolean	true/false	true	Shows/hides the title of the graph.
<i>TitleString</i>	String	Any characters	"Chart Title"	Defines the title of the graph.
<i>ToolTipDelay</i>	integer	Any integer value	500	Sets the object tool tip delay in milliseconds.
<i>ToolTipDisplay</i>	boolean	true/false	false	Shows/hides the tool tips.
<i>ToolTipMode</i>	boolean	true/false	false	Defines the tool tip mode.
<i>ToolTipTimeoutEnabled</i>	boolean	true/false	false	Enables/disables the tool tip time-out.
<i>TransparentMarkers</i>	boolean	true/false	false	Enables/disables the transparent markers.
<i>TransparentMarkerSize</i>	integer	0...100	0	Sets the size of a transparent marker.
<i>UseDefaultBubbleMarker</i>	boolean	true/false	true	Uses the default (all circles) markers in a bubble graph.

Properties	Data Type	Range	Default Value	Description
<i>UseOffScreen</i>	boolean	true/false	true	Enables/disables the use of an off-screen buffer to draw graphs.
<i>UseSeriesBorderDefaults</i>	boolean	true/false	false	Enables/disables the use of the SeriesDefaultBorderColor or the SeriesDefaultTransparent BorderColor.
<i>UseSeriesShapes</i>	boolean	true/false	false	Enables/disables the use of series shapes.
<i>ViewableGroups</i>	integer	0...#Groups in graph	0	Defines the number of groups to include in graphs starting at the ScrollOffsetGroup.
<i>ViewableGroupsStock</i>	integer	0...#Groups in graph	0	Defines the number of groups to include in a stock graph starting at the ScrollOffsetGroup.
<i>ViewableSeries</i>	integer	0...#Series in graph	0	Defines the number of series to include in graph starting at ScrollOffsetSeries.
<i>ViewableSeriesStock</i>	integer	0...#Series in graph	0	Defines the number of series to include in a stock graph starting at the ScrollOffsetSeries.
<i>Viewing3DAnglePreset</i>	integer	-1...15	-1	Selects a preset 3D viewing angle.
<i>WaterfallLastGroupTotal</i>	boolean	true/false	false	Turns on/off display of last riser in a Waterfall chart as a total that touches the x-axis.

Properties	Data Type	Range	Default Value	Description
<i>WaterfallStackColorMode</i>	boolean	true/false	false	Enables/disables color mode for Waterfall charts which displays positive values as green and negative values as red.
<i>X1AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending X1 axis.
<i>X1AxisLineDisplay</i>	boolean	true/false	true	Shows/hides the X1 axis line.
<i>X1AxisSide</i>	integer	0...2	0	Defines where the X1 axis is imaged.
<i>X1ExcludeMaxLabel</i>	boolean	true/false	false	Excludes/includes the maximum label on the X1 axis.
<i>X1ExcludeMinLabel</i>	boolean	true/false	false	Excludes/includes the minimum label on the X1 axis.
<i>X1LabelDisplay</i>	boolean	true/false	true	Shows/hides the X1 axis labels.
<i>X1LabelFormat</i>	integer	-1...21	0	Defines the format of labels on the X1 axis.
<i>X1LabelFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a format pattern for X1 label values.
<i>X1LabelRotate</i>	integer	0...3	0	Defines the rotation of labels on the X1 axis.
<i>X1LabelStagger</i>	boolean	true/false	false	Enables/disables staggered labels on the X1 axis
<i>X1LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the X1 axis.

Properties	Data Type	Range	Default Value	Description
<i>X1MajorGridDisplay</i>	boolean	true/false	true	Shows/hides major grid lines on the X1 axis.
<i>X1MajorGridStep</i>	double	Any positive number	15.0	Defines the number of major grid steps on the X1 axis.
<i>X1MajorGridStepAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of major grid steps on the X1 axis.
<i>X1MajorGridStyle</i>	integer	0...4	0	Selects the style of major grid lines on the X1 axis.
<i>X1MinorGridDisplay</i>	boolean	true/false	false	Shows/hides minor grid lines on the X1 axis.
<i>X1MinorGridStep</i>	double	Any positive number	15.0	Defines the number of minor grid steps on the X1 axis.
<i>X1MinorGridStepAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of minor grid steps on the X1 axis.
<i>X1MinorGridStyle</i>	integer	0...4	0	Selects the style of minor grid lines on the X1 axis.
<i>X1MustIncludeZero</i>	boolean	true/false	false	Enables/disables the <i>Must Include Zero</i> option the X1 axis.
<i>X1OffScaleDisplay</i>	boolean	true/false	true	Shows/hides the offscale values on the X1 axis.
<i>X1ScaleMax</i>	double	Any value	100.0	Defines the maximum value on the X1 axis.

Properties	Data Type	Range	Default Value	Description
<i>X1ScaleMaxAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the maximum value on the X1 axis.
<i>X1ScaleMin</i>	double	Any value	0.0	Defines the minimum value on the X1 axis.
<i>X1ScaleMinAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the minimum value on the X1 axis.
<i>X1TitleDisplay</i>	boolean	true/false	false	Shows/hides the X1 axis title.
<i>X1TitleString</i>	String	Any characters	Null String	Defines the X1 axis title String.
<i>X1ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the X1 axis.
<i>Y1AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending Y1 axis.
<i>Y1AxisLineDisplay</i>	boolean	true/false	true	Shows/hides the Y1 axis line.
<i>Y1AxisSide</i>	integer	0...2	0	Defines where the Y1 axis is imaged.
<i>Y1ExcludeMaxLabel</i>	boolean	true/false	false	Excludes/includes the maximum label on the Y1 axis.
<i>Y1ExcludeMinLabel</i>	boolean	true/false	false	Excludes/includes the minimum label on the Y1 axis.
<i>Y1LabelDisplay</i>	boolean	true/false	true	Shows/hides Y1 axis labels.

Properties	Data Type	Range	Default Value	Description
<i>Y1LabelFormat</i>	integer	-1...21	0	Selects a preset format for labels on the Y1 axis.
<i>Y1LabelFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a standard Java format pattern for Y1 label values.
<i>Y1LabelRotate</i>	integer	0...2	0	Defines the rotation of labels on the Y1 axis.
<i>Y1LabelStagger</i>	boolean	true/false	false	Enables/disables staggered labels on the Y1 axis.
<i>Y1LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the Y1 axis.
<i>Y1MajorGridDisplay</i>	boolean	true/false	true	Shows/hides major grid lines on the Y1 axis.
<i>Y1MajorGridStep</i>	double	Any positive number	15.0	Defines the number of major grid steps on the Y1 axis.
<i>Y1MajorGridStepAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of major grid steps on the Y1 axis.
<i>Y1MajorGridStyle</i>	integer	0...4	0	Selects the style of major grid lines on the Y1 axis.
<i>Y1MinorGridDisplay</i>	boolean	true/false	false	Shows/hides minor grid lines on the Y1 axis.
<i>Y1MinorGridStepAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of minor grid steps on the Y1 axis.
<i>Y1MinorGridStyle</i>	integer	0...4	0	Selects the style of minor grid lines on the Y1 axis.

Properties	Data Type	Range	Default Value	Description
<i>Y1MustIncludeZero</i>	boolean	true/false	false	Enables/disables the <i>Must Include Zero</i> option on the Y1 axis.
<i>Y1OffScaleDisplay</i>	boolean	true/false	true	Shows/hides the offscale values on the Y1 axis.
<i>Y1ScaleMax</i>	double	Any value	70.0	Defines the maximum value on the Y1 axis.
<i>Y1ScaleMaxAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the maximum value on the Y1 axis.
<i>Y1ScaleMin</i>	double	Any value	0.0	Defines the minimum value on the Y1 axis.
<i>Y1ScaleMinAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the minimum value on the Y1 axis.
<i>Y1TitleDisplay</i>	boolean	true/false	false	Shows/hides the Y1 axis title.
<i>Y1TitleString</i>	String	Any characters	Null String	Defines the Y1 axis title String.
<i>Y1ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the Y1 axis.
<i>Y2AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending Y2 axis.
<i>Y2AxisLineDisplay</i>	boolean	true/false	true	Shows/hides the Y2 axis line.
<i>Y2AxisSide</i>	integer	0...2	1	Defines where the Y2 axis is imaged.

Properties	Data Type	Range	Default Value	Description
<i>Y2ExcludeMaxLabel</i>	boolean	true/false	false	Excludes/includes the maximum label on the Y2 axis.
<i>Y2ExcludeMinLabel</i>	boolean	true/false	false	Excludes/includes the minimum label on the Y2 axis.
<i>Y2LabelDisplay</i>	boolean	true/false	true	Shows/hides the Y2 axis labels.
<i>Y2LabelFormat</i>	integer	-1...21	0	Selects a preset format for labels on the Y1 axis.
<i>Y2LabelFormatPattern</i>	string	pattern:= subpattern {;subpattern}	"#.#"	Defines a standard Java format pattern for Y2 label values.
<i>Y2LabelRotate</i>	integer	0...2	0	Defines the rotation of labels on the Y2 axis.
<i>Y2LabelStagger</i>	boolean	true/false	false	Enables/disables the staggered labels on the Y2 axis.
<i>Y2LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the Y2 axis.
<i>Y2MajorGridDisplay</i>	boolean	true/false	true	Shows/hides major grid lines on the Y2 axis.
<i>Y2MajorGridStep</i>	double	Any positive number	15.0	Defines the number of major grid steps on the Y2 axis.
<i>Y2MajorGridStepAuto</i>	boolean	true/false	false	Enables/disables the automatic calculation of major grid steps on the Y2 axis.
<i>Y2MajorGridStyle</i>	integer	0...4	0	Selects the style of major grid lines on the Y2 axis.

Properties	Data Type	Range	Default Value	Description
<i>Y2MinorGridDisplay</i>	boolean	true/false	false	Shows/hides minor grid lines on the Y2 axis.
<i>Y2MinorGridStep</i>	double	Any positive number	15.0	Defines the number of minor grid steps on the Y2 axis.
<i>Y2MinorGridStepAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of minor grid steps on the Y2 axis.
<i>Y2MinorGridStyle</i>	integer	0...4	0	Selects the style of minor grid lines on the Y2 axis.
<i>Y2MustIncludeZero</i>	boolean	true/false	false	Enables/disables the <i>Must Include Zero</i> option on the Y2 axis.
<i>Y2OffScaleDisplay</i>	boolean	true/false	true	Shows/hides offscale values on the Y2 axis.
<i>Y2ScaleMax</i>	double	Any value	80.0	Defines the maximum value on the Y2 axis.
<i>Y2ScaleMaxAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the maximum value on the Y2 axis.
<i>Y2ScaleMin</i>	double	Any value	10.0	Defines the minimum value on the Y2 axis.
<i>Y2ScaleMinAuto</i>	boolean	true/false	true	Enables/disables the automatic calculation of the minimum value on the Y2 axis.
<i>Y2TitleDisplay</i>	boolean	true/false	false	Shows/hides the Y2 axis title.

Properties	Data Type	Range	Default Value	Description
<i>Y2TitleString</i>	String	Any characters	Null String	Defines the Y2 axis title String.
<i>Y2ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the Y2 axis.
<i>Y3AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending Y3 axis.
<i>Y3LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the Y3 axis.
<i>Y3ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the Y3 axis.
<i>Y4AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending Y4 axis.
<i>Y4LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the Y4 axis.
<i>Y4ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the Y4 axis.
<i>Y5AxisDescending</i>	boolean	true/false	false	Sets the descending/ascending Y5 axis.
<i>Y5LogScale</i>	boolean	true/false	false	Sets the linear/logarithmic scale on the Y5 axis.
<i>Y5ZeroLineDisplay</i>	boolean	true/false	false	Shows/hides the zero line on the Y5 axis.
<i>ZeroValueDataTextDisplay</i>	boolean	true/false	true	Shows/hides zero data values when the data has the value zero.

Properties	Data Type	Range	Default Value	Description
<i>ZeroValueDataTextStackedTotalOnTopDisplay</i>	boolean	true/false	true	Shows/hides zero data values when a stack has a cumulative total of zero.
<i>ZoomDirection</i>	integer	0...2	0	Defines the direction to use for the doZoomIn() and doZoomOut() property.
<i>ZoomPercentage</i>	double	0...100	50.0	Defines the number to use for the doZoomIn() and doZoomOut() property.

Graph Properties Reserved For Future Use

Properties	Data Type	Range	Default Value	Descriptions
CustomCurveFitEndGroup	integer	0...# groups	0	Sets the end group for a custom curved fit line.
CustomCurveFitNumCoeff	integer	0...100	0	Sets the custom curved fit coefficient.
CustomCurveFitSeries	integer	0...# series	0	Sets the series for the custom curved fit.
CustomCurveFitStartGroup	integer	0...# groups	0	Sets the start group for a custom curved fit line.
CustomCurveFitType	integer	0...10000	1	Sets the custom curved fit type.
FunnelVertExplosionFactor	integer	0...100	50	Sets the vertical explosion of a funnel graph.
NestedLabelsUniformStyle	boolean	true/false	false	Enables/disables the uniform style of all nested labels.

Properties	Data Type	Range	Default Value	Descriptions
QOEnabled	integer	true/false	false	Enables/disables quality objects.
RRAllocateEntireResource	boolean	true/false	false	Allocates the entire resource in the resource return graph.
RRIgnoreMinimumValues	boolean	true/false	false	Ignores minimum values in the resource return graph.
RRTotalResource	double	Any value	1000	Totals resources available for the resource return graph.

Graph Properties Reserved for Internal Use

Properties	Data Type	Range	Default Value	Descriptions
DeadZonePercent	Integer	0...100	0	Sets the percentage of the dead zone.
DisplaySystemConsoleMessages	boolean	true/false	true	Shows/hides the system console messages for programmer errors.
LegendFontSizeSpacing	boolean	true/false	false	Enables/disables legend spacing to reflect font sizing.
TextDirection	integer	Unknown	0	Sets special text direction.

Graph Methods Index

This appendix describes methods used in the WebFOCUS Graph API.

In this appendix:

- ❑ [Current Graph Methods](#)
- ❑ [Reserved for Future Use](#)
- ❑ [Reserved for Internal Use](#)

Current Graph Methods

See [Graph Properties Index](#) on page 1281 for methods that get and set property values. See [IdentObj Methods Index](#) on page 1339 for methods that return an object ID in the form IdentObj.

Return Value	Method	Description
int	<code>getAltFmtFrameNumColors()</code>	Gets the number of colors/sections to be used for alternate formatting of an axis. See get/setAltFmtFrameNumColors() on page 176.
double	<code>getAltFmtFrameSeparator()</code>	Gets the location on an axis where alternate formatting is applied. See get/setAltFmtFrameSeparator() on page 176.
int	getAutoSkip()	Gets label automatic skip mode.
int	getAxisAssignment()	Gets axis assignment.
boolean	getAxisDescending()	Gets axis descending.
int	getAxisOffset()	Gets axis offset value for Multi-Y graph.
int	getAxisSide()	Gets axis side.

Return Value	Method	Description
int	<i>getAxisSize()</i>	Gets axis size value for Multi-Y graph.
String	<i>getBorder()</i>	Gets the border assigned to an object.
color	<i>getBorderColor()</i>	Gets the border color of an object.
int	<i>getBorderType()</i>	Gets the border type of an object.
Insets	<i>getBorderWidth()</i>	Gets the width of a border assigned to an object.
boolean	<i>getChartFromFTP()</i>	Gets a graph from an FTP server.
int	<i>getCornerArcWidth/Height()</i>	Gets the width/height of the corner arc of a border applied to an object.
int	<i>getCurveFitPolynomialOrder()</i>	Gets polynomial order for a curve fit line.
int	<i>getCurveFitType()</i>	Gets the curve fit type.
boolean	<i>getCustomDataText()</i>	Gets custom data text.
int	<i>getCustomDataTextFormat()</i>	Gets custom data text format.
String	<i>getCustomDataTextFormatPattern()</i>	Gets custom data text format pattern.
double	<i>getDataAsDouble()</i>	Gets data at row/col as double.
String	<i>getDataAsString()</i>	Gets data at row/col as a string.
String	<i>getDataLabel()</i>	Gets the data label at series/group.
int	<i>getDataTextAngle()</i>	Gets data text angle.
int	<i>getDataTextRadius()</i>	Gets data text radius.
double	<i>getDataValue()</i>	Gets data value assigned to the object identified by id.
boolean	<i>getDisplay()</i>	Gets the display mode of an object.
boolean	<i>getDisplayOffScale()</i>	Gets display offscale mode.

Return Value	Method	Description
int	<i>getDocumentHeightInPoints()</i>	Gets document height in points.
int	<i>getDocumentWidthInPoints()</i>	Gets document width in points.
boolean	<i>getDrawTicksPerpendicular()</i>	Gets draw ticks perpendicular setting.
int	<i>getEffectiveColorMode()</i>	Returns the actual color mode used.
int	<i>getEffectiveGroupOffset()</i>	Gets the effective group offset.
int	<i>getEffectiveSeriesOffset()</i>	Gets the effective series offset.
int	<i>getEffectiveViewableGroups()</i>	Gets the effective number of viewable groups.
int	<i>getEffectiveViewableSeries()</i>	Gets the effective number of viewable series.
Hashtable	<i>getExceptionalRisers()</i>	Determines if there are any exceptional risers. See <i>Exceptional Risers</i> on page 96.
boolean	<i>getExcludeMaxLabel()</i>	Determines if maximum label is imaged on an axis.
boolean	<i>getExcludeMinLabel()</i>	Determines if minimum label is imaged on an axis.
Color	<i>getFillColor()</i>	Gets fill color.
int	<i>getFillType()</i>	Gets fill type.
String	<i>getFontName()</i>	Gets the font name assigned to a text object.
int	<i>getFontSize()</i>	Gets the font size in destination coordinates.
boolean	<i>getFontSizeAbsolute()</i>	Gets font size absolute setting.
int	<i>getFontSizeInPoints()</i>	Gets font size in points.
int	<i>getFontSizeVC()</i>	Gets the font size in virtual coordinates.
int	<i>getFontStyle()</i>	Gets the font style of a text object.
Point	<i>getFrameDepthOffset()</i>	Gets frame depth offset.

Return Value	Method	Description
Rectangle	<i>getFrameRect()</i>	Gets the size and location of the graph frame.
int	<i>getGradientDirection()</i>	Gets the gradient direction.
int	<i>getGradientNumPins()</i>	Gets the number of pins in a gradient.
Color	<i>getGradientPinLeftColor()</i>	Gets the left color used by the gradient pin.
double	<i>getGradientPinPosition()</i>	Gets the gradient pin position.
Color	<i>getGradientPinRightColor()</i>	Gets the right color used by the gradient pin.
int	<i>getGridCount()</i>	Gets grid count.
double	<i>getGridStep()</i>	Gets number of grid steps on an axis.
boolean	<i>getGridStepAuto()</i>	Determines if grid steps are automatically calculated.
int	<i>getGridStyle()</i>	Gets style of grid lines assigned to an axis.
String	<i>getGroupLabel()</i>	Gets a group label.
boolean	<i>getIgnoreSeries()</i>	Determines if a series is ignored.
Insets	<i>getInsets()</i>	Gets the insets of an object.
int	<i>get...inset()</i>	Gets the top/bottom and right/left insets of borders of a specified object.
boolean	<i>getLabelStagger()</i>	Determines if staggered labels are used.
int	<i>getLegendMinWidth()</i>	Gets the minimum legend width.
Rectangle	<i>getLegendRect()</i>	Gets the size and location of the legend area.
String	<i>getLevelDateFormat()</i>	Gets level date format for time scale axis.
Color	<i>getLevelFillColor()</i>	Gets level fill color for time scale axis.
int	<i>getLevelHeight()</i>	Gets level height for time scale axis.
int	<i>getLevelInterval()</i>	Gets level interval for time scale axis.

Return Value	Method	Description
int	<i>getLevelTextFormat()</i>	Gets level test format for time scale axis.
int	<i>getLineBasicStrokeType()</i>	Gets a Java 2D basic stroke type of the line object.
int	<i>getLineCap()</i>	Gets a Java 2D cap style of the line object.
Vector	<i>getLineDash()</i>	Gets a Java 2D dash definition of the line object as a vector.
float[]	<i>getLineDashFloat()</i>	Gets a Java 2D dash definition of the line object as a floating point array.
float	<i>getLineDashPhase()</i>	Gets a Java 2D dash offset of the line object.
int	<i>getLineJoin()</i>	Gets a Java 2D join style of the line object.
float	<i>getLineMiterLimit()</i>	Gets a Java 2D line miter limit of the line object.
int	<i>getLineWidth()</i>	Gets the width of a line.
boolean	<i>getLogScale()</i>	Gets logarithmic/linear scaling assigned to an axis.
double	<i>getLogScaleBase()</i>	Gets logarithmic scale base.
int	<i>getMarkerShape()</i>	Gets marker shape.
int	<i>getMarkerSize()</i>	Gets marker size.
MarkerTemplate	<i>getMarkerTemplate()</i>	Gets a marker template.
int	<i>getMaxNumAnnotations()</i>	Gets the maximum number of annotations.
long	<i>getMillisFromDate()</i>	Returns the current date as a long.
Dimension	<i>getMinimumSize()</i>	Gets the minimum size of a graph.
int	<i>getMinNumGroups()</i>	Gets minimum # of groups for this graph type.
int	<i>getMinNumSeries()</i>	Gets minimum # of series for the graph type.

Return Value	Method	Description
boolean	<i>getNonZeroBaseline()</i>	Gets the designated baseline.
double	<i>getNonzeroBaselineValue()</i>	Gets the designated non-zero baseline value.
int	<i>getNumColumnsInGroup()</i>	Gets the number of columns in a group for this graph type.
int	<i>getNumGroups()</i>	Gets the number of groups in the graph.
int	<i>getNumNonIgnoredSeries()</i>	Gets the number of the series in the graph that are not ignored.
int	<i>getNumSeries()</i>	Gets the number of series in the graph.
int	<i>getNumTotalGroups()</i>	Gets the total number of groups in the graph.
int	<i>getNumTotalSeries()</i>	Gets the total number of series in the graph.
boolean	<i>getPieSliceDelete()</i>	Determines if a pie slice is deleted.
int	<i>getPieSliceDetach()</i>	Determines if a pie slice is detached.
int	<i>getPlaceAlign()</i>	Gets alignment setting for automatic graph layout manager.
Rectangle	<i>getPlaceMargin()</i>	Gets graph margin used by automatic graph layout manager.
int	<i>getPlaceMaxNumWordWrapLines()</i>	Gets maximum number of word-wrap lines for automatic graph layout.
int	<i>getPlaceMinFontSize()</i>	Gets minimum font point size for automatic graph layout.
Rectangle	<i>getRect()</i>	Gets the bounding rectangle for a user-movable object in virtual coordinates (x, y, width, height).
Color	<i>getSeriesBorderColor()</i>	Gets the border color of a series.
Color	<i>getSeriesFillColor()</i>	Gets the fill color of a series.

Return Value	Method	Description
String	<i>getSeriesLabel()</i>	Gets series label.
int	<i>getSeriesType()</i>	Gets series type.
Color	<i>getShadowColor()</i>	Gets the Shadow Color of an object.
boolean	<i>getShadowDisplay()</i>	Determines if a shadow has been applied to an object.
int	<i>getShadowFillType()</i>	Gets the fill type assigned to a Java 2D text object.
Color	<i>getShadowGradientEndColor()</i>	Gets the end color for <code>setShadowFillType(FILL_COLOR)</code> .
Color	<i>getShadowGradientStartColor()</i>	Gets the start color for <code>setShadowFillType(FILL_COLOR)</code> .
Image/String	<i>getShadowTextureURL()</i>	Gets the texture for <code>setShadowFillType(FILL_TEXTURE)</code> .
int	<i>getShadowXOffset()</i>	Gets shadow X-Offset.
int	<i>getShadowYOffset()</i>	Gets shadow Y-Offset.
int	<i>getSkipBegin()</i>	Gets label skip beginning number.
int	<i>getSkipCount()</i>	Gets label skip count.
Color	<i>getTextBackgroundColor()</i>	Gets a Java 2D text object's background color.
String	<i>getTextFormatPattern()</i>	Gets text format pattern.
int	<i>getTextFormatPreset()</i>	Gets text format preset.
int	<i>getTextJustHoriz()</i>	Gets text horizontal justification.
int	<i>getTextJustVert()</i>	Gets text vertical justification.
int	<i>getTextRotation()</i>	Gets text rotation.

Return Value	Method	Description
boolean	<i>getTextStrikethrough()</i>	Gets a Java 2D text object's strike-through mode.
String	<i>getTextString()</i>	Gets text string.
String	<i>getTexture()</i>	Gets texture.
Color	<i>getTextureBGColor()</i>	Gets texture background color.
int	<i>getTextureDisplayMode()</i>	Gets texture display mode assigned to object id.
int	<i>getTextureFlipMode()</i>	Gets texture flip mode.
boolean	<i>getTextWrap()</i>	Gets text wrap mode (on or off).
int	<i>getTickCount()</i>	Gets tick count.
int	<i>getTickLength()</i>	Gets tick length.
double	<i>getTickStep()</i>	Gets tick step.
boolean	<i>getTickStepAuto()</i>	Gets tick step auto setting.
int	<i>getTickStyle()</i>	Gets tick style.
int	<i>getTickWidth()</i>	Gets tick width.
boolean	<i>getTransparentBorderColor()</i>	Gets transparent border color.
boolean	<i>getTransparentFillColor()</i>	Gets transparent fill color.
boolean	<i>getUseTextBackgroundColor()</i>	Gets a Java 2D text object's use of background color.
int	<i>getWaterfallGroupMode()</i>	Gets Waterfall graph group mode.
double	<i>getX1MajorGridStepAutoValue()</i>	Gets automatically calculated major grid steps on the X-Axis.
double	<i>getX1ScaleMaxAutoValue()</i>	Gets automatically calculated maximum scale value on the X-Axis.

Return Value	Method	Description
double	<i>getX1ScaleMinAutoValue()</i>	Gets automatically calculated minimum scale value on the X-Axis.
double	<i>getY1MajorGridStepAutoValue()</i>	Gets automatically calculated major grid steps on the Y1-Axis.
double	<i>getY1ScaleMaxAutoValue()</i>	Gets automatically calculated maximum scale value on the Y1-Axis.
double	<i>getY1ScaleMinAutoValue()</i>	Gets automatically calculated minimum scale value on the Y1-Axis.
double	<i>getY2MajorGridStepAutoValue()</i>	Gets automatically calculated major grid steps on the Y2-Axis.
double	<i>getY2ScaleMaxAutoValue()</i>	Gets automatically calculated maximum scale value on the Y2-Axis.
double	<i>getY2ScaleMinAutoValue()</i>	Gets automatically calculated minimum scale value on the Y2-Axis.
boolean	<i>isChart3DType()</i>	Determines whether or not the current graph is one of the 3D graphs. See GraphType Methods on page 66.
boolean	<i>isChartBiPolar()</i>	Determines whether or not the current graph is bi-polar. See GraphType Methods on page 66.
boolean	<i>isChartBLAType()</i>	Determines whether or not the current graph is a Bar, Line, or Area graph. See GraphType Methods on page 66.
boolean	<i>isChartDualY()</i>	Determines whether or not the current graph is a dual-axes graph. See GraphType Methods on page 66.

Return Value	Method	Description
boolean	isChartOrientHorz()	Determines whether or not the current graph is horizontal (false=vertical graph). See GraphType Methods on page 66.
boolean	isChartPieType()	Determines whether or not the current graph is one of the pie graphs. See GraphType Methods on page 66.
boolean	isChartScatter()	Determines whether or not the current graph is a scatter graph. See GraphType Methods on page 66.
boolean	isChartStockType()	Determines whether or not the current graph is one of the stock graphs. See GraphType Methods on page 66.
boolean	isExceptionalAllowed()	Determines if exceptional risers are allowed. See Exceptional Risers on page 96.
boolean	isGroupScrollerPresent()	Determines if the group data scroller bar is present.
boolean	isO1AxisPresent()	Determines if the O1 axis is present in the graph. See Ordinal Axis Methods on page 82.
boolean	isO2AxisPresent()	Determines if the O2 axis is present in the graph. See Ordinal Axis Methods on page 82.
boolean	isValidZoomIndex()	Determines if a particular data zooming index is valid. See Data Zooming on page 191.

Return Value	Method	Description
boolean	<code>isX1AxisPresent()</code>	Determines if the X1 axis is present in the graph. See Numeric Axis Methods on page 83.
boolean	<code>isY1AxisPresent()</code>	Determines if the Y1 axis is present in the graph. See Numeric Axis Methods on page 83.
boolean	<code>isY2AxisPresent()</code>	Determines if the Y2 axis is present in the graph. See Numeric Axis Methods on page 83.
boolean	<code>isZ1AxisPresent()</code>	Determines if the Z1 axis is present in the graph. See Numeric Axis Methods on page 83.
boolean	<code>isZoomingArmed()</code>	Determines if <code>doZoomIn()</code> or <code>doZoomOut()</code> has been called. See Data Zooming on page 191.
boolean	<code>optimizeData()</code>	Asks the graph to optimize the data.
String	<code>outputParamString()</code>	Gets a list of graph parameters.
void	<code>removeReferenceLineObject()</code>	Removes a reference line object. See APIs Related to Reference Lines on page 184.
void	<code>restoreAllSlices()</code>	Restores all deleted/detached slices from a pie.
void	<code>restoreDefaults()</code>	Restores all default values.
void	<code>save()</code>	Saves a graph to an output stream.
boolean	<code>sendChartToFTP()</code>	Sends a graph to an FTP server.
void	<code>sendGIFToFile()</code>	Sends a graph to a file as a .GIF image.

Return Value	Method	Description
void	<i>sendGIFToFileExc()</i>	Sends a graph to a file as a GIF image and throws the appropriate exceptions.
void	<i>sendGIFtoFTP()</i>	Sends a graph to an FTP server as a GIF image.
void	<i>sendGIFtoStream()</i>	Sends a graph to an output stream as a GIF image.
void	<i>sendImageMapToStream()</i>	Sends an image map to a stream.
void	<i>sendJPEGToStream()</i>	Sends a graph to an output stream as a JPEG image.
void	<i>sendPNGToFile()</i>	Sends a graph to a file as a PNG image.
void	<i>sendPNGToFileExc()</i>	Sends a graph to a file as a PNG image and throw the appropriate exceptions.
void	<i>sendPNGtoStream()</i>	Sends a graph to an output stream as a PNG image.
void	<i>sendPNGtoStreamExc()</i>	Sends a graph to an output stream as a PNG image and throw the appropriate exceptions.
void	<i>setAltFmtFrameNumColors()</i>	Sets the number of colors/sections to be used for alternate formatting of an axis.
void	<i>setAltFmtFrameSeparator()</i>	Sets the location on an axis where alternate formatting is applied.
void	<i>setAutoColorLerpFactor()</i>	Sets a factor for lightening or darkening groups of risers.
void	<i>setAutoSkip()</i>	Sets label skip mode.
void	<i>setAxisAssignment()</i>	Sets axis assignment.
void	<i>setAxisDescending()</i>	Sets axis ascending/descending.
void	<i>setAxisOffset()</i>	Sets Axis Offset in Multi-Y graphs.

Return Value	Method	Description
void	<i>setAxisSide()</i>	Sets axis side.
void	<i>setAxisSize()</i>	Sets Axis Size in Multi-Y graphs.
void	<i>setBorder()</i>	Assigns a border to an object.
void	<i>setBorderColor()</i>	Sets border color.
void	<i>setBorderType()</i>	Sets the border type for the specified object.
void	<i>setBorderWidth()</i>	Sets the width of a border assigned to an object.
void	<i>setCornerArc()</i>	Sets the corner arc of an object.
void	<i>setCornerArcWidth/Height()</i>	Sets the width/height of the corner arc of a border applied to an object.
void	<i>setCubeRotationMatrix()</i>	Sets the rotation matrix of the 3D cube.
void	<i>setCurveFitPolynomialOrder()</i>	Sets polynomial order for a curve fit line.
void	<i>setCurveFitType()</i>	Sets the curve fit type.
void	<i>setCustomDataText()</i>	Sets custom data text.
void	<i>setCustomDataTextFormat()</i>	Sets custom data text format.
void	<i>setCustomDataTextFormatPattern()</i>	Sets custom data text format pattern.
void	<i>setData()</i>	Sets data in a graph.
void	<i>setDataLabel()</i>	Assigns a data label to a series/group.
void	<i>setDataRange()</i>	Sets the range of data to be imaged.
void	<i>setDataRangeToExtent()</i>	Sets the data range to the values set by <i>setData()</i> .
void	<i>setDataTextAngle()</i>	Sets the angle of data text.
void	<i>setDataTextRadius()</i>	Sets data text radius.

Return Value	Method	Description
void	setDataValue()	Assigns a data value to an object.
void	setDisplay()	Enables/disables display of an object.
void	setDisplayOffScale()	Sets display off scale.
void	setDrawTicksPerpendicular()	Sets draw ticks perpendicular to axis.
void	setEndDate()	Sets the end date in a graph. See Time Scale Axis Graphs on page 221.
void	setExceptionalRiser()	Sets exceptional risers.
void	setExcludeMaxLabel()	Excludes/Includes maximum label on an axis.
void	setExcludeMinLabel()	Excludes/Includes minimum label on an axis.
void	setFillColor()	Sets the fill color of an object.
void	setFillType()	Sets fill type.
void	setFontName()	Sets font name.
void	setFontSize()	Sets font size in points (deprecated).
void	setFontSizeAbsolute()	FALSE = Use setFontSizeVC() TRUE = Use setFontSizeInPoints()
void	setFontSizeVC()	Sets font size in virtual coordinates.
void	setFontSizeInPoints()	Sets font size in points.
void	setFontStyle()	Sets font style.
void	setGaugeAlphaChannel()	Sets the transparency of a gauge object. See Gauge Graph Properties & Methods on page 146.
void	setGradientDirection()	Sets gradient direction.
void	setGradientNumPins()	Sets number of pins in a gradient.

Return Value	Method	Description
void	<i>setGradientPinLeftColor()</i>	Sets gradient pin left color.
void	<i>setGradientPinPosition()</i>	Sets gradient pin position.
void	<i>setGradientPinRightColor()</i>	Sets gradient pin right color.
void	<i>setGridCount()</i>	Sets grid count.
void	<i>setGridStep()</i>	Sets number of grid steps.
void	<i>setGridStepAuto()</i>	Enables/disables automatic calculation of grid steps.
void	<i>setGridStyle()</i>	Sets the style of grid lines.
void	<i>setGroupLabel()</i>	Defines a group label.
void	<i>setIgnoreSeries()</i>	Includes/Excludes a series in a graph.
void	<i>setInset()</i>	Sets insets of an object.
void	<i>set...Insets</i>	Sets the top/bottom and right/left insets of borders of a specified object.
void	<i>setLabelStagger()</i>	Enables/disables staggered labels.
void	<i>setLegendPosition()</i>	Sets the legend position.
void	<i>setLegendRect()</i>	Sets the location and size of the legend area.
void	<i>setLevelBorderColor()</i>	Sets level border color for a time scale axis.
void	<i>setLevelDateFormat()</i>	Sets level date format for a time scale axis.
void	<i>setLevelFillColor()</i>	Sets level fill color for a time scale axis.
void	<i>setLevelHeight()</i>	Sets level height for a time scale axis.
void	<i>setLevelInterval()</i>	Sets level interval for a time scale axis.
void	<i>setLevelTextColor()</i>	Sets level text color for a time scale axis.
void	<i>setLevelTextFormat()</i>	Sets level text format for a time scale axis.

Return Value	Method	Description
void	setLineBasicStrokeType()	Sets a Java 2D line object's basic stroke type.
void	setLineCap()	Sets a Java 2D line object's cap style.
void	setLineDash()	Sets a Java 2D line object's dash definition.
void	setLineDashPhase()	Sets a Java 2D line object's dash offset.
void	setLineJoin()	Sets a Java 2D line object's join style.
void	setLineMiterLimit()	Sets a Java 2D line object's line miter limit.
void	setLineWidth()	Sets line width.
void	setLineWidthAllSeries()	Sets the line width for all series in a graph.
void	setLogScale()	Sets logarithmic/linear scaling.
void	setLogScaleBase()	Sets logarithmic scale base.
void	setMarkerShape()	Sets the shape of a marker.
void	setMarkerSize()	Set marker size for the first item in the selection list.
void	setMaxNumAnnotations()	Sets the maximum number of annotations.
void	setNoExceptionalRiser()	Disallows exceptional risers.
void	setNonZeroBaseline()	Sets Non-Zero Base Line.
void	setNonZeroBaselineValue()	Sets Non-Zero Base Line Value.
void	setNoSelection()	De-select all items in the selection list.
void	setO1LabelCallback()	Sets a callback for O1-axis nested labels. See Using Nested/Multi-Dimensional Labels on page 124.
void	setPieSliceDelete()	Sets/clear deleted pie slice.

Return Value	Method	Description
void	<i>setPieSliceDetach()</i>	Sets distance a slice is detached from a pie graph.
void	<i>setPlaceAlign()</i>	Sets alignment mode for automatic graph layout.
void	<i>setPlaceMargin()</i>	Sets graph margin used by automatic graph layout manager.
void	<i>setPlaceMaxNumWordWrapLines()</i>	Sets maximum number of word wrap Lines for automatic graph layout.
void	<i>setPlaceRect()</i>	Sets bounding rectangle for automatic graph layout.
void	<i>setPlaceResize()</i>	Sets resize mode for automatic graph layout.
void	<i>setPlaceSkip()</i>	Sets skip mode for automatic graph layout.
void	<i>setPlaceTruncateCount()</i>	Sets truncate count for automatic graph layout.
void	<i>setPlaceWordWrap()</i>	Sets word wrap mode for automatic graph layout.
void	<i>setQuadrantLineValueX()</i>	Sets scale value at which to place vertical quadrant line.
void	<i>setQuadrantLineValueY()</i>	Sets scale value at which to place horizontal quadrant line.
void	<i>setRect()</i>	Sets the location and size of an object.
void	<i>setReferenceLine()</i>	Sets a reference line in a graph.
void	<i>setReferenceLineValue()</i>	Sets the value at which a reference line displays in a graph.
void	<i>setScaleMax()</i>	Assigns a maximum scale value.
void	<i>setScaleMaxAuto()</i>	Enables/disables automatic calculation of the maximum scale value.

Return Value	Method	Description
void	<code>setScaleMin()</code>	Assigns a minimum scale value.
void	<code>setScaleMinAuto()</code>	Enables/disables automatic calculation of the minimum scale value.
void	<code>setScaleMustIncludeZero()</code>	Enables/disables zero on scale.
void	<code>setSeriesAreRows()</code>	Assigns a value to the SeriesAreRows property and enables/disables swapping of labels.
void	<code>setSeriesBorderColor()</code>	Assigns a border color to a series.
void	<code>setSeriesFillColor()</code>	Assigns a fill color to a series.
void	<code>setSeriesLabel()</code>	Assigns a series label.
void	<code>setSeriesLooping()</code> on page 980	Sets an index that defines the number of risers in a group for automatic lightening or darkening of riser colors.
void	<code>setSeriesType()</code>	Assigns a series type.
void	<code>setShadowColor()</code>	Assigns a shadow color to an object.
void	<code>setShadowDisplay()</code>	Enables/disables shadow display.
void	<code>setShadowFillType()</code>	Sets the fill type of a Java 2D text object.
void	<code>setShadowGradientEndColor()</code>	Sets the end color for <code>setShadowFillType(FILL_COLOR)</code> .
void	<code>setShadowGradientStartColor()</code>	Sets the start color for <code>setShadowFillType(FILL_COLOR)</code> .
void	<code>setShadowTextureURL()</code>	Sets the texture for <code>setShadowFillType(FILL_TEXTURE)</code> .
void	<code>setShadowXOffset()</code>	Sets shadow X-offset.
void	<code>setShadowYOffset()</code>	Sets shadow Y-offset.

Return Value	Method	Description
void	setSize()	Inherited by Perspective from Java Components.
void	setSkipBegin()	Assigns a label skip begin value.
void	setSkipCount()	Assigns a label skip interval.
void	setStartDate()	Sets the end date in a graph. See Time Scale Axis Graphs on page 221.
void	setTextBackgroundColor()	Sets a Java 2D text object's background color.
void	setTextFormatPattern()	Assigns a standard Java text format pattern.
void	setTextFormatPreset()	Assigns a preset text format.
void	setTextJustHoriz()	Sets text horizontal justification.
void	setTextJustVert()	Sets text vertical justification.
void	setTextRotation()	Sets text rotation.
void	setTextStrikethrough()	Sets a Java 2D text object's strike-through mode.
void	setTextString()	Sets text string.
void	setTextureBGColor()	Sets texture background color.
void	setTextureDisplayMode()	Sets texture display mode.
void	setTextureFlipMode()	Sets texture flip mode.
void	setTextureResource()	Specifies a texture file as a resource instead of a file-system URL.
void	setTextureURL()	Loads a texture from a URL.
void	setTextWrap()	Sets text wrap mode.
void	setTickCount()	Sets tick count.

Return Value	Method	Description
void	setTickLength()	Sets tick length.
void	setTickStep()	Sets tick step.
void	setTickStepAuto()	Enables/disables automatic calculation of tick steps.
void	setTickStyle()	Sets tick style.
void	setTickWidth()	Sets tick width.
void	setToolTipBackdropColor()	Sets Tooltip backdrop color.
void	setToolTipBackdropSize()	Sets Tooltip backdrop size.
void	setToolTipBackgroundColor()	Sets Tooltip background color.
void	setToolTipColor()	Sets Tooltip color.
void	setToolTipFontName()	Sets Tooltip font name as defined in java.awt.Font.
void	setToolTipFontSize()	Sets Tooltip font size as defined in java.awt.Font.
void	setToolTipFontStyle()	Set Tooltip font style as defined in java.awt.Font.
void	setToolTipOff()	Disables tool tip event generation.
void	setToolTipOn()	Enables tool tip event generation.
void	setTransparentBorderColor()	Sets transparent border color.
void	setTransparentFillColor()	Sets transparent fill color.
void	setURL()	Sets URL. See URL Methods on page 170.
void	setURLTarget()	Sets the frame target for a URL. See URL Methods on page 170.

Return Value	Method	Description
void	setUserToolTip()	Sets user-mode custom tool tip string.
void	setUserToolTipDefault()	Selects default user-mode tool tips.
void	setUseTextBackgroundColor()	Enables/disables a Java 2D text object's background color.
void	setWaterfallGroupMode()	Sets Waterfall Group Mode in a waterfall graph.
void	setZoomIndex()	Sets the current zoom index in the zoom list.

Reserved for Future Use

Return Value	Method	Description
AlternateFormatter	getAlternateFormatting()	Gets alternate formatting for an object.
String	getAltString()	Gets ALT string assigned to an object.
double	getCustomCurveFitCoeff()	Gets custom curve fit coefficient.
boolean	getForecast()	Gets forecast flag assigned to an object.
Color	getForecastColor()	Gets forecast color assigned to an object.
String	getForecastImage()	Gets forecast image assigned to an object.
int	getMajorTickSide()	Gets major tick side.
int	getMinorTickSide()	Gets minor tick side.
boolean	getQAEnabled()	Quality Axis: Enables/disables
boolean	getQAHighArrow()	Quality Axis: Draws higher band as an arrow or a rectangle
double	getQAHigherSplitValue()	Quality Axis: High Split Value
int	getQALabelAlignment()	Quality Axis: Label Alignment

Return Value	Method	Description
boolean	getQALabelMode()	Quality Axis: Label Mode
int	getQALabelPosition()	Quality Axis: Label Position
boolean	getQALowArrow()	Quality Axis: Draws lower band as an arrow or a rectangle
double	getQALowerSplitValue()	Quality Axis: Lower Split Value
int	getQAOffsetFromAxis()	Quality Axis: Offset of the Quality Axis from the graph frame (virtual coordinates)
boolean	getQASmoothBlend()	Quality Axis: Enables/Disables Smooth Blend
int	getQAThickness()	Quality Axis: Thickness of the band (virtual coordinates)
boolean	getQAUserDefinedHigherSplitValue()	Quality Axis: User-Defined Higher Split Value
boolean	getQAUserDefinedLowerSplitValue()	Quality Axis: User-Defined Lower Split Value
double	getQORangeValue()	Quality Object: Gets Quality Object Range Value
String	getSummaryString()	Gets summary string assigned to an object.
void	setAlternateFormatting()	Sets alternate formatting on a object.
void	setAltString()	Assigns ALT string to an object.
void	setForecast()	Assigns forecast flag to an object.
void	setForecastColor()	Assigns forecast color to an object.
void	setForecastImage()	Assigns forecast image to an object.
void	setCustomCurveFitCoeff()	Sets Custom Curve Fit Coefficient.
void	setMajorTickSide()	Sets the axis side to draw major ticks.
void	setMinorTickSide()	Sets the axis side to draw minor ticks.

Return Value	Method	Description
void	setQAEnabled()	Quality Axis: Enables/Disables
void	setQAHighArrow()	Quality Axis: Draws higher band as an arrow or a rectangle
void	setQAHigherSplitValue()	Quality Axis: Sets Higher Split Value
void	setQALabelAlignment()	Quality Axis: Alignment
void	setQALabelMode()	Quality Axis: Mode
void	setQALabelPosition()	Quality Axis: Label Position
void	setQALowArrow()	Quality Axis: Draws lower band as an arrow or a rectangle
void	setQALowerSplitValue()	Quality Axis: Lower Split Value
void	setQAOffsetFromAxis()	Quality Axis: Offset of the Quality Axis from the graph frame (virtual coordinates). Negative values place the QA inside the frame
void	setQASmoothBlend()	Quality Axis: Smooth Blend
void	setQAThickness()	Quality Axis: Thickness of the band (virtual coordinates)
void	setQAUserDefinedHigherSplitValue()	Quality Axis: User-Defined Higher Split Value
void	setQAUserDefinedLowerSplitValue()	Quality Axis: User-Defined Lower Split Value
void	setQORangeValue()	Sets Quality Object Range Value.
void	getSummaryString()	Assigns summary string to an object.

Reserved for Internal Use

Return Value	Method	Description
void	_paint()	Paints graphics.

Return Value	Method	Description
void	actionPerformed()	Implements a timer for displaying management of ChartTips.
boolean	applyAntialiasing()	Applies antialiasing.
boolean	areNoSeriesIgnored()	Are any series ignored in a given series range.
void	beginDrawProcess()	Begins drawing graph.
void	calc()	Used by Place functionality.
void	chartCreationFailed()	Displays a graph state error message.
Object	clone ()	Returns a graph instance.
void	copy()	Copies a graph instance.
IdentObj	createTestAnnotation()	Creates a test annotation object.
void	deleteUserDefinedQAPositions()	Deletes user-defined Quality Axis Positions.
Exception	doPainting()	Do painting.
void	errorEvent()	Responds to error event.
void	focusCube()	Sets focus for selected 3D graph object.
Access	getAccess()	Returns the Access object.
Object	getAttrValue()	Gets attribute value.
double	getAutoHigherSplitValue()	Returns automatic higher axis split value.
double	getAutoLowerSplitValue()	Returns automatic lower axis split value.
BadDogCallback	getBadDogCallBack()	Gets BadDogCallBack function.
synchronized int	getChartState()	Returns the graph state.
String	getCodebase()	Gets code base.

Return Value	Method	Description
Rectangle	getDataExtents()	Uses by setDataRangeToExtent() to determine the range of data in the graph.
DateDataObj	getDateData()	Gets date data.
Detectiv	getDetectiv()	Gets tdg draw detectiv.
TDGErrorManager	getErrorManager()	Gets Error Manager.
TDGEventManager	getEventManager()	Gets Event Manager.
FontCache	getFontCache()	Gets Font Cache.
Rectangle	getFrameFrontRect()	Gets Frame Front Rectangle.
JChart_Base	getGraphObject()	Gets Graph Object.
boolean	getHasGraphics2D()	Does Perspective instance have Graphics2D object.
static CategoryIF	getInstance()	Unknown.
JGraphType	getJGraphType()	Returns a J-Graph Type.
final int	getKeyByteAt()	Callback delegate to describe collating sequence to RadixSort.
int	getLabelMargin()	Gets the margin assigned to a label.
Frame	getOffscreenFrame()	Gets offscreen frame.
TDGPreScaleIF	getPreScaleCallback()	Gets PreScale Callback.
boolean	getPrintingInProgress()	Determines if printing is in progress.
double	getReferenceLineValue()	Gets the location of a reference line for a particular axis and specific identifier.
void	getReferenceObjects()	Gets Reference Objects.
double	getResourceMax()	Gets Maximum allowable allocation for this series.

Return Value	Method	Description
double	getResourceMin()	Gets Minimum allowable allocation for this series.
SmartText	getSmartText()	Gets Smart Text.
SVGDrawList	getSVGDrawList()	Gets SVG Draw List.
boolean	getTextTruncate()	Basic text truncate for PlaceDefaultElements.
TDGDataCallbackIF	getTrueDataCallback()	Gets True Data Callback.
String	getToolTipCustomString()	Custom string for ToolTip Customization.
boolean	getToolTipDisplayTimeOut()	Gets the actual state of the timeout event.
String	getUserTooltipStrings()	Creates Image Maps with Tooltip strings.
VC	getVC()	Gets Virtual Coordinates.
ZoomStateList	getZoomList()	Gets zoom list.
void	initGauges()	Initializes Gauge graph.
void	IBIInit()	Initialization function for IBI.
void	initQualityObjects()	Initializes Quality Objects.
boolean	isApplet()	Method was created by a SmartGuide
synchronized boolean	isChartReady()	Determines if a graph is in the ready state.
void	keyPressed()	Key Pressed Event
void	keyReleased()	Key Released Event
void	keyTyped()	Key Typed Event
void	mouseClicked()	Mouse Click Event
void	mouseDragged()	Mouse Drag Event

Return Value	Method	Description
void	mouseEntered()	Mouse Enter Event
void	mouseExited()	Mouse Exit Event
void	mouseMoved()	Mouse Moved Event
void	mousePressed()	Mouse Button Pressed Event
void	mouseReleased()	Mouse Button Released Event
void	needRecalc()	Sets graph needs recalculation.
void	paint()	First entry point from the framework.
void	place()	Places default elements code.
void	print()	Prints graph.
void	paintComponent()	Paint component
void	resetNonMatrixTransformParameters()	Resets the non-matrix transformation parameters: pan, scalecube, focus, and so on from a previous save.
void	resetWorkDataStorage()	When user adds or changes data to the "original data", this method is called to make sure we don't try to use a working data sets that is obsolete.
void	rotateCubeX()	Rotate Cube X.
void	rotateCubeY()	Rotate Cube Y.
void	rotateCubeZ()	Rotate Cube Z.
void	saveNonMatrixTransformParameters()	Saves the non-matrix transformation parameters: pan, scale cube, focus, and so on so they can be restored by the user.
void	scaleCube()	Scale Cube.

Return Value	Method	Description
void	setBadDogCallback()	Method in BadDogCallback Interface.
void	setCodebase()	Sets code base.
void	setDateData()	Sets date data.
void	setEditChart()	Sets Edit Chart On or Off.
void	setEditZoom()	Sets Edit Zoom On Or Off.
void	setHasGraphics2D()	Sets has 2D Graphics.
void	setLabelMargin()	Enables/disables label margins.
void	setResourceMax()	Maximum allowable allocation for this series.
void	setResourceMin()	Minimum allowable allocation for this series.
void	setRiserDetectionLimitMaxObjects(n)	n represents the number of risers that can have a tooltip and drilldowns. The Default is 500 and the range is from 0 to 64,000.
void	setTextTruncate()	Basic text truncate property.
void	setToolTipCustomString()	Custom string for ToolTip Customization.
void	setToolTipDisplayTimeOut()	The actual state of the timeout event.
void	setTrueDataCallback()	Sets TrueDataCallback function.
void	startListeners()	Starts Event Listeners.
void	stopListeners()	Stops Event Listeners.
void	sync3DPresetParameters()	Synchronizes 3D Parameters.
void	translateCube()	Translates Cube.
void	update()	Updates.

Return Value	Method	Description
void	updateAccess()	Updates access.
void	updateParamsFromMatrix()	Updates Parameters from Matrix.
void	useWorkDataStorage()	Makes a working copy of the original data and sets the data storage to point to it.
void	writeTextRepresentation()	Writes text representation.

IdentObj Methods Index

This appendix describes IdentObj methods used for graph types.

In this appendix:

- ❑ [Current IdentObj Methods](#)
- ❑ [IdentObj Methods Reserved for Future Use](#)

Current IdentObj Methods

The methods listed in this appendix return the object ID of an object in a graph.

Return Value	Method	Returns the Object ID of the...
IdentObj	getAnnotation()	Annotation
IdentObj	getAnnotationBox()	Annotation Box
IdentObj	getAnnotationLabel()	Annotation Label
IdentObj	getAnnotationLine()	Annotation Line
IdentObj	getAreaRiser()	Area riser
IdentObj	getAreaRiserBottom()	Bottom of an area riser
IdentObj	getAreaRiserTop()	Top of an area riser
IdentObj	getBarRiser()	Bar risers
IdentObj	getBarRiserBottom()	Bottom of bar risers
IdentObj	getBarRiserSide()	Side of bar risers
IdentObj	getChartBackground()	Graph background
IdentObj	getColorByHeight()	Gradient

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getCubeFloor()</i>	3D-cube floor
IdentObj	<i>getCubeFloorGridX()</i>	3D-cube floor X-axis grid lines
IdentObj	<i>getCubeFloorGridZ()</i>	3D-cube floor Z-axis grid lines
IdentObj	<i>getCubeLeftWall()</i>	3D-cube left wall
IdentObj	<i>getCubeLeftWallGridY()</i>	3D-cube left wall Y-axis grid lines
IdentObj	<i>getCubeLeftWallGridZ()</i>	3D-cube left wall Z-axis grid lines
IdentObj	<i>getCubeRightWall()</i>	3D-cube right wall
IdentObj	<i>getCubeRightWallGridX()</i>	3D-cube right wall X-axis grid lines
IdentObj	<i>getCubeRightWallGridZ()</i>	3D-cube right wall Y-axis grid lines
IdentObj	<i>getCubeRiserFace()</i>	3D-cube riser face
IdentObj	<i>getCubeRiserGridX()</i>	3D-cube riser X-axis grid lines
IdentObj	<i>getCubeRiserGridY()</i>	3D-cube riser Y-axis grid lines
IdentObj	<i>getCubeRiserGridZ()</i>	3D-cube riser Z-axis grid lines
IdentObj	<i>getCurveFitCorrelationText()</i>	Curve-fit correlation text
IdentObj	<i>getCurveFitCorrelationTextbox()</i>	Text box behind curve-fit correlation text
IdentObj	<i>getCurveFitEquationText()</i>	Curved fit equation text
IdentObj	<i>getCurveFitEquationTextbox()</i>	Text box behind curved fit equation text
IdentObj	<i>getCurveFitLineRiser()</i>	Curved fit line
IdentObj	<i>getDataText()</i>	Data text
IdentObj	<i>getExceptionalRiser()</i>	Exceptional riser
IdentObj	<i>getFootnote()</i>	Graph footnote

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getFootnoteBox()</i>	Text box behind a footnote
IdentObj	<i>getFrame()</i>	Graph frame
IdentObj	<i>getFrameBottom()</i>	Bottom of the graph frame
IdentObj	<i>getFrameSeparator()</i>	Frame separator object
IdentObj	<i>getFrameSide()</i>	Side of the graph frame
IdentObj	<i>getFunnelCrust()</i>	Crust in a funnel graph
IdentObj	<i>getFunnelFeeler()</i>	Feeler line in a funnel graph
IdentObj	<i>getFunnelLabel()</i>	Labels in a funnel graph
IdentObj	<i>getFunnelValueLabel()</i>	Value label in a funnel graph
IdentObj	<i>getGaugeArea()</i>	Area in a gauge graph
IdentObj	<i>getGaugeBackground()</i>	Background area behind a gauge graph
IdentObj	<i>getGaugeBand1...5()</i>	Bands around the gauge
IdentObj	<i>getGaugeBand1...5Label()</i>	Labels next to gauge bands
IdentObj	<i>getGaugeLabel()</i>	Value labels on the gauge
IdentObj	<i>getGaugeMajorTick()</i>	Major tick marks on the gauge
IdentObj	<i>getGaugeMinorTick()</i>	Minor tick marks on the gauge
IdentObj	<i>getGaugeNeedle()</i>	Gauge needle
IdentObj	<i>getGaugeScaleArea()</i>	Gauge scale area
IdentObj	<i>getGaugeTitle()</i>	Gauge title
IdentObj	<i>getGlobal()</i>	Global IdentObj
IdentObj	<i>getGroup()</i>	A specified group

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getGroupScroller()</i>	Group scroller bar
IdentObj	<i>getGroupScrollerEditBox()</i>	Group scroller edit box
IdentObj	<i>getGroupScrollerEditLeft()</i>	Group scroller edit left bar
IdentObj	<i>getGroupScrollerEditRight()</i>	Group scroller edit right bar
IdentObj	<i>getGroupScrollerEditThumb()</i>	Group scroller edit thumb bar
IdentObj	<i>getGroupScrollerThumb()</i>	Group scroller thumb
IdentObj	<i>getHistogramRiser()</i>	Risers in a histogram graph
IdentObj	<i>getLegendArea()</i>	Legend area
IdentObj	<i>getLegendMarker()</i>	Legend marker
IdentObj	<i>getLegendScroller()</i>	Legend scroller
IdentObj	<i>getLegendScrollerThumb()</i>	Legend scroller thumb
IdentObj	<i>getLegendText()</i>	Legend text
IdentObj	<i>getLegendTextBox()</i>	Text box behind the legend
IdentObj	<i>getLineMarker()</i>	Markers in a line graph
IdentObj	<i>getLineRiser()</i>	Risers in a line graph
IdentObj	<i>getLineRiserSide()</i>	Side of the risers in a line graph
IdentObj	<i>getLineRiserTop()</i>	Top of the risers in a line graph
IdentObj	<i>getNestedO1Label0...8()</i>	Nested labels for levels zero through eight on the O1 axis
IdentObj	<i>getNestedO1Label0...8Box()</i>	Text box behind nested labels
IdentObj	<i>getNestedO1LabelLine()</i>	Nested labels line
IdentObj	<i>getNonZeroBaseLineZeroLine()</i>	Non-zero baseline zero line

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getO1Axis()</i>	O1 axis
IdentObj	<i>getO1AxisLine()</i>	O1 axis line
IdentObj	<i>getO1Label()</i>	O1 axis labels
IdentObj	<i>getO1MajorGrid()</i>	O1 axis major grid lines
IdentObj	<i>getO1MajorTick()</i>	O1 axis major tick marks
IdentObj	<i>getO1MinorGrid()</i>	O1 axis minor grid lines
IdentObj	<i>getO1MinorTick()</i>	O1 axis minor tick marks
IdentObj	<i>getO1Title()</i>	O1 axis title
IdentObj	<i>getO1TitleBox()</i>	Text box behind the O1 title
IdentObj	<i>getO2Axis()</i>	O2 axis
IdentObj	<i>getO2Label()</i>	O2 axis labels
IdentObj	<i>getO2MajorGrid()</i>	O2 axis major grid lines
IdentObj	<i>getO2MinorGrid()</i>	O2 axis minor grid lines
IdentObj	<i>getO2Title()</i>	O2 axis title
IdentObj	<i>getO2TitleBox()</i>	Text box behind the O2 axis title
IdentObj	<i>getParetoLineMarker()</i>	Line marker in a Pareto graph
IdentObj	<i>getParetoLineRiser()</i>	Line riser in a Pareto graph
IdentObj	<i>getParetoLineRiserSide()</i>	Side of the line riser in a Pareto graph
IdentObj	<i>getParetoLineRiserSide1()</i>	Side of a line riser in a Pareto graph
IdentObj	<i>getParetoLineRiserSide2()</i>	Side of a line riser in a Pareto graph
IdentObj	<i>getParetoLineRiserTop()</i>	Top of a line riser in a Pareto graph

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getParetoLineRiserTop1()</i>	Top of a line riser in a Pareto graph
IdentObj	<i>getParetoLineRiserTop2()</i>	Top of a line riser in a Pareto graph
IdentObj	<i>getPieBarConnectLineBottom()</i>	Bottom line that connects the bar to the pie in a pie-bar graph
IdentObj	<i>getPieBarConnectLines()</i>	Lines that connect the bar to the pie in a pie-bar graph
IdentObj	<i>getPieBarLabel()</i>	Bar label in a pie-bar graph
IdentObj	<i>getPieBarRiserTop()</i>	Top of the bar in a pie-bar graph
IdentObj	<i>getPieBarSlice()</i>	Pie slice in a pie-bar graph
IdentObj	<i>getPieBarSliceCrust()</i>	Pie slice crust in a pie-bar graph
IdentObj	<i>getPieFeelerLine()</i>	Feeler line in a pie graph
IdentObj	<i>getPieFrame()</i>	Frame in a pie graph
IdentObj	<i>getPieLabel()</i>	Labels in a pie graph
IdentObj	<i>getPieRingLabel()</i>	Label in the center of a pie ring graph
IdentObj	<i>getPieSliceLabel()</i>	Slice labels in a pie graph
IdentObj	<i>getPlaceHorzO1LabelsTitle()</i>	For horizontal graphs, rectangle of O1 labels and O1 titles used by an automatic graph layout
IdentObj	<i>getPlaceVertO1LabelsTitle()</i>	For vertical graphs, rectangle of O1 labels and O1 titles used by an automatic graph layout
IdentObj	<i>getQuadrantLine()</i>	Get the object ID of quadrant lines in a bubble graph
IdentObj	<i>getReferenceLine()</i>	Gets the reference line object for a particular axis and specific identifier

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getReferenceLineLegendText()</i>	Gets the reference line legend text
IdentObj	<i>getReferenceLineO1()</i>	A reference line for axis O1.
IdentObj	<i>getReferenceLineO2()</i>	A reference line for axis O2.
IdentObj	<i>getReferenceLineX1()</i>	A reference line for axis X1.
IdentObj	<i>getReferenceLineY1()</i>	A reference line for axis Y1.
IdentObj	<i>getReferenceLineY2()</i>	A reference line for axis Y2.
IdentObj	<i>getReferenceLineTextO1()</i>	A reference line label for axis O1.
IdentObj	<i>getReferenceLineTextO2()</i>	A reference line label for axis O2.
IdentObj	<i>getReferenceLineTextX1()</i>	A reference line label for axis X1.
IdentObj	<i>getReferenceLineTextY1()</i>	A reference line label for axis Y1.
IdentObj	<i>getReferenceLineTextY2()</i>	A reference line label for axis Y2.
IdentObj	<i>getScatterQuadrantLine()</i>	Quadrant lines in a scatter graph
IdentObj	<i>getSeries()</i>	A specified series
IdentObj	<i>getSeriesGroup()</i>	A series/group
IdentObj	<i>getSeriesScroller()</i>	Series scroller bar
IdentObj	<i>getSeriesScrollerEditBox()</i>	Series scroller edit box
IdentObj	<i>getSeriesScrollerEditLeft()</i>	Series scroller edit left bar
IdentObj	<i>getSeriesScrollerEditRight()</i>	Series scroller edit right bar
IdentObj	<i>getSeriesScrollerEditThumb()</i>	Series scroller edit thumb bar
IdentObj	<i>getSeriesScrollerThumb()</i>	Series scroller thumb bar
IdentObj	<i>getSlice()</i>	Slice in a pie graph

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getSliceCrust()</i>	Slice crust in a pie graph
IdentObj	<i>getSliceRing()</i>	Slice ring in a pie graph
IdentObj	<i>getSpectralLegendMarker()</i>	Legend markers in a spectral map
IdentObj	<i>getSpectralMarker()</i>	Markers in a spectral map
IdentObj	<i>getStackConnectLine()</i>	Connected line in a stacked chart
IdentObj	<i>getStock52WeekHighLine()</i>	52-week high line in a stock graph
IdentObj	<i>getStock52WeekLowLine()</i>	52-week low line in a stock graph
IdentObj	<i>getStockCloseTick()</i>	Close ticks in a stock graph
IdentObj	<i>getStockFallingRiser()</i>	Falling risers in a stock, candle graph
IdentObj	<i>getStockHighLine()</i>	High line in a stock graph
IdentObj	<i>getStockLowerRiser()</i>	Lower edge of risers in a stock graph
IdentObj	<i>getStockLowLine()</i>	Low line in a stock graph
IdentObj	<i>getStockMALine()</i>	Moving average line in a stock graph
IdentObj	<i>getStockOpenTick()</i>	Open ticks in a stock graph
IdentObj	<i>getStockRiser()</i>	Risers in a stock graph
IdentObj	<i>getStockRisingRiser()</i>	Rising risers in a stock, candle graph
IdentObj	<i>getStockVolumeRiser()</i>	Volume risers in a stock graph
IdentObj	<i>getSubTitle()</i>	Graph subtitle
IdentObj	<i>getSubTitleBox()</i>	Text box behind a subtitle
IdentObj	<i>getThermGaugeArea()</i>	Tube in a thermometer gauge graph
IdentObj	<i>getThermGaugeBackground()</i>	Background area outside the tube in a thermometer gauge graph

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getThermGaugeNeedle()</i>	Liquid in a thermometer gauge graph
IdentObj	<i>getTimeScaleLevel()</i>	Time Scale Level
IdentObj	<i>getTitle()</i>	Graph title
IdentObj	<i>getTitleBox()</i>	Text box behind a graph title
IdentObj	<i>getWaterFallLine()</i>	Line that connects risers in a waterfall graph
IdentObj	<i>getX1Axis()</i>	X1 axis
IdentObj	<i>getX1AxisLine()</i>	X1 axis base line
IdentObj	<i>getX1Label()</i>	X1 axis labels
IdentObj	<i>getX1MajorGrid()</i>	X1 axis major grid lines
IdentObj	<i>getX1MajorTick()</i>	X1 axis major tick marks
IdentObj	<i>getX1MinorGrid()</i>	X1 axis minor grid lines
IdentObj	<i>getX1MinorTick()</i>	X1 axis minor tick marks
IdentObj	<i>getX1Title()</i>	X1 axis title
IdentObj	<i>getX1TitleBox()</i>	Text box behind the X1 axis title
IdentObj	<i>getX1ZeroLine()</i>	X1 axis zero line
IdentObj	<i>getY1Axis()</i>	Y1 axis
IdentObj	<i>getY1AxisLine()</i>	Y1 axis line
IdentObj	<i>getY1Label()</i>	Y1 axis labels
IdentObj	<i>getY1MajorGrid()</i>	Y1 axis major grid lines
IdentObj	<i>getY1MajorTick()</i>	Y1 axis major tick marks
IdentObj	<i>getY1MinorGrid()</i>	Y1 axis minor grid lines

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getY1MinorTick()</i>	Y1 axis minor tick marks
IdentObj	<i>getY1Title()</i>	Y1 axis title
IdentObj	<i>getY1TitleBox()</i>	Text box behind the Y1 axis title
IdentObj	<i>getY1ZeroLine()</i>	Y1 axis zero line
IdentObj	<i>getY2Axis()</i>	Y2 axis
IdentObj	<i>getY2AxisLine()</i>	Y2 axis base line
IdentObj	<i>getY2Label()</i>	Y2 axis labels
IdentObj	<i>getY2MajorGrid()</i>	Y2 axis major grid lines
IdentObj	<i>getY2MajorTick()</i>	Y2 axis major tick marks
IdentObj	<i>getY2MinorGrid()</i>	Y2 axis minor grid lines
IdentObj	<i>getY2MinorTick()</i>	Y2 axis minor tick marks
IdentObj	<i>getY2Title()</i>	Y2 axis title
IdentObj	<i>getY2TitleBox()</i>	Text box behind the Y2 axis title
IdentObj	<i>getY2ZeroLine()</i>	Y2 axis zero line
IdentObj	<i>getY3Axis()</i>	Y3 axis
IdentObj	<i>getY3AxisLine()</i>	Y3 axis line
IdentObj	<i>getY3Label()</i>	Y3 axis labels
IdentObj	<i>getY3MajorGrid()</i>	Y3 axis major grid lines
IdentObj	<i>getY3MajorTick()</i>	Y3 axis major tick marks
IdentObj	<i>getY3MinorGrid()</i>	Y3 axis minor grid lines
IdentObj	<i>getY3MinorTick()</i>	Y3 axis minor tick marks

Return Value	Method	Returns the Object ID of the...
IdentObj	<i>getY3Title()</i>	Y3 axis title
IdentObj	<i>getY3TitleBox()</i>	Text box behind the Y3 axis title
IdentObj	<i>getY3ZeroLine()</i>	Y3 axis zero line
IdentObj	<i>getY4Axis()</i>	Y4 axis
IdentObj	<i>getY4AxisLine()</i>	Y4 axis line
IdentObj	<i>getY4Label()</i>	Y4 axis labels
IdentObj	<i>getY4MajorGrid()</i>	Y4 axis major grid lines
IdentObj	<i>getY4MajorTick()</i>	Y4 axis major tick marks
IdentObj	<i>getY4MinorGrid()</i>	Y4 axis minor grid lines
IdentObj	<i>getY4MinorGrid()</i>	Y4 axis minor tick marks
IdentObj	<i>getY4Title()</i>	Y4 axis title
IdentObj	<i>getY4TitleBox()</i>	Text box behind the Y4 axis title
IdentObj	<i>getY4ZeroLine()</i>	Y4 axis zero line
IdentObj	<i>getY5Axis()</i>	Y5 axis
IdentObj	<i>getY5AxisLine()</i>	Y5 axis line
IdentObj	<i>getY5Label()</i>	Y5 axis labels
IdentObj	<i>getY5MajorGrid()</i>	Y5 axis major grid lines
IdentObj	<i>getY5MajorTick()</i>	Y5 axis major tick marks
IdentObj	<i>getY5MinorGrid()</i>	Y5 axis minor grid lines
IdentObj	<i>getY5MinorTick()</i>	Y5 axis minor tick marks
IdentObj	<i>getY5Title()</i>	Y5 axis title

Return Value	Method	Returns the Object ID of the...
IdentObj	getY5TitleBox()	Text box behind the Y5 axis title
IdentObj	getY5ZeroLine()	Y5 axis zero line

IndentObj Methods Reserved for Future Use

Return Value	Method	Returns the Object ID of the...
IdentObj	getBreakevenLossArea()	Loss area in a Breakeven graph
IdentObj	getBreakevenLossLabel()	Loss label in a Breakeven graph
IdentObj	getBreakevenProfitArea()	Profit area in a Breakeven graph
IdentObj	getBreakevenProfitLabel()	Profit label in a Breakeven graph
IdentObj	getBreakevenSummaryLabel()	Summary label in a Breakeven graph
IdentObj	getQualityObject()	Quality object
IdentObj	getQualityObjectLabel()	Quality object label
IdentObj	getResourceRiser()	Resource riser in a Resource Return graph
IdentObj	getTotalResourceMarker()	Resource marker in a Resource Return graph
IdentObj	getX1QualityAxisHigher()	QualityAxis high band fill color on the X1 axis
IdentObj	getX1QualityAxisLower()	QualityAxis low band fill color on the X1 axis
IdentObj	getX1QualityAxisMedium()	QualityAxis medium band fill color on the X1 axis
IdentObj	getX1QualityLabelHigher()	QualityAxis label on the X1 axis

Return Value	Method	Returns the Object ID of the...
IdentObj	getX1QualityLabelLower()	QualityAxis label on the X1 axis
IdentObj	getX1QualityLabelMedium()	QualityAxis label on the X1 axis
IdentObj	getY1QualityAxisHigher()	QualityAxis high band fill color on the Y1 axis
IdentObj	getY1QualityAxisLower()	QualityAxis low band fill color on the Y1 axis
IdentObj	getY1QualityAxisMedium()	QualityAxis medium band fill color on the Y1 axis
IdentObj	getY1QualityLabelHigher()	QualityAxis label on the Y1 axis
IdentObj	getY1QualityLabelLower()	QualityAxis label on the Y1 axis
IdentObj	getY1QualityLabelMedium()	QualityAxis label on the Y1 axis
IdentObj	getY2QualityAxisHigher()	QualityAxis high band fill color on the Y2 axis
IdentObj	getY2QualityAxisLower()	QualityAxis low band fill color on the Y2 axis
IdentObj	getY2QualityAxisMedium()	QualityAxis medium band fill color on the Y2 axis
IdentObj	getY2QualityLabelHigher()	QualityAxis label on the Y2 axis
IdentObj	getY2QualityLabelLower()	QualityAxis label on the Y2 axis
IdentObj	getY2QualityLabelMedium()	QualityAxis label on the Y2 axis
IdentObj	getZ1TitleBox()	Z-axis title box

Static Variables

This appendix lists static variables for graph types.

In this appendix:

- ❑ [Current Static Variables](#)

Current Static Variables

The following topic table lists static variables for graph types, their descriptions, and the methods in which they are used.

Variable Name	Description	Where Used
CLOSE_VALUE_MACRO	"[CLOSE]"	DataTextTemplateHiLo
FILL_COLOR=1	Java 2D Text Fill Type	setShadowFillType()
FILL_GRADIENT_LEFTTORIGHT=2	Java 2D Text Fill Type	setShadowFillType()
FILL_TEXTURE=3	Java 2D Text Fill Type	setShadowFillType()
FIT_EXPONENTIAL	Exponential curve fit line	getCurveFitType() , setCurveFitType()
FIT_GEOMETRIC	Geometric curve fit line	getCurveFitType() , setCurveFitType()
FIT_HYPERBOLIC	Hyperbolic curve fit line	getCurveFitType() , setCurveFitType()
FIT_LINEAR	Linear curve fit line	getCurveFitType() , setCurveFitType()
FIT_LOGARITHMIC	Logarithmic curve fit line	getCurveFitType() , setCurveFitType()

Variable Name	Description	Where Used
FIT_LOGQUADRATIC	Log quadratic curve fit line	getCurveFitType() , setCurveFitType()
FIT_MODEXPONENTIAL	Modified exponential curve fit line	getCurveFitType() , setCurveFitType()
FIT_MODHYPERBOLIC	Modified hyperbolic curve fit line	getCurveFitType() , setCurveFitType()
FIT_NONE	No curve fit line	getCurveFitType() , setCurveFitType()
FIT_POLYNOMIAL	Polynomial curve fit line	getCurveFitType() , setCurveFitType()
FIT_QUADRATIC	Quadratic curve fit line	getCurveFitType() , setCurveFitType()
FIT_RATIONAL	Rational curve fit line	getCurveFitType() , setCurveFitType()
FORMAT_LONG=2	Long text format	setLevelTextFormat()
FORMAT_MEDIUM=1	Medium text format	setLevelTextFormat()
FORMAT_RAW=3	Use standard Java format	setLevelTextFormat()
HI_VALUE_MACRO	"[HIGH]"	DataTextTemplateHiLo
LOCATION_VALUES_ONLEFT _LABELS_ON_RIGHT=2;	Locate values on the left and labels on the right in a funnel graph	FunnelLabelValueLocation
LOCATION_LABELS_ONLEFT _VALUES_ON_RIGHT=1;	Locate labels on the left and values on the right in a funnel graph	FunnelLabelValueLocation

Variable Name	Description	Where Used
LOCATION_LABELS_ONTOP_OF_VALUES_BOTH_ON_RIGHT=0	Locate labels on top of values drawn on the right of a funnel graph	FunnelLabelValueLocation
LOW_VALUE_MACRO	"[LOW]"	DataTextTemplateHiLo
OPEN_VALUE_MACRO	"[OPEN]"	DataTextTemplateHiLo
ROTATE_270=2	Rotate text 270 degrees	O1LabelRotate , O2LabelRotate , X1LabelRotate , Y1LabelRotate , Y2LabelRotate , setTextRotation()
ROTATE_45=3	Rotate text 45degrees	O1LabelRotate , O2LabelRotate , X1LabelRotate , Y1LabelRotate , Y2LabelRotate , setTextRotation()
ROTATE_90=1	Rotate text 90 degrees	O1LabelRotate , O2LabelRotate , X1LabelRotate , Y1LabelRotate , Y2LabelRotate , setTextRotation()

Variable Name	Description	Where Used
ROTATE_NONE=0	Do not rotate text	O1LabelRotate , O2LabelRotate , X1LabelRotate , Y1LabelRotate , Y2LabelRotate , setTextRotation()
SCALING_ABSOLUTE	Scale type	getScaleMax() and getScaleMin()
SCALING_HIST	Scale type	getScaleMax() and getScaleMin()
SCALING_PERCENT	Scale type	getScaleMax() and getScaleMin()
SCALING_STACKED	Scale type	getScaleMax() and getScaleMin()
SIZE_VALUE_MACRO	"[SIZE]"	DataTextTemplateBubble
TIME_INTERVAL_DAYS=4	Days Time Interval	setLevelInterval()
TIME_INTERVAL_HOURS=3	Hours Time Interval	setLevelInterval()
TIME_INTERVAL_MINUTES=2	Minutes Time Interval	setLevelInterval()
TIME_INTERVAL_MONTHS=6	Months Time Interval	setLevelInterval()
TIME_INTERVAL_QUARTERS=7	Quarters Time Interval	setLevelInterval()
TIME_INTERVAL_SECONDS=1	Seconds Time Interval	setLevelInterval()
TIME_INTERVAL_UNDEFINED=0	Undefined Time Interval	setLevelInterval()
TIME_INTERVAL_WEEKS=5	Weeks Time Interval	setLevelInterval()

Variable Name	Description	Where Used
TIME_INTERVAL_YEARS=8	Years Time Interval	setLevelInterval()
UNDERLINE_LOW_DASHED=5	Java 2D text underline type	
UNDERLINE_LOW_DOTTED=3	Java 2D text underline type	
UNDERLINE_LOW_GRAY=4	Java 2D text underline type	
UNDERLINE_LOW_ONE_PIXEL=1	Java 2D text underline type	
UNDERLINE_LOW_TWO_PIXEL=2	Java 2D text underline type	
WATERFALL_HORZ=101	Horizontal waterfall graph	GraphType
WATERFALL_VERT=100	Vertical waterfall graph	GraphType
WATERFALLGROUPMODE_EXTRA=3	Extra waterfall group mode	getWaterfallGroupMode() and setWaterfallGroupMode()
WATERFALLGROUPMODE_NORMAL=0	Normal waterfall group mode	getWaterfallGroupMode() and setWaterfallGroupMode()
WATERFALLGROUPMODE_SUBTOTAL=1	Subtotal waterfall group mode	getWaterfallGroupMode() and setWaterfallGroupMode()
WATERFALLGROUPMODE_TOTAL=2	Total waterfall group Mode	getWaterfallGroupMode() and setWaterfallGroupMode()
X_VALUE_MACRO	"[X]"	DataTextTemplateScatter

Variable Name	Description	Where Used
X_Y_VALUE_MACRO	"[X], [Y]"	<i>DataTextTemplateScatter</i>
Y_VALUE_MACRO	"[Y]"	<i>DataTextTemplateScatter</i>

Index

- 2.5D effect 78, 92, 308, 334
- 3D Charts 67, 140, 164, 196, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226
 - Example Data 196
 - Panning, Rotating, and Scaling 164
 - Preset Viewing Angles 140
- 3D Cube
 - Characteristics 139
 - Preset Viewing Angles 1271
- 3D Cube Methods
 - setCubeRotationMatrix 865
- 3D Cube Objects
 - getCubeFloor 1044
 - getCubeFloorGridX 1045
 - getCubeFloorGridZ 1046
 - getCubeLeftWall 1047
 - getCubeRightWall 1049
- 3D Cube Properties
 - Autoshading 250
 - CubeFocusFactor 267
 - CubeIsometricProjection 269
 - CubeLightSourceX 271
 - CubeLightSourceY 272
 - CubeLightSourceZ 274
 - CubePanX 275
 - CubePanY 277
 - CubeRiserInterpolation 280
 - CubeSizeX 280, 281, 283
 - CubeSquareRisers 284
 - CubeTranslationX 286
 - CubeTranslationY 288
 - CubeTranslationZ 290
 - CubeViewerX 292
 - CubeViewerY 293
 - CubeViewerZ 294
 - CubeWallThickX 296
 - CubeWallThickY 298
 - CubeWallThickZ 300
 - CubeZoomFactor 302
 - Display3DFloor 336
 - Display3DLeftWall 338
 - Display3DRightWall 340
 - Viewing3DAnglePreset 571

A

- AdjustAutoScaleForRefLines 239
- Adobe SVG Viewer 46
- AlignMultiYBars 241
- alternate formatting 243
- AltFmtFrameSeparator 243
- AltFmtMatchAxisLabelColor 246
- API syntax rules for graphs 61
- Applet Rectangle 53
- Area Charts 1233, 1234, 1235, 1236, 1237, 1238
- Autofitting 138
- AutoHideDataTextOnSmallRisers 248
- Axes
 - Attributes 79
 - Numeric 82, 83
 - Ordinal 82
- Axis Methods
 - getAxisDescending() 717
 - getAxisSide() 720
 - getX1Axis 1147
 - getX1AxisLine 1148
 - getY1Axis 1156
 - getY1AxisLine 1157
 - getY2Axis 1167
 - getY2AxisLine 1168
 - setAxisAssignment() 850
 - setAxisDescending() 853
 - setAxisSide 856

B

- Background
 - getChartBackground 1043
- Bar Charts 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233

- bar risers
 - aligning 241
- Bar/Line/Area Charts
 - Example Data 197
- beveled borders 178
- Bi-Polar Charts 1228, 1229, 1232, 1234, 1235, 1237, 1240, 1241, 1244, 1255, 1257, 1258
- bitmapped graphs 45
- borders 178, 181, 725
 - Corner Arc 181, 725
- Borders
 - Corner Arc 864
- Bubble Charts 73, 143, 144, 204, 329, 499, 500, 1262, 1263, 1264
 - DataTextTemplateBubble 329
 - Example Data 204
 - Properties and Methods 143, 144
 - QuadrantLineCountX 499
 - QuadrantLineCountY 500

C

- Candle Stock Charts 1252, 1253, 1254, 1260
- Cartesian Coordinate System 47
- Chart Editing 165
- Chart Frame
 - FrameDisplay 362
 - getFrame 1059
 - getFrameBottom 1059
 - getFrameDepthOffset 753
 - getFrameRect 754
 - getFrameSeparator 1059
 - getFrameSide 1060
- Chart Sizing 67
- Color Methods
 - getBorderColor 723
 - getEffectiveColorMode 739
 - getFillColor 746
 - getSeriesBorderColor 807
 - getSeriesFillColor 808
 - getShadowColor 811

- Color Methods (*continued*)
 - getTransparentBorderColor 824
 - getTransparentFillColor 825
 - setBorderColor 860
 - setFillColor 890
 - setFillType 892
 - setSeriesBorderColor 976
 - setSeriesFillColor 977
 - setTransparentBorderColor 1022
 - setTransparentFillColor 1022
- Color Objects
 - getColorByHeight 1044
- Color of Y axis labels
 - Match to alternate frame region 246
- Color Properties
 - ColorMode 260
 - ExactColorByHeight 352
 - SeriesDefaultBorderColor 524
 - SeriesDefaultTransparentBorderColor 526
 - SeriesLineWidthDefault 527
 - SeriesLooping 528
 - UseSeriesBorderDefaults 562

- Colors
 - Properties and Methods 155

- colors, defining 158

- Curve Fit Lines 90

- Curve Fits
 - CurveFitEquationDisplay 303
 - CurveFitHighOrderFirst 305
 - CurveFitNumSegments 306
 - getCurveFitCorrelationText 1053
 - getCurveFitEquationText 1054
 - getCurveFitLineRiser 1055
 - getCurveFitPolynomialOrder() 726
 - getCurveFitType() 726

- customizing graphs 61

D

- Data
 - How Missing Data is Handled 193
 - Limiting the Range of Data that is Imaged 188
 - Requirements for Different Chart Types 193

Data Methods

- getDataAsDouble 731
- getDataAsString 731
- getDataCoord 732
- getDataLabel 733
- getDataValue 735
- setData 875
- setDataRange 877
- setDataRangeToExtent 878
- setDataValue 882

Data Properties

- DataItemsAlongSeries 306
- InterpretAsHLOC 396

Data Scrolling 188, 189, 310, 311, 519, 520, 566, 567, 568, 569, 740, 742, 743, 1071, 1072, 1073, 1074, 1124, 1126, 1127

- DataScrollerPresenceGroups 310
- DataScrollerPresenceSeries 311
- getEffectiveGroupOffset 740
- getEffectiveViewableGroup 742
- getEffectiveViewableSeries 743
- getGroupScroller 1071
- getGroupScrollerEditBox 1072
- getGroupScrollerEditLeft 1073
- getGroupScrollerEditRight 1073
- getGroupScrollerEditThumb 1074
- getGroupScrollerThumb 1074
- getSeriesScroller 1124
- getSeriesScrollerEditBox 1126
- getSeriesScrollerEditLeft 1126
- getSeriesScrollerEditRight 1126
- getSeriesScrollerThumb 1127
- ScrollOffsetGroup 519
- ScrollOffsetSeries 520
- ViewableGroups 566
- ViewableGroupsStock 567
- ViewableSeries 568
- ViewableSeriesStock 569

Data Sorting

- DataSortType 312

Data Text 125, 128

- Customizing 128

Data Text Methods

- get/setCustomDataText() 129
- getDataTextAngle 733
- getDataTextRadius 734

Data Text Methods (*continued*)

- setDataTextAngle 879
- setDataTextRadius 881

Data Text Objects

- getDataText 1056

Data Text Properties

- DataTextAngleDefault 314
- DataTextDisplay 316
- DataTextFormat 318
- DataTextFormatPattern 320
- DataTextPosition 322
- DataTextRadiusDefault 324
- DataTextTemplateBubble 329
- DataTextTemplateHiLo 330
- DataTextTemplateScatter 332
- StackedDataValueSum 533

Data Zooming 191, 1031

- setZoomIndex 1031

Dimensions

- getMinimumSize 779

Drill-downs 170**Dual-Axes Properties**

- DualAxisLineDisplay 350
- DualAxisSplitPosition 351

Dual-Axis Charts 1227, 1228, 1231, 1239, 1240, 1243, 1249, 1250, 1251, 1252, 1255, 1256, 1258

E

ENDSTYLE command 60

Exceptional Risers

- getExceptionalRiser 1056
- setExceptionalRisers 887
- setNoExceptionalRiser 944

F**Files**

- save 840

Floating marker 262

Font Size

- FontSizeAbsolute 418
- getFontSize 749

Font Size (*continued*)

- getFontSizeVC 751
- setFontSize 894, 897
- setFontSizeVC 898

- font sizes in graphs 51

Fonts

- getFontName 748
- getFontStyle 752
- setFontName 893
- setFontStyle 899

Footnote

- FootnoteDisplay 356
- FootnoteString 357
- getFootnote 1057

- formatting, alternate 243

- Frame Rectangle 53, 55

FTP

- getChartFromFTP 724
- sendChartToFTP 840
- sendGIFtoFTP 842

G

- Gauge charts 147, 367

- GaugeColorCenterByQuality 367

GIF

- sendGIFtoFile 841
- sendGIFtoFTP 842
- sendGIFtoStream 843

- GIF graphs 45, 47

- Gradients 164, 755, 756, 892, 902, 903, 904, 906, 908

- getGradientDirection 755
 - getGradientNumPins 755
 - getGradientPinLeftColor 756
 - getGradientPinPosition 756
 - getGradientPinRightColor 756
 - setFillType 892
 - setGradientDirection 902
 - setGradientNumPins 903
 - setGradientPinLeftColor 904
 - setGradientPinPosition 906
 - setGradientPinRightColor 908

- graph API calls 60

- graph API syntax rules 61

Graph Properties

- ParetoDisplayThreshold 466

- GraphType 64, 366, 371, 373, 374, 376, 377, 379, 380, 1213

Grid Line Methods

- getGridCount() 757
 - getGridStep() 758
 - getGridStepAuto() 759
 - getGridStyle() 760
 - getX1MajorGridStepAutoValue 826
 - getY1MajorGridStepAutoValue 830
 - getY2MajorGridStepAutoValue 835
 - setGridCount() 909, 1009
 - setGridStep() 910
 - setGridStepAuto() 911
 - setGridStyle() 913

Grid Line Objects

- getCubeLeftWallGridY 1047
 - getCubeLeftWallGridZ 1048
 - getCubeRightWallGridX 1049
 - getCubeRightWallGridY 1050
 - getCubeRiserGridX 1051
 - getCubeRiserGridY 1052
 - getCubeRiserGridZ 1053
 - getO1MajorGrid 1087
 - getO1MinorGrid 1089, 1096
 - getX1MajorGrid 1150
 - getX1MinorGrid 1152
 - getY1MajorGrid 1160
 - getY1MinorGrid 1162
 - getY2MajorGrid 1171
 - getY2MinorGrid 1173

Grid Line Properties

- Grid3DFloorDisplayX 381
 - Grid3DFloorDisplayZ 382
 - Grid3DLeftWallDisplayY 384
 - Grid3DLeftWallDisplayZ 385
 - Grid3DRightWallDisplayX 387
 - Grid3DRightWallDisplayY 388
 - Grid3DRiserDisplayX 390
 - Grid3DRiserDisplayY 391
 - Grid3DRiserDisplayZ 393

Grid Line Properties (*continued*)

- O1MajorGridDisplay 443
- O1MajorGridStyle 444, 446, 452
- O1MinorGridCount 446, 450
- O1MinorGridStyle 449
- X1MajorGridDisplay 591
- X1MajorGridStep 592
- X1MajorGridStepAuto 594
- X1MajorGridStyle 596
- X1MinorGridDisplay 598
- X1MinorGridStep 599
- X1MinorGridStepAuto 601
- X1MinorGridStyle 603
- Y1MajorGridDisplay 630
- Y1MajorGridStep 632
- Y1MajorGridStepAuto 633
- Y1MajorGridStyle 634
- Y1MinorGridDisplay 636
- Y1MinorGridStyle 639
- Y2MajorGridDisplay 667
- Y2MajorGridStep 668
- Y2MajorGridStepAuto 670
- Y2MajorGridStyle 671
- Y2MinorGridDisplay 673
- Y2MinorGridStep 675
- Y2MinorGridStepAuto 676
- Y2MinorGridStyle 677

Grid Lines 87, 88, 89

- 3D Graph 89
- Numeric Axis 88
- Ordinal Axis 87

Groups

- ColorMode 156, 260
- getGroup 1070
- getGroupLabel 761
- getNumGroups() 784
- getNumTotalGroups 786
- ReverseGroups 509
- RiserBarGroupSpacing 513
- ScrollOffsetGroup 519
- setGroupLabel 914
- ViewableGroups 566
- ViewableGroupsStock 567

H

- HiiteBarResize 394

- Histogram Charts 72, 1075, 1261
 - getHistogramRiser 1075

I

- implementing graph API calls 60

J

- Java OutputStream 840

- JPEG graphs 45, 47

L

Label Methods

- getAutoskip 715
- getDataLabel() 733
- getExcludeMaxLabel() 744
- getExcludeMinLabel() 745
- getGroupLabel() 761
- getLabelStagger() 764
- getSeriesLabel() 809
- getSkipBegin 814
- getSkipCount 815
- setAutoSkip 849
- setDataLabel() 876
- setExcludeMaxLabel 888
- setExcludeMinLabel() 889
- setGroupLabel 914
- setLabelStagger() 918
- setSeriesLabel() 979
- setSkipBegin 991
- setSkipCount 992

Label Objects

- getO1Label 1086
- getO2Label 1095
- getX1Label 1149
- getY1Label 1159
- getY2Label 1170

Label Properties

- NestedLabels 415
- O1ExcludeMaxLabel 428
- O1ExcludeMinLabel 429
- O1LabelAutoSkip 432
- O1LabelDisplay 433
- O1LabelMargin 434
- O1LabelRotate 435
- O1LabelSkipBegin 436
- O1LabelSkipCount 438
- O1LabelStagger 439
- O1LabelWrap 441
- O2ExcludeMaxLabel 454
- O2ExcludeMinLabel 456
- O2LabelAutoSkip 457
- O2LabelRotate 459
- O2LabelSkipBegin 461
- O2LabelSkipCount 463
- O2LabelWrap 463
- X1ExcludeMaxLabel 579
- X1ExcludeMinLabel 581
- X1LabelDisplay 582
- X1LabelFormat 584
- X1LabelFormatPattern 585
- X1LabelRotate 586
- X1LabelStagger 588
- Y1ExcludeMaxLabel 621
- Y1ExcludeMinLabel 622
- Y1LabelDisplay 624
- Y1LabelFormat 625
- Y1LabelFormatPattern 626
- Y1LabelRotate 627
- Y1LabelStagger 628
- Y2ExcludeMaxLabel 657
- Y2ExcludeMinLabel 658
- Y2LabelDisplay 660
- Y2LabelFormat 661
- Y2LabelFormatPattern 662
- Y2LabelRotate 663
- Y2LabelStagger 664

Labels 117, 124, 481

- Nested 124
- PieLabelDisplay 481

Labels and Titles 117, 118, 119

- Methods 119
- Numeric Axis 119

Labels and Titles (*continued*)

- Ordinal Axis 118

Legend Methods

- getLegendMinWidth 765
- getLegendRect() 765
- setLegendRect() 921

Legend Objects

- getLegendArea 1075
- getLegendMarker 1076
- getLegendText 1078
- getLegendTextBox 1079

Legend Properties

- LegendAutomatic 397
- LegendDisplay 398
- LegendMarkerPosition 400
- LegendReverse 404
- NullLegendMarkerShapesAsSquares 416

Legend Rectangle 53, 58

Legends

- Methods 102
- Properties 101

Line Charts 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245

Lines

- DataLineThickness 308
- getLineWidth 771

M

Markers

- ConnectLineMarkers 262
- ConnectScatterMarkers 265
- getLineMarker 1079
- getMarkerShape 774
- getMarkerSize 776
- getMarkerTemplate 777
- MarkerDisplay 411
- MarkerSizeDefault 412
- setMarkerShape 940
- setMarkerSize 942
- SquareMarkers 532

N

Number Formats 130, 133, 136, 478, 479, 486, 487, 584, 585, 625, 626, 661, 662

- Macros 136
- Patterns 133
- PieFeelerTextFormat 478
- PieFeelerTextFormatPattern 479
- PieRingTotalFormat 486
- PieRingTotalFormatPattern 487
- Preset 130
- X1LabelFormat 584
- X1LabelFormatPattern 585
- Y1LabelFormat 625
- Y1LabelFormatPattern 626
- Y2LabelFormat 661
- Y2LabelFormatPattern 662

Numeric Axis

- Grid Lines 88
- Labels and Titles 119
- Methods 83
- Properties 82

O

Object IDs 1339

ON GRAPH SET STYLE * command 60

Ordinal Axis

- Grid Lines 87
- Labels and Titles 118
- Methods 82
- Properties 82

Ordinal Axis Properties

- O1AxisLineDisplay 423
- O1AxisSide 425
- O1ExcludeMaxLabel 428
- O1ExcludeMinLabel 429
- O1LabelAutoSkip 432
- O1LabelDisplay 433, 434
- O1LabelRotate 435
- O1LabelSkipBegin 436
- O1LabelSkipCount 438
- O1LabelStagger 439
- O1LabelWrap 441

Ordinal Axis Properties (*continued*)

- O1MajorGridDisplay 443, 445, 451
- O1MajorGridStyle 444, 446, 452
- O1MinorGridCount 446, 450
- O1MinorGridDisplay 447
- O1MinorGridStyle 449
- O1TitleDisplay 452
- O1TitleString 454
- O2LabelAutoSkip 457
- O2LabelSkipBegin 461
- O2LabelSkipCount 463

P

Panning, Rotating, and Scaling a 3D Cube 165

Pareto series bars 263

Perspective distortion 267

PGSDK

- Support for Open-Hi-Lo-Close data sets 396

Pie Chart Methods

- getPieSliceDelete 789
- getPieSliceDetach 790
- getPieSliceFeelerPercent 1108
- getPieSliceFeelerValue 1109
- restoreAllSlices 838
- setPieSliceDelete 948
- setPieSliceDetach 949

Pie Chart Objects

- getPieBarConnectLineBottom 1103
- getPieBarConnectLines 1103
- getPieBarSlice 1104
- getPieBarSliceCrust 1104
- getPieFeelerLine 1105
- getPieFrame 1105
- getPieLabel 1106
- getPieRingLabel 1107
- getPieSliceLabel 1110
- getSlice 1127
- getSliceCrust 1127
- getSliceRing 1128

Pie Chart Properties

- OtherPercentage 420
- OtherSeries 421
- PieBarLabelString 473

Pie Chart Properties (*continued*)

- PieBarSeries 471, 473
- PieDepth 474
- PieFeelerTextDisplay 477
- PieFeelerTextFormat 478
- PieFeelerTextFormatPattern 479
- PieLabelDisplay 481
- PieRingSize 483
- PieRingTotalDisplay 484
- PieRingTotalFormat 486
- PieRingTotalFormatPattern 487
- PieRotate 488
- PieSorting 491
- PiesPerRow 492
- PieTilt 493
- Pie Charts 70, 141, 198, 1245, 1246, 1247, 1248
 - Example Data 198
 - Methods and Properties 141
- plotting a Frame Rectangle 55
- plotting a Legend Rectangle 58
- plotting a three-dimensional graph 57
- PNG (Portable Network Graphics) 47
- Polar Charts 201, 1250, 1251
 - Example Data 201
- Portable Network Graphics (PNG) 47
- Preset Viewing Angles for 3D charts 140

Q**Quadrant Lines**

- getQuadrantLine 1115
- getQuadrantLineValueX 795
- getQuadrantLineValueY 796
- getScatterQuadrantLine 1121
- QuadrantLineCountX 499
- QuadrantLineCountY 500
- setQuadrantLineValueX 961
- setQuadrantLineValueY 962

R

- Radar Charts 202, 1251, 1252
 - Example Data 202
- reference line scales, autoadjusting 239
- reference line value 965
- reference lines 183
- Risers
 - Autoshading 250
 - CubeSquareRisers 284
 - DisplayBarAsPictograph 342
 - getAreaRiser 1040
 - getBarRiser 1042
 - getExceptionalRiser 1056
 - getHistogramRiser 1075
 - getLineRiser 1080
 - Riser3DThicknessY 512
 - RiserBarGroupSpacing 513
 - RiserBorderMode 514
 - RiserWidth 516
 - setNoExceptionalRiser 944
- Risers and Markers 92, 93
 - Methods 93
 - Properties 92

S**Scalable Vector Graphics (SVG)** 45**Scaling Methods**

- getDisplayOffscale 737
- getLogScale 772
- getLogScaleBase 773
- getScaleMax 803
- getScaleMaxAuto 804
- getScaleMin 805
- getScaleMinAuto 806
- getScaleMustIncludeZero 807
- getX1ScaleMaxAutoValue 827
- getX1ScaleMinAutoValue 827
- getY1ScaleMaxAutoValue 831
- getY1ScaleMinAutoValue 832
- getY2ScaleMaxAutoValue 836
- getY2ScaleMinAutoValue 837

Scaling Methods (*continued*)

- setDisplayOffScale 884
- setLogScale 938
- setLogScaleBase 940
- setScaleMax 968
- setScaleMaxAuto 969
- setScaleMin 970
- setScaleMinAuto 971
- setScaleMustIncludeZero 972

Scaling Properties

- ScaleFromZero 517
- X1LogScale 589
- X1MustIncludeZero 605
- X1OffScaleDisplay 606
- X1ScaleMax 608
- X1ScaleMaxAuto 610
- X1ScaleMinAuto 613
- X1ZeroLineDisplay 616
- Y1LogScale 629
- Y1MustIncludeZero 641
- Y1OffScaleDisplay 642
- Y1ScaleMax 643
- Y1ScaleMaxAuto 645
- Y1ScaleMin 646
- Y1ScaleMinAuto 648
- Y1ZeroLineDisplay 651, 690
- Y2LogScale 666
- Y2MustIncludeZero 679
- Y2OffScaleDisplay 680
- Y2ScaleMax 682
- Y2ScaleMaxAuto 683
- Y2ScaleMin 685
- Y2ScaleMinAuto 686
- Y3LogScale 693
- Y3ZeroLineDisplay 695
- Y4LogScale 698
- Y4ZeroLineDisplay 700
- Y5LogScale 704
- Y5ZeroLineDisplay 706

Scatter Charts 71, 199, 265, 332, 1248, 1249, 1250

- ConnectScatterMarkers 265
- DataTemplateScatter 332
- Example Data 199

Selection Lists

- SelectionEanbleMove 523

Selection Lists (*continued*)

- SelectionEnable 522

Series

- ColorMode 260
- DataItemsAlongSeries 306
- getIgnoreSeries 762
- getNumNonIgnoredSeries 782
- getNumSeries 785
- getNumTotalSeries 787
- getSeries 1122
- getSeriesBorderColor 807
- getSeriesFillColor 808
- getSeriesGroup 1123
- getSeriesLabel() 809
- getSeriesType() 810
- OtherSeries 421
- ReverseSeries 510
- ScrollOffsetSeries 520
- SeriesDefaultBorderColor 524
- SeriesDefaultTransparentColor 526
- SeriesLineWidthDefault 527
- SeriesLooping 528
- setIgnoreSeries 915
- setSeriesAreRows 974
- setSeriesBorderColor() 976
- setSeriesFillColor() 977
- setSeriesLabel() 979
- setSeriesLooping() 980
- setSeriesType() 981
- UseSeriesBorderDefaults 562
- UseSeriesShapes 562
- ViewableSeries 568
- ViewableSeriesStock 569

Series Axis Properties

- O2ExcludeMaxLabel 454
- O2ExcludeMinLabel 456
- O2LabelDisplay 458
- O2LabelRotate 459
- O2LabelWrap 463
- O2TitleDisplay 464
- O2TitleString 466

SET commands

- ON GRAPH SET STYLE * 60

setAutoColorLerpFactor() 848

setReferenceLineValue() 965

Shadows

- getShadowDisplay 812
- getShadowXOffset 814
- getShadowYOffset 814
- Properties and Methods 157
- setShadowColor 984
- setShadowDisplay 985
- setShadowXOffset 989
- setShadowYOffset 990
- ShadowXOffsetDefault 530
- ShadowYOffsetDefault 530

Spectral Maps 73, 203, 1128, 1129, 1262

- Example Data 203
- getSpectralLegendMarker 1128
- getSpectralMarker 1129

Stacked charts

- getStackConnectLine 1129

Stock Chart Objects

- getStock52WeekHighLine 1130
- getStock52WeekLowLine 1131
- getStockCloseTick 1132
- getStockFallingRiser 1133
- getStockHighLine 1134
- getStockLowerRiser 1134
- getStockLowLine 1134
- getStockMALine 1134
- getStockOpenTick 1135
- getStockRiser 1136
- getStockRisingRiser 1136
- getStockVolumeRiser 1137

Stock Chart Properties

- DataTextTemplateHiLo 330
- Stock52WeekHighValue 535, 536
- Stock52WeekLowDisplay 538
- Stock52WeekLowValue 539
- StockCloseSplitDisplay 543
- StockCloseTicksDisplay 544
- StockMovingAverageDisplay 545
- StockOpenTicksDisplay 546
- StockTickLength 547
- ViewableGroups 567
- ViewableSeriesStock 569

Stock Charts 71, 142, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260

- Methods and Properties 142

Subtitle

- getSubtitle 1138
- SubtitleDisplay 549
- SubtitleString 550

SVG (Scalable Vector Graphics) 45

SVG Viewer 46

T

Text

- Autofitting 87
- Formatting 136
- Reverse Alignment 255

Text Methods

- getTextFormatPattern 815
- getTextFormatPreset 816
- getTextJustHoriz 817
- getTextJustVert 818
- getTextRotation 818
- getTextString 819
- getTextWrap 822
- setTextFormatPattern 994
- setTextFormatPreset 995
- setTextJustHoriz 996
- setTextJustVert 998
- setTextRotation 1000
- setTextString 1002
- setTextWrap 1008

Textures 164, 819, 821, 892, 992, 1004, 1007

- getTexture 819
- getTextureDisplayMode 821
- setFillType 892
- setTextureDisplayMode 1004
- setTextureURL 1007

three-dimensional graph effect 57

Times

- UseTimeScaleAxis 564

Titles 117, 147, 452, 454, 464, 466, 553, 555, 614, 649, 650, 688, 689, 1091, 1092, 1097, 1098, 1144, 1153, 1164, 1175

- Chart-Wide 117

- for Gauge charts 147

- getO1Title 1091, 1092, 1098

- getO2Title 1097

Titles (continued)

- getTitle 1144
- getX1Title 1153
- getY1Title 1164
- getY2Title 1175
- O1TitleDisplay 452
- O1TitleString 454
- O2TitleDisplay 464
- O2TitleString 466
- TitleDisplay 553
- TitleString 555
- X1TitleDisplay 614
- Y1TitleDisplay 649
- Y1TitleString 650
- Y2TitleDisplay 688
- Y2TitleString 689

Titles and Labels 117**Tool Tips**

- setToolTipOff 1021
- setToolTipOn 1022
- setUserToolTip 1023
- setUserToolTipDefault 1025
- ToolTipDelay 556
- ToolTipDisplay 556
- ToolTipMode 556

U

- URLs 170, 1007
 - setTextureURL 1007

- UseTimeScaleAxis 564

V

- vector graphs 45

- Viewing3DAnglePreset 571

- Virtual Coordinate System 47

Virtual Coordinates

- getRect 797
- setRect 963

Visibility

- getDisplay 736

X**X1 Axis Properties**

- X1AxisDescending 575
- X1AxisLineDisplay 576
- X1AxisSide 578
- X1ExcludeMaxLabel 579
- X1ExcludeMinLabel 581
- X1LabelDisplay 582
- X1LabelFormat 584
- X1LabelFormatPattern 585
- X1LabelRotate 586
- X1LabelStagger 588
- X1LogScale 589
- X1MajorGridDisplay 591
- X1MajorGridStep 592
- X1MajorGridStepAuto 594
- X1MajorGridStyle 596
- X1MinorGridDisplay 598
- X1MinorGridStep 599
- X1MinorGridStepAuto 601
- X1MinorGridStyle 603
- X1MustIncludeZero 605
- X1OffScaleDisplay 606
- X1ScaleMax 608
- X1ScaleMaxAuto 610
- X1ScaleMin 611
- X1ScaleMinAuto 613
- X1TitleDisplay 614
- X1TitleString 615
- X1ZeroLineDisplay 616

Y**Y1 Axis Properties**

- Y1AxisDescending 617
- Y1AxisLineDisplay 618
- Y1AxisSide 619
- Y1ExcludeMaxLabel 621
- Y1ExcludeMinLabel 622
- Y1LabelDisplay 624
- Y1LabelFormat 625
- Y1LabelFormatPattern 626
- Y1LabelRotate 627
- Y1LabelStagger 628

Y1 Axis Properties (*continued*)

- Y1LogScale 629
- Y1MajorGridDisplay 630
- Y1MajorGridStep 632
- Y1MajorGridStepAuto 633
- Y1MajorGridStyle 634
- Y1MinorGridDisplay 636
- Y1MinorGridStepAuto 638
- Y1MinorGridStyle 639
- Y1MustIncludeZero 641
- Y1OffScaleDisplay 642
- Y1ScaleMax 643
- Y1ScaleMaxAuto 645
- Y1ScaleMin 646
- Y1ScaleMinAuto 648
- Y1TitleDisplay 649
- Y1TitleString 650
- Y1ZeroLineDisplay 651, 690

Y2 Axis Properties

- Y2AxisDescending 653
- Y2AxisLineDisplay 654
- Y2AxisSide 655
- Y2ExcludeMaxLabel 657
- Y2ExcludeMinLabel 658
- Y2LabelDisplay 660
- Y2LabelFormat 661
- Y2LabelFormatPattern 662
- Y2LabelRotate 663
- Y2LabelStagger 664
- Y2LogScale 666

Y2 Axis Properties (*continued*)

- Y2MajorGridDisplay 667
- Y2MajorGridStep 668
- Y2MajorGridStepAuto 670
- Y2MajorGridStyle 671
- Y2MinorGridDisplay 673
- Y2MinorGridStep 675
- Y2MinorGridStepAuto 676
- Y2MinorGridStyle 677
- Y2MustIncludeZero 679
- Y2OffScaleDisplay 680
- Y2ScaleMax 682
- Y2ScaleMaxAuto 683
- Y2ScaleMin 685
- Y2ScaleMinAuto 686
- Y2TitleDisplay 688
- Y2TitleString 689

Y3 Axis Properties

- Y3AxisDescending 691
- Y3LogScale 693
- Y3ZeroLineDisplay 695

Y4 Axis Properties

- Y4AxisDescending 696
- Y4LogScale 698
- Y4ZeroLineDisplay 700

Y5 Axis Properties

- Y5AxisDescending 702
- Y5LogScale 704
- Y5ZeroLineDisplay 706



Customer Connections

Where users partner with Information Builders to provide feedback and create quality documentation.

Partner With Us

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 join a usability study to get an early look at new documentation products and services. Your participation helps us create documentation products that meet your needs.

To send us feedback or make a customer connection, contact Michele Brady, Service Director, Technical Content Management at Michele_Brady@ibi.com.

Request Permission to Use Copyright Material

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.

Visit the Information Builders Technical Content Library

To visit the Technical Content Library, go to <http://documentation.informationbuilders.com/>.

WebFOCUS 8

WebFOCUS Graphics
Release 8.1 Version 03