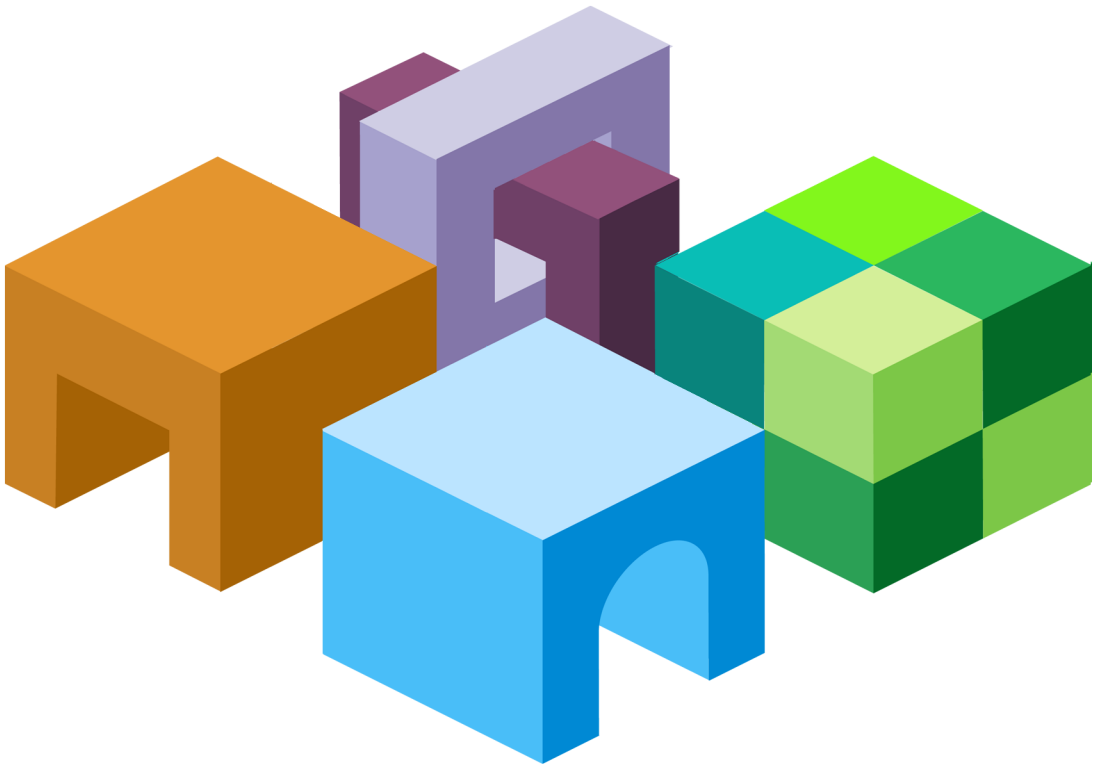

HYPERION® INTERACTIVE REPORTING

RELEASE 11.1.2

OBJECT MODEL AND DASHBOARD DEVELOPMENT SERVICES DEVELOPER'S GUIDE

VOLUME IV: OBJECT MODEL GUIDE TO PROPERTIES AND CONSTANTS

ORACLE
ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM

Interactive Reporting Object Model and Dashboard Development Services Developer's Guide, 11.1.2

Authors: EPM Information Development Team

Contents

1 Properties, Constants, and the Object Model

Welcome to the *Hyperion® System™ 9 BI+ ™ Oracle's Hyperion® Interactive Reporting, Object Model and Dashboard Development Services Development Guide, Volume IV: Properties and Constants*. This guide describes the properties (attributes) and constants associated with the Interactive Reporting Object Model, a hierarchical representation of Interactive Reporting.

Properties are characteristic of objects. Use properties to change object attributes. Properties may include object names, values, alignment, and colors. Whereas methods are associated with what objects do, properties are associated with what objects are. Constants are values that never changes. They define predetermined values. The Object Model contains a number of constants that are hardcoded in the client application.

Properties are simple string, numeric or true/false statements that can be set or read. For example, a section has a property called *Name*. Name can be set simple by assigning a string value; or read by assigning Name to a variable.

An object tracks its properties. Properties can be:

- **Read-only**— Designers can access, but cannot change data.
- **Read-write** —Designers can access and change values. Changing properties affects actions. For example, changing a toolbar property can make it visible or not visible.

Access object properties as follows:

- By name using [“ “] or a .
- By index using [ ]
- Use . when accessing a known object property
- Use [ ] when accessing elements within a collection.

This guide provides examples and code that you can use to create scripts for your dashboard sections.

2 Properties

In This Chapter

Accessibility (Property)

Applies To:

Document object

Description:

Enables Dashboard section designers to display another Dashboard section or object. The property settings are determined by the '508 Compliance Preferences' user setting in the Browse/Publish application, and used in the Oracle Enterprise Performance Management Workspace, Fusion Edition session. The property is not persisted with the document.

When an Interactive Reporting document is retrieved to the EPM Workspace, the Interactive Reporting Service determines if the document is to be shown with Accessibility features enabled (display the speed menu frame). Although this is specified by *508 Compliance Preferences*' setting in the Browse/Publish application, Interactive Reporting Service enables the Accessibility features when the document is displayed in EPM Workspace.

These rules determine which Boolean value applies to the property:

- If the accessibility option is enabled in the Browse/Publish application, the property is set to true.
- If the accessibility option is disabled, the property is set to false.

The default value for this property is false.

The property is not set when selections are made on Oracle's Hyperion® Interactive Reporting Web Client document link. Regardless of the accessibility status, the Interactive Reporting Web Client is not Section 508 compliant.

Action:

Read-only, Boolean

Example:

This example shows how to display another Dashboard section or object:

```
Documents["Sales.bqy"].Accessibility = true
ActiveDocument.Accessibility=true
```

AccessibilityText (Property)

Applies To:

(Live) BarChart object, (Live) BlockChart object, Bullet object, Checkbox object, CommandButton object, DropDown object, EmbeddedBrowser object, (Live) FunnelChart object, Hyperlink object, HzLine object, Line object, (Live) LineChart object, ListBox object, Picture object, (Live) PieChart object, (Live) RadarChart object, RadioButton object, Rect object, Slider object, Speedometer object, TextBox object, TextLabel object, Thermometer object, TrafficLight object, Vtline object

Description:

This property allows you to enter user-defined 508 text for embedded sections or graphics. The text is read to the 508 user when the object is encountered on the HTML page of the dashboard section. This property corresponds to the "Descriptive text for Intelligence iServer in (508) accessibility mode" field in the Accessibility dialog box. The default value for this property is no text.

Note: This property can be used in conjunction with the "AppendObjectText (Property)" on page 29, which enables you to append the object type after the accessibility text.

Action:

Read-write, String

Example

This example shows how to capture the accessibility text selected for a rectangle in the Console window::

```
Console.Write(Rect1.AccessibilityText = "The rectangle has a blue border and yellow
fill.")
```

Active (Property)

Applies To:

ChartSection object, (CubeQuery)QuerySection object, DataModelSection object, Document object, DashboardSection object, OLAPQuerySection object, PivotSection object, PluginDocument object, QuerySection object, ResultsSection object, Section object, TableSection object, ReportSection object

Description:

Section Object: Returns true if the section object refers to the current section.

Document Object: Returns true if the document object refers to the current document.

Do not use Session.Active in Interactive Reporting documents to be deployed in the EPM Workspace .

Action:

Read-only, Boolean

Example:

This example shows how to find the active section in the document:

```
var SectionCount = ActiveDocument.Sections.Count
for(j = 1 ; j <= SectionCount ; j++)
{
     if(ActiveDocument.Sections[j].Active == true)
         Alert ("The Active section is "+ActiveDocument.Sections[j].Name)
}
```

AdaptiveState (Property)

Applies To:

PluginDocument (Interactive Reporting Web Client only)

Description:

Returns the current Adaptive state mode to which the plug-in belongs.

Action:

Read-only

Constants:

The BqAdaptiveState constant group has these values:

- bqStateAnalyzeOnly
- bqStateAnalyzeProcess
- bqStateDataModelAnalyze
- bqStateNormal
- bqStateQueryAnalyze
- bqStateViewOnly
- bqStateViewProcess

Example:

This example shows how to use the AdaptiveState property to execute scripts conditionally:

```
var CurState = ActiveDocument.AdaptiveState
if( CurState == bqStateAnalyzeOnly || CurState == bqStateViewOnly)
        ActiveDocument.Sections["Start Here"].Activate()
else
        ActiveDocument.Sections["Query"].Activate()
```

AdjustableScale (Property)

Applies To:

ValuesAxis object

Description:

Enables the calculation of the minimum / maximum Y range based on the first two thousand bars. As you scroll through this range (moving to the right [X] or back [Z]}, or if any bar is above the maximum or below the minimum, the Y scale is expanded to all bars.

Action:

Read-Write, Boolean

Example:

This example shows how to calculate the minimum/maximum (Y) range based on the first two thousand bars for a chart:

```
ActiveDocument.Sections["Chart"].ValuesAxis.AdjustableScale = true
```

AliasTable (Property)

Applies To:

QueryOptions object

Description:

Sets and reads the name of the current alias table when the UseAliasTable (Property) is enabled.

An exception is thrown if this property is modified when the query is disconnected. An exception is thrown if this property is modified when the UseAliasTable (Property) is set to false.

Action:

Read-write, String

Example 1:

This example sets the alias table if the UseAliasTable is enabled:

```
var opts = Sections["Query"].QueryOptions;
var aliasTable = "";
if (opts.UseAliasTable == true)
    aliasTable = opts.AliasTable;
else {
```

Example 2:

This example shows how to read and set an alias table:

```
var opts = Sections["Query"].QueryOptions;
var aliasTable = "";
if (opts.UseAliasTable == true)
    aliasTable = opts.AliasTable;
else {
    opts.AliasTable = "Long Names";
    opts.UseAliasTable = true;
}
```

Alignment (Property)

Applies To:

Column object, Shape object, EmbeddedBrowser object, HyperLink object

Description:

Returns or sets the horizontal alignment of text in columns or shapes.

Action:

Read-write

Constants:

The BqHorizontalAlignment constant group consists of these values:

- bqAlignCenter
- bqAlignLeft
- bqAlignRight

Example:

This example shows how to change the horizontal alignment of column text:

```
var MyResults=ActiveDocument.Sections["SalesResults"]
var ColCount = MyResults.Columns.Count
for (j = 1 ; j <= ColCount ; j++)
     if (MyResults.Columns[j].DataType == bqDataTypeString)
         MyResults.Columns[j].Alignment = bqAlignLeft
     else
          MyResults.Columns[j].Alignment = bqAlignRight
```

AllowNonJoinedQueries (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the non-joined query value of a connection object. If set to true, the application processes queries with non-joined topics.

Action:

Read-write, Boolean

Example:

This example opens a connection file, SQL.oce, sets the user name and password, changes the connection file to support non-joined topics, and connects to the data source:

```
ActiveDocument.Sections["Query"].DataModel.Connection.Open("c:\\OCEs\\SQL.oce")
ActiveDocument.Sections["Query"].DataModel.Connection.Username = "hyperion"
ActiveDocument.Sections["Query"].DataModel.Connection.SetPassword("hyperionhyperion")
ActiveDocument.Sections["Query"].DataModel.Connection.AllowNonJoinedQueries = true
ActiveDocument.Sections["Query"].DataModel.Connection.Connect()
```

AppendObjectText (Property)

Applies To:

(Live) BarChart object, (Live) BlockChart object, Bullet object, Checkbox object, CommandButton object, DropDown object, EmbeddedBrowser object, (Live) FunnelChart object, Hyperlink object, HzLine object, Line object, (Live) LineChart object, ListBox object, Picture object, (Live) PieChart object, (Live) RadarChart object, RadioButton object, Rect object, Slider object, Speedometer object, TextBox object, TextLabel object, Thermometer object, TrafficLight object, Vtline object

Description:

This property enables you to append the object type after the accessibility text. This property corresponds to the "Include auto-generated descriptive text" field on the Accessibility dialog box. For example if the object is a rectangle, its object type is "Rectangle". If this property is set to true, the object type is appended after the accessibility text. If this property is set to false, the object type is not appended. For example if the text entered for the AccessiblityText property is "blue border and yellow fill, and the AppendObjectText property is true, the descriptive text in the EPM Workspace shows as "blue border yellow filled Rectangle." If AppendObjectText is false, the descriptive text in tc is "blue border yellow filled".

Note that the ObjectText is different for dashboard object types. For example:

- command button—command button, which is hard coded and localized
- rectangle— "rectangle",which is hard coded and localized
- embedded section object chart— "Chart1" which depends on Object name

Note: This property can be used in conjunction with the "AccessibilityText (Property) " on page 24, which enables you to enter user-defined 508 text for embedded sections or graphics.

Action:

Read-write, Boolean

Example

This example shows how to capture the accessibility text selected for a rectangle and append the object type in the Console window:

```
Rect1.AppendObjectText = true;
Console.Write( Rect1.AcessibilityText);
```

API (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the value of APIs associated with connection files.

Action:

Read-write

Constants:

The BqApi constant group consists of these values:

- bqApiCTLib
- bqApiAnalytic Services
- bqApiMetaCube
- bqApiNone
- bqApiODBC
- bqApiOLEDB
- bqApiOpenClient
- bqApiOracleExpress6
- bqApiPersonalOracleExpress5
- bqApiSQLNet

Example:

This example shows how to create a connection file and save it to a local file:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api =bqApiSQLNet
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.SaveAs("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\PlutoSQL.oce")
//Now use this connection in  a datamodel
ActiveDocument.Sections["SalesQuery"].DataModel.Connection.Open("c:\\Program Files\
\Hyperion\\BIPlus\\data\\Open Catalog Extensions\\PlutoSQL.oce")
```

AttributeDimension (Property)

Applies To:

Dimension object, OLAPLevelOrHierarchyNew collection

Description:

A read-only boolean property which can be used to test whether this is an attribute dimension or not. An attribute dimension level can be used with any base dimension to enhance the meaning of dimensional data. It displays in the Catalog Pane of the OLAP tree following the regular

Dimension name. The Attribute Dimension has text (attribute) following the name. Attribute Calculations also display as separate dimensions and the calculation functions are used in combination with any attribute dimension level or member.

Action:

Read only, Boolean

Example 1:

This example shows how to set the Attribute Dimension property to true:

```
ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Market"].AttributeDimension =
true
```

Example 2:

This example shows how to test whether a dimension is an attribute dimension or not:

```
var products = Sections["Query"].Catalog.Dimensions["Product"];
if (products.AttributeDimension == true)
    // Product is an attribute dimension
```

AutoAlias (Property)

Applies To:

DataModel object

Description:

Returns or sets the value of a Data Model AutoAlias property.

Action:

Read-write, Boolean

Example:

This example shows how to activate AutoAliasing and AutoJoining:

```
ActiveDocument.Sections["Query"].DataModel.AutoAlias = true
ActiveDocument.Sections["Query"].DataModel.AutoJoin = true
```

AutoCommit (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the value of a connection object AutoCommit property. Make this property false if your database does not support Auto commit. Use Auto Commit to send commit statements to the database server with each Interactive Reporting SQL statement to unlock used tables for which users are waiting.

Action:

Read-write, Boolean

Example:

This example shows how to create connections and save them locally to file:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api =bqApiSQLNet
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.AutoCommit = false
myCon.SaveAs("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\PlutoSQL.oce")
//Now use this connection in a datamodel
ActiveDocument.Sections["SalesQuery"].DataModel.Connection.Open("c:\\Program Files\
\Hyperion\\BIPlus\\data\\Open Catalog Extensions\\PlutoSQL.oce")
```

AutoFrequency (Property)

Applies To:

XAxisLabel object

Description:

Returns or sets the value of a Chart object AutoFrequency property. This enables the Chart function to choose the display frequency on the X-axis.

Action:

Read-write, Boolean

Example:

This example shows how to have a Chart X-axis use Auto Frequency:

```
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.AutoFrequency = true
```

AutoInterval (Property)

Applies To:

LeftAxis object

Description:

Returns or sets the value of a chart object AutoInterval (Property). This property enables the chart function to use the data interval on the left axis.

Action:

Read-write, Boolean

Example:

This example shows how to change a left-axis to support auto interval:

```
ActiveDocument.Sections["Chart"].ValuesAxis.LeftAxis.AutoInterval = true
```

AutoJoin (Property)

Applies To:

DataModel object

Description:

Enables a Data Model function to create automatic joins between topics that are added to it.

Action:

Read-write, Boolean

Example:

This example shows how to enable Auto Aliasing and Auto Joining:

```
ActiveDocument.Sections["Query"].DataModel.AutoAlias = true
ActiveDocument.Sections["Query"].DataModel.AutoJoin = true
```

AutoProcess (Property)

Applies To:

QuerySection object

Description:

Enables a Query to process automatically when it is opened or downloaded from the repository.

Action:

Read-write, Boolean

Example:

This example shows how to enable AutoProcess:

```
ActiveDocument.Sections["Query"].AutoProcess = true
```

AutoRefreshQuery (Property)

Applies To:

QueryOptions object

Description:

Queries the database automatically when an item is added to or removed from the data layout. If Auto-Refresh is disabled, you must click Process or use the Process (Method) to query the database whenever a change is made in the data layout. To improve system performance, it is recommended that when you enable Auto Generate Results option, you disable this property.

Actions:

Read-write, Boolean

Example:

This example shows how to enable the AutoRefreshQuery (Property):

```
ActiveDocument.Sections["Query"].QueryOptions.AutoRefreshQuery = true
```

AutoResize (Property)

Applies To:

ChartSection object

Description:

Enables all Chart components to be redrawn so that the Chart border expands to use the maximum height or width settings of the Contents Pane.

Action:

Read-write, Boolean

Example:

This example shows how to add chart title and expand the Chart border automatically:

```
ActiveDocument.Sections["Chart"].AutoResize = true
```

AutoRotate (Property)

Applies To:

Chart object

Description:

Rotates new bar charts 3º horizontally by 2º vertically when a component is added to the X axis. If a component is added to the Z axis, a rotation of 15º horizontally by 20º vertically is used. If this option is disabled, the rotation in the Horizontal\Vertical rotation degrees fields is used.

Action:

Read-write, boolean

Example:

This example shows how to enable AutoRotate (Property):

```
ActiveDocument.Sections["Chart"].AutoRotate = true
```

AutoScale (Property)

Applies To:

FactsAxis object, LeftAxis object, RightAxis object, NumericRange object

Description:

Enables a Chart axis scale to be selected between specified minimum and maximum values.

For a NumericRange object, this property enables the calculation of the lower and upper values scale of the data set for a guageautomatically. The scale is recalculated automatically when the data set is processed. The default value is a minimum of 0 and a maximum of 100 equally spaced ranges. To specify a minimum and maximum scale manually, disable this property.

Action:

Read-write, Boolean

Example:

This example shows how to enable auto-scaling on left and right values axis:

```
ActiveDocument.Sections["Chart"].ValuesAxis.LeftAxis.AutoScale = true
ActiveDocument.Sections["Chart"].ValuesAxis.RightAxis.AutoScale = true.
```

AutoSize (Property)

Applies To:

Legend object

Description:

Automatically resizes chart legend borders.

Action:

Read-write, Boolean

Example:

This example shows how to enable legend autosizing:

```
ActiveDocument.Sections["BooksChart"].Legend.AutoSize = true
```

Axis (Property)

Applies To:

TrendLine object, ReferenceLine object

Description:

Returns the axis to which the trend line is bound.

Action:

Read only

Constants:

The BqChartAxisType constant group consists of the following read only constants:

- bqChartNoAxis = 0
- bqChartXAxis = 1
- bqChartY1Axis = 4
- bqChartYAxis = 2
- bqChartZAxis = 3

Example:

This example shows how to read the axis on which the trend line has been applied:

```
Alert(ActiveDocument.Sections["Chart6"].TrendLines[1].Axis)
```

AxisPlotValues (Property)

Applies To:

Line Chart Facts object

Description:

Returns or sets the axis plot value of fact in Line Charts. This property corresponds to the features on the Line Chart Axis Properties dialog box.

Action:

Read-write

Constants:

The BqChartAxisPlotValue constant group consists of the bqChartAxisPlotPrimary and bqChartAxisPlotSecondary constants.

Example:

This example shows how to set the axis plot value to the primary or left axis:

```
ActiveDocument.Sections["Chart"].Facts["Unit_Sales"].AxisPlotValue=
bqChartAxisPlotPrimary
```

AxisType (Property)

Applies To:

CategoryItems collection

Description:

Returns an enumerated type that represents the axis type (X, Y or Z).

Action:

Read-only

Constants:

The BqChartAxisType group consists of these values:

- bqChartXAxis
- bqChartYAxis
- bqChartZAxis

Example:

This example shows how to determine the Chart axis type:

```
switch(ActiveDocument.Sections["Chart"].XCategories.AxisType )
{
case bqChartXAxis:
Alert("The axis is X)
Break
case bqChartYAxis:
Alert("The axis is Y)
Break
case bqChartZAxis:
Alert("The axis is Z)
Break
}
```

BackgroundAlternateColor (Property)

Applies To:

ResultsSection object, TableSection object, ReportTable object

Description:

Defines the background color of staggered (alternate) table rows.

Action:

Read–write, BqColorType

Constants:

The BackgroundAlternateColor property uses the BqColorType constant group, which consists of these values:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937
- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119
- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120

- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410
- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232
- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637
- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215
- bqYellow = 16776960

Example:

This example shows how to make the alternate background color of every second row light yellow:

```
ActiveDocument.Sections["Results"].BackgroundAlternateColor =
bqLightYellowActiveDocument.Sections["Results"].BackgroundAlternateFrequency = 1
```

BackgroundAlternateFrequency (Property)

Applies To:

ResultsSection object, TableSection object, ReportTable object

Description:

Specifies how often alternate colored rows occur, such as every third row.

Action:

Read-write, Number

Example:

This example shows how to use alternate colored rows on every other row. It also changes the background alternate color to light yellow:

```
ActiveDocument.Sections["Table2"].BackgroundAlternateColor =
bqLightYellowActiveDocument.Sections["Table2"].BackgroundAlternateFrequency = 1
```

BackgroundColor (Property)

Applies To:

CellFormat, object, ResultsSection object, TableSection object, ReportTable object, Measure object

Description:

Sets the background color of rows in a Table section.

Action:

Read-write, BqColorType constant group

Constants:

The BackgroundColor property uses the BqColorType constant group, which includes:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937
- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119

- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120
- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410
- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232
- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637
- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215
- bqYellow = 16776960

The BackgroundColor (Property) for the CellFormat object uses the BqDefaultFormatsColor constant group, which includes the following values:

- bqDefaultFormatsBlack = 0
- bqDefaultFormatsBlue = 255
- bqDefaultFormatsCrimson = 8323072
- bqDefaultFormatsCyan = 65535
- bqDefaultFormatsDarkGrey = 8355711
- bqDefaultFormatsGreen = 65280
- bqDefaultFormatsJade = 32639
- bqDefaultFormatsLavender = 8323327
- bqDefaultFormatsLightGrey = 12566463
- bqDefaultFormatsLightYellow =16777104
- bqDefaultFormatsMustard = 8355584
- bqDefaultFormatsNavy = 127
- bqDefaultFormatsOlive =32512
- bqDefaultFormatsPurple = 8323199
- bqDefaultFormatsRed = 16711680
- bqDefaultFormatsWhite = 16777215
- bqDefaultFormatsYellow = 16776960

Example:

This example shows how to make the background color of rows in a Table section light green:

```
ActiveDocument.Sections["Table"].BackgroundColor = bqLightGreen
```

BackgroundShowAlternateColor (Property)

Applies To:

ResultsSection object, TableSection object, ReportTable object

Description:

Enables alternate row colors at the column or table level.

Action:

Boolean

Example:

This example shows how to enable the show alternate color feature and have alternate colored rows on every other row. It also sets the background alternate color to bright green:

```
ActiveDocument.Sections["MusicTable"].BackgroundShowAlternateColor = true
```

```
ActiveDocument.Sections["MusicTable"].BackgroundAlternateFrequency = 1
ActiveDocument.Sections["MusicTable"].BackgroundAlternateColor = bqBrightGreen
```

BeginLimitName (Property)

Applies To:

Parentheses object

Description:

When Parentheses (Collections) is invoked, the BeginLimitName (Property) enables you to set the limit value before the first parentheses is inserted. This property is often used with the "EndLimitName (Property)" on page 94.

Action:

Read-only, BeginLimitName as String

Example:

This example shows how to display the name of the beginning limit value enclosed in a parenthetical expression on the limit line:

```
Alert(ActiveDocument.Sections["Query"].Limits.Parentheses["State
Province,City"].BeginLimitName)
```

BorderColor (Property)

Applies To:

CellFormat object, ResultsSection object, TableSection object

Description:

Sets the color of a table border.

Action:

Read-write, BqColorType

Constants:

The BorderColor property uses the BqColorType constant group, which consists of these values:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937

- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119
- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120
- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410
- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232
- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637
- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215

- bqYellow = 16776960

The BorderColor (Property) for the CellFormat object uses the BqDefaultFormatsColor constant group, which includes the following values:

- bqDefaultFormatsBlack = 0
- bqDefaultFormatsBlue = 255
- bqDefaultFormatsCrimson = 8323072
- bqDefaultFormatsCyan = 65535
- bqDefaultFormatsDarkGrey = 8355711
- bqDefaultFormatsGreen = 65280
- bqDefaultFormatsJade = 32639
- bqDefaultFormatsLavender = 8323327
- bqDefaultFormatsLightGrey = 12566463
- bqDefaultFormatsLightYellow =16777104
- bqDefaultFormatsMustard = 8355584
- bqDefaultFormatsNavy = 127
- bqDefaultFormatsOlive =32512
- bqDefaultFormatsPurple = 8323199
- bqDefaultFormatsRed = 16711680
- bqDefaultFormatsWhite = 16777215
- bqDefaultFormatsYellow = 16776960

Example:

This example shows how to specify a red table border:

```
ActiveDocument.Sections["Table2"].BorderColor = bqRed
```

BorderStyle (Property)

Applies To:

CellFormat object

Description:

Sets the style of the border.

Action:

Read-write, BqBorderStyle

Constants:

The BqBorderStyle constant group consists of the following values:

- bqBorderStyle_Horizontal = 2
- bqBorderStyle_None = 0
- bqBorderStyle_Raise1 = 4
- bqBorderStyle_Sink1 = 5
- bqBorderStyle_VertAndHorz = 3
- bqBorderStyle_Vertical = 1

Example:

This example shows how to set the default border style for a Pivot data value to "raised":

```
DefaultFormats.Pivot.DataValues.BorderStyle = bqBorderStyle_Raise1
```

BorderWidth (Property)

Applies To:

ResultsSection object, Table object, ReportTable object

Description:

Sets the width of borders (in points).

Action:

Read-write, Number

Example:

This example shows how to specify a border width of three points:

```
ActiveDocument.Sections["Results"].BorderWidth = 3
```

BottomMargin (Property)

Applies To:

ReportSection object

Description:

Sets the bottom margin of reports. Margins are set for the entire report.

Note: If the "SuspendCalculation (Property)" on page 226 is true (which is the), use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Number

Example:

This example shows how to make a bottom report margin .25 inches:

```
ActiveDocument.Sections["Report"].BottomMargin = .25
```

BrushStyle (Property)

Applies To:

Legend Collection

Description:

Sets the brush style property of a legend item.

Action:

Read-write, Number

Constants:

The BrushStyle property uses the BqBrushStyle constant group, which consists of these values:

- bqBrushStyleCross
- bqBrushStyleDiagCross
- bqBrsuhStyleDiagDown
- bqBrushStyleDiagUp
- bqBrushStyleHollow
- bqBrushStyleHorizontal
- bqBrushStyleSolid
- bqBrushStyleVertical

Example:

This example shows how to set the brush style of a legend item property to solid:

```
ActiveDocument.Sections["Chart2"].Legend.Items[1].Fill.BrushStyle=bqBrushStyleSolid
```

CatalogDisplayMembers (Property)

Applies To:

QueryOptions object

Description:

Sets the maximum number of members that can be displayed at any given level.

Action:

Read-write, BQOLAPCatalogDisplayMembers constant group

Constants:

The BQOLAPCatalogDisplayMembers constant group consists of these values:

- bqOLAPCatalogDisplayMembers10
- bqOLAPCatalogDisplayMembers100
- bqOLAPCatalogDisplayMembers20
- bqOLAPCatalogDisplayMembers250
- bqOLAPCatalogDisplayMembers5
- bqOLAPCatalogDisplayMembers50
- bqOLAPCatalogDisplayMembers500
- bqOLAPCatalogDisplayMembers10

Example:

This example shows how to display an unlimited number of members:

```
ActiveDocument.Sections["Query"].QueryOptions.CatalogDisplayMembers =
bqOLAPCatalogDisplayMembersUnlimited
```

CellValue (Property)

Applies To:

Embedded Pivot Objects

Description:

Returns the value in a cell that was double-clicked. Accessible for Dashboard Pivot Embedded Sections Objects (ESOs) included in an Interactive Reporting document. You can only use this property when Pivots are active.

Reading the CellValue property is the same as accessing values by clicking cell in a Pivot Embedded Section Object. The CellValue Property is read-only. The property returns a Value

of type VARIANT representing the unformatted JavaScript content of the cell. The user must format it using JavaScript. Default value of the property is the value of the cell selected.

If no data exists at the intersection, a catchable “no such property” exception is returned, and the script halts without a try/catch statement. The scripts continues only when the catch is employed.

The CellValue (Property) is available for Oracle's Hyperion® Interactive Reporting Studio and Interactive Reporting Web Client. It is unavailable for the EPM Workspace or the Jobs/Scheduler.

Action:

Read only, undefined

Example:

This example shows how to return the value of a double-clicked cell. It includes a try-catch statement and the default values.

```
//Clear all TextBoxes except TextBox5

TextBox1.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

TextBox4.Text = ActiveSection.Name

var ActionName = "CellValue"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")

{

TextBox1.Text ="Step 1a"

try

{

PivotESOTextBox.Text = "CellValueDefault is: " + ActiveSection.Shapes["Pivot1"].CellValue

}

catch(e)

{

TextBox2.Text = "Caught: " + e.toString()

}

TextBox1.Text ="Step 1b"

try
```

```
{

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nCellValuetoString is: " +
ActiveSection.Shapes["Pivot1"].CellValue.toString()

}

catch(e)

{

TextBox2.Text = "Caught: " + e.toString()

}

}

else

{

TextBox1.Text ="Step 2"

try

{

ActiveDocument.Sections["Pivot"].Facts.CellValue = TextBox5.Text

}

catch(e)

{

TextBox2.Text = "Caught: " + e.toString()

}

TextBox1.Text ="Step 3a"

try

{

PivotESOTextBox.Text = "CellValueDefault is: " + ActiveSection.Shapes["Pivot1"].CellValue

}

catch(e)

{

TextBox2.Text = "Caught: " + e.toString()

}

TextBox1.Text ="Step 3b"

try

{
```

```
PivotESOTextBox.Text = "CellValuetoString() is: " + PivotESOTextBox.Text + "\r\n" +
ActiveSection.Shapes["Pivot1"].CellValue.toString()

}

catch(e)

{

TextBox2.Text = "Caught: " + e.toString()

}

}

TextBox1.Text ="End " + ActionName
```

In this example, these values are returned:

CellValueDefault is: 5361.3

CellValuetoString is: 5361.3

ChartType (Property)

Applies To:

ChartSection object

Description:

Sets which type of chart is displayed in a Chart section.

Action:

Read-write

Constants:

The BqChartType constant group consists of these values:

- bqChartTypeArea
- bqChartTypeAreaLine
- bqChartTypeBarLine
- bqChartTypeClusterBar
- bqChartTypeHorizontalBar
- bqChartTypeHorizontalStackBar
- bqChartTypeLine
- bqChartTypeNone
- bqChartTypePie
- bqChartTypeRibbon

- bqChartTypeStackArea
- bqChartTypeVerticalBar
- bqChartTypeVerticalStackBar

Example:

This example shows how to change chart properties based on the chart type:

```
if (ActiveDocument.Sections["Chart"].ChartType == bqChartTypeBarLine)
{
ActiveDocument.Sections["Chart"].BarLineChart.ClusterBy = bqClusterByZ
ActiveDocument.Sections["Chart"].BarLineChart.IgnoreNulls = false
ActiveDocument.Sections["Chart"].BarLineChart.ShiftPoints = bqShiftCenter
ActiveDocument.Sections["Chart"].BarLineChart.StackClusterType = bqBarLineCluster
ActiveDocument.Sections["Chart"].BarLineChart.ShowBarValues = false
}
```

Checked (Property)

Applies To:

ControlsCheckBox object, ControlsRadioButton object

Description:

Sets the selection of a check box or radio button controls.

Action:

Read-write, Boolean

Example:

This example shows how to change a Radio button or check box selection, assuming you are using: RadioButton1 and CheckBox1 in the Dashboard:

```
if (RadioButton1.Checked == true)
     CheckBox1.Checked = false
else
     CheckBox1.Checked = true
```

ClickX (Property)

Applies To:

bqoEvent object

Description:

Represents the mouse pointer horizontal position on a picture or table object when the bqoEvent passes event-related information as an optional parameter to an Event handler (OnClick handler). If the ClickX values cannot be read, 0 is returned.

Action:

Read-write, Number

Example:

This example shows to read the values on different sectors of a picture on an OnClick Event. Values are written to the Console:

```
TBConsole.Text = TBConsole.Text + "\r\nPic1 Start"
TBConsole.Text = TBConsole.Text + "\r\nXCoord: " + bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

if (bqoEvent.ClickX > 390 && bqoEvent.ClickX < 480 && bqoEvent.ClickY >280 &&
bqoEvent.ClickY <425)
{
TBConsole.Text = TBConsole.Text + "\r\nFront Wheel"
}
else
{
TBConsole.Text = TBConsole.Text + "\r\nNot a Wheel"
}

TBConsole.Text = TBConsole.Text + "\r\nXCoord: " +  bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

bqoEvent.ClickX = bqoEvent.ClickX - 50
bqoEvent.ClickY = bqoEvent.ClickY + 50

TBConsole.Text = TBConsole.Text + "\r\nXCoord: " +  bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

Picture2.OnClick(bqoEvent)

TBConsole.Text = TBConsole.Text + "\r\nPic1 End"
```

ClickY (Property)

Applies To:

bqoEvent object

Description:

Represents the mouse pointer vertical position on a picture or table object when the bqoEvent passes event-related information as an optional parameter to an Event handler (OnClick handler). If the ClickX values cannot be read, 0 is returned.

Action:

Read-write, Number

Description:

This example shows to read the values on different sectors of a picture on an OnClick Event. Values are written to the Console:

```
TBConsole.Text = TBConsole.Text + "\r\nPic1 Start"
TBConsole.Text = TBConsole.Text + "\r\nXCoord: " + bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

if (bqoEvent.ClickX > 390 && bqoEvent.ClickX < 480 && bqoEvent.ClickY >280 &&
bqoEvent.ClickY <425)
{
TBConsole.Text = TBConsole.Text + "\r\nFront Wheel"
}
else
{
TBConsole.Text = TBConsole.Text + "\r\nNot a Wheel"
}

TBConsole.Text = TBConsole.Text + "\r\nXCoord: " +  bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

bqoEvent.ClickX = bqoEvent.ClickX - 50
bqoEvent.ClickY = bqoEvent.ClickY + 50

TBConsole.Text = TBConsole.Text + "\r\nXCoord: " +  bqoEvent.ClickX
TBConsole.Text = TBConsole.Text + "\r\nYCoord: " + bqoEvent.ClickY

Picture2.OnClick(bqoEvent)

TBConsole.Text = TBConsole.Text + "\r\nPic1 End"
```

ClientScriptStatus (Property)

Applies To:

ActiveSection object, EmbeddedBrowser object

Description:

Client-side JavaScript instructs the server code to run or not run, using these status indicators:

- ActiveSection.ClientScriptStatus
- Object.ClientScriptStatus

For example, if the client-side JavaScript (which might be activated by a command button) determines a user did not make a selection, or entered alphabetic instead of numeric data, the server side JavaScript does not run until the user enters the correct data.

Use the ActiveSection.ClientScriptStatus status indicator to coordinate actions between objects. This indicator takes a Boolean value.

Use the Object.ClientScriptStatus status indicator to initiate actions for independent objects. The *object* in the Object.ClientScriptStatus status indicator is replaced with the object name, such as CommandButton1.ClientScriptStatus (*object*) cannot exist by itself and takes a Boolean value. Independent objects use the ActiveSection's script status settings.

Action:

Read-write, Boolean

Example:

This example shows how to set the indicator value for embedded objects to true:

```
EmbeddedBrowser1.ClientScriptStatus=true
```

Clusterby (Property)

Applies To:

BarChart object, BarLineChart object

Description:

Determines the clustering used when displaying Bar or Bar Line charts.

Action:

Read-write

Constants:

The BqClusterBarType constant group consists of the bqClusterByY and bqClusterByZ values.

Example:

This example shows how to cluster data according to the values on the Z-axis:

```
ActiveDocument.Sections["Chart"].BarChart.ClusterBy = bqClusterByZ
```

Color (Property)

Applies To:

ColorRange object, Font object, Fill object, Line object

Description:

Sets the color of text associated with a font object. This property may be set using the values in the BqColorType constant group or by using a hexadecimal number that represents a RGB color value.

For a gauge object (Bullet, Speedometer, Thermometer, TrafficLight) sets the color for values assigned in the numeric scale.

Action:

Read-write, Number

Constants:

These values are some of those in the BqColorType constant group. For a complete list, see the Interactive Reporting Object Model Script Editor:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937
- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119
- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120
- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410
- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232

- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637
- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215
- bqYellow = 16776960

Example:

This example shows how to set the color, width and dash style of the border of an Dashboard text label box:

```
MyColor = ActiveDocument.Sections["Dashboard"].Shapes["TextLabel"]
MyColor.Line.Color = bqRed
MyColor.Line.Width = 4
MyColor.Line.DashStyle = bqDashStyleDotDotDash
```

ColorScheme (Property)

Applies To:

ChartSection object

Description:

Sets the chart color scheme for data points in the Chart section.

- Standard—default color scheme used for charts created in Release 8.4 and later.
- Legacy—color scheme used for charts in Release 8.3 and earlier
- Custom—color scheme defined by the user

This property corresponds to the UseLegacyColors (Property) in Release 11.1.1.3 and earlier. If this property is set to bqChartColorSchemeLegacy for new charts, the colors in the Legacy color

scheme (in Release 8.3 and earlier) are used. The value of UseLegacyColors property is mapped to ColorScheme property as follows:

Table 1 ColorScheme (Property) to UseLegacyColors (Property)

ColorScheme	UseLegacyColors
bqChartColorSchemeStandard	false
bqChartColorSchemeLegacy	true
bqChartColorSchemeCustom	false

Action:

Read-write, BqChartColorScheme

Constants:

The BqChartColorScheme constant group (read only) consists of the following values:

- bqChartColorSchemeCustom = 3
- bqChartColorSchemeLegacy = 2
- bqChartColorSchemeStandard = 1

Example:

This example shows how to write the standard color scheme to the Console:

```
Console.Write(ActiveDocument.Sections["Chart"].ColorScheme = bqChartColorSchemeStandard)
```

ColumnType (Property)

Applies To:

Column object

Description:

Returns a value that represents the type of Results or Table column. Possible column types are: Normal, Computed, Date and Grouped.

Action:

Read-only

Constants:

The BqColumnType constant group consists of these values:

- bqColumnNone
- bqComputedColumn

- bqDateColumn
- bqGroupedColumn
- bqStandardColumn

Example:

This example shows how to determine the column type in a Results section:

```
for (j = 1 ; j < = ActiveDocument.Sections["Results"].Columns.Count ;j++)
{
          MyCol = ActiveDocument.Sections["Results"].Column[j].
          switch (MyCol.Type)
          {
                    case bqComputedColumn:
                    Alert ("The column named "+MyCol.Name + "is a Computed column")
                    Break
                    case bqDateColumn:
                    Alert ("The column named "+MyCol.Name + "is a Date column")
                    Break
                    case bqGroupedColumn:
                    Alert ("The column named "+MyCol.Name + " is a Grouped column")
                    Break
                    case bqStandardColumn:
                    Alert ("The column named "+MyCol.Name + "" is a Standard column")
                    Break
           }
}
```

Comments (Property)

Applies To:

Bullet object, (Live) BlockChart object, (Live) FunnelChart object, (Live) LineChart object, (Live) PieChart object, (Live) RadarChart object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Descriptions:

Returns or sets a comment (tooltip) for a gauge object or (Live) chart object. Comments typically show description or supplemental information about the object and display when the user hovers the cursor over an object without selecting it.

Action:

Read-write, String

Example:

This example shows how to set a comment for a bullet object:

```
ActiveDocument.Sections["Dashboard"].Shapes["Bullet"].Comments = "SouthWestern Sales -
1st quarter"
```

Connected (Property)

Applies To:

Connection object

Description:

Returns a value that represents the connection status of a connection object. This property returns true if the user is connected to the data source; otherwise, it returns false.

Action:

Read-only, Boolean

Example:

This example shows how to check the connection status of a connection object and prompt users to connect:

```
var MyCon =ActiveDocument.Sections["SalesQuery"].DataModel.Connection
if (MyCon.Connected ==false)
{
    if (Alert
    ("Do you want to connect to the database?", "Get Connected"," Yes"," No")==1)
        MyCon.Connect()
 }
```

ContainsHybridAnalysisData (Property)

Applies To:

OLAPLevelorHierarchy

Description:

Indicates if a dimension level or hierarchy has hybrid analysis values available.

Action:

Read-only, Boolean

Example:

This example shows how to enable Hybrid Analysis for an OLAP query and indicate that Hybrid Analysis is available in an Alert box:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.EnableForHybridAnalysis= true
ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Location"].ContainsHybridAnalys
isData = true
Alert("Dimension has Hybrid Analysis")
```

Count (Property)

Applies To:

AggregateLimits collection, AliasTableName collection, AppendQueries collection, Association collection, AxisLabels collection, CategoryItems collection, ColorRanges collection, Columns collection, Controls collection, ControlsDropDown object, ControlsListBox object, ColorRanges collection, , DerivableQueries collection, DerivedItems collection, DerivedTables collection, DMCatalogItem collection, DMResults collection, Documents collection, EventScripts collection, Fact collection, Joins collection, Images collection, Limits collection, LimitValues collection, LocalJoins, LocalResults, OLAPCatalogNew object, OLAPDimensionNew collection, OLAPLabel, OLAPLabels, OLAPLevelOrHierarchyNew collection, OLAPMeasure, OLAPMeasures, OLAPSlicer, OLAPSlicers, Parentheses collection, PivotLabels collection, PivotLabelTotals collection, QueryLabel object, RecentFiles collection, Requests collection, Results collection, Sections collection, SortItems collection, Sorts, TargetFact collection, Toolbars collection, TopicItems collection, Topics collection, TopLabels collection, TrendLines collection

Description:

Returns the number of items in a collection. The Count property is a standard property of all collections.

Action:

Read-only, Number

Example:

This example shows how to determine the number of sections in a document and columns in a Results section:

```
var NumSections = ActiveDocument.Sections.Count
var NumColumns = ActiveDocument.Sections["Results"].Columns.Count
```

CreatedAppType (Property)

Applies To:

Document object

Description:

Returns the application name in which a document was created.

Action:

Read-only as BqAppTyoe

Constants:

The CreatedAppType (Property) uses the BqAppType constant group, which consists of these values:

- bqAppTypeDesktopClient = 1
- bqApTypePlugInClient = 2
- bqAppTypeScheduler = 3
- bqAppTypeSmartViewClient = 4
- bqAppTypeThinClient = 5
- byAppTypeUnknown = 6

Example:

This example shows how to display the application type for the Chart.bqy in an Alert box:

```
Alert(Documents["Sample - charts.bqy"].CreatedAppType =bqAppTypeDesktopClient)
```

CreatedAppVersion (Property)

Applies To:

Document object

Description:

Returns the application version in which a document was created.

Action:

Read-only as String

Example:

This example shows how to display the application type for the Chart.bqy in an alert box:

```
Alert(Documents["Sample - charts.bqy"].CreatedAppVersion = 8.4)
```

CubeName (Property)

Applies To:

OlapConnection object

Description:

Returns the name of a cube object.

Action:

Read-only, String

Example:

This example shows how to return the cube name for an OLAP query in an Alert box:

```
Alert(ActiveDocument.Sections["OLAPQuery"].Connection.CubeName)
```

CurrentDir (Property)

Applies To:

Application object

Description:

Returns a value that represents the working directory of an application. This is the directory Interactive Reporting uses when using relative referencing.

Action:

Read-write, String

Example:

This example shows how to change the working directory:

Note: JavaScript treats "\" as a special character.

```
var MyDir = "c:\Documents\Demos\JavaScript"
Application.CurrentDir = MyDir
```

CSSExport (Property)

Applies To:

PivotSection object, ResultsSection object, ReportSection object,

Description:

Exports HTML pages with a Cascading Style Sheet (CSS file). In the user interface, CSS files apply a common appearance to related HTML pages. Interactive Reporting creates CSS files by default when exporting most sections to HTML format. Sections have the same high fidelity rendition in HTML as they do in Interactive Reporting. For ease of use, Interactive Reporting enables you to embed style information in HTML files.

Action:

Read-write, Boolean

Example:

This example shows how to export a CSS file with the Report section:

```
ActiveDocument.Sections["Report"].CSSExport = true
```

CustomSQL (Property)

Applies To:

Limit object

Description:

Returns or sets the value of the CustomSQL strings in a limit.

Action:

Read-write, String

Example:

This example shows how to set the value of the custom SQL for a limit:

```
var SQLString = "SELECT Name From Customers WHERE Cust_ID > 200"
ActiveDocument.Sections["Query"].Limits[1].CustomSQL = SQLString
```

DashStyle (Property)

Applies To:

Line object

Description:

Returns or sets the type of border style for a shape or control.

Action:

Read-write, BqDashStyle constant

Constants:

The BqDashStyle constant group consists of these values:

- bqDashStyleDash
- bqDashStyleDot

- bqDashStyleDotDash
- bqDashStyleDotDotDash
- bqDashStyleSolid

Example:

This example shows how to change border color, width, and the dash style of a rectangle:

```
MyRectangle = ActiveDocument.Sections["Dashboard"].Shapes["Rectangle"]
MyRectangle.Line.Color = bqRed
MyRectangle.Line.Width = 4
MyRectangle.Line.DashStyle = bqDashStyleDotDotDash
```

Database (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the name of the database vendor and version number.

Action:

Read-write

Constants:

These values consists of BqDataBase constant group:

- bqDatabaseAS400
- bqDatabaseDB2
- bqDatabaseDB2Olap
- bqDatabaseEssbase
- bqDatabaseInformix7
- bqDatabaseMicrosoftSQLServer7
- bqDatabasenone
- bqDatabaseODBC
- bqDatabaseOLEDB
- bqDatabaseOracle8
- bqDatabaseRedBrick5Warehouse
- bqDatabaseSybaseSystem
- bqDatabaseTeraData

For all values, see the Interactive Reporting Object Model Script Editor.

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api =bqApiSQLNet
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.SaveAs("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\PlutoSQL.oce")
//Now use this connection in  a datamodel
ActiveDocument.Sections["SalesQuery"].DataModel.Connection.Open("c:\\Program Files\
\Hyperion\\BIPlus\\data\\Open Catalog Extensions\\PlutoSQL.oce")
```

DatabaseList (Property)

Applies To:

Connection object, MetaDataConnection object (Sybase Only and SQL Server only)

Description:

Returns or sets the list of databases to which the .Interactive Reporting database connection file (.oce) can connect.

Action:

Read-write, String

Example:

This example shows how to create an Interactive Reporting database connection file and set the list of databases to which the Interactive Reporting database connection file can connect:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api =bqApiSQLNet
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DatabaseList = "master, customer, sales"
myCon.SaveAs("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\PlutoSQL.oce"
//Now use this connection in a data model
ActiveDocument.Sections["SalesQuery"].DataModel.Connection.Open("c:\\Program Files\
\Hyperion\\BIPlus\\data\\Open Catalog Extensions\\PlutoSQL.oce")
```

DatabaseName (Property)

Applies To:

DMCatalogItem object

Description:

Returns the name of the database associated with a table in the Table catalog. To retrieve this property correctly, you must connect to the database successfully, expand the Query Catalog tree manually (in Designer or the web client) or use the Catalog.Refresh() method, and use this property.

Action:

Read-only

Example 1:

This example shows how to expand the Table Catalog, count the number of catalog items, and write name of the database associated with a table to the Console window. The example assumes that there is a connection to the database, and the Table Catalog has been refreshed and expanded.

```
var  TableCatalog = ActiveDocument.Sections["SalesQuery"].DataModel.Catalog
var TableCount = ActiveDocument.TableCatalog.CatalogItems.Count
for (j=1;j<=TableCount;j++)
Console.Writeln (TableCatalog.CatalogItems[j].Name)
```

Example 2:

This example shows how to return the connection status of the database, connect to the database, refresh the Table Catalog, expand the Table Catalog, count the number of catalog items, and write the database name associated with a table to Console window. When using this property, the database connection must have the Prompt for Database Name enabled; otherwise, database names are not tricked in the Table Catalog.

```
if (!ActiveDocument.Sections["Query"].DataModel.Connection.Connected)
ActiveDocument.Sections["Query"].DataModel.Connection.Connect(true)
var  TableCatalog =
ActiveDocument.Section["Query"].DataModel.CatalogTableCatalog.Refresh();
var TableCount = TableCatalog.CatalogItems.Count for (j=1;j<=TableCount; j++)
Console.Writeln (TableCatalog.CatalogItems[j].DatabaseName
+"."+TableCatalog.CatalogItems[j].Owner+"."+ TableCatalog.CatalogItems[j].Name)
```

DatabaseTotals (Property)

Applies To:

OLAPQuery object

Description:

Returns database totals when the OLAPQuery is processed.

Action:

Read-write, Boolean

Example 1:

This example shows how to return database totals when the OLAP query is processed:

```
ActiveDocument.Sections["OLAPQuery"].DatabaseTotals = true
```

Example 2:

This example shows how to use an *if...else* statement to show if database totals are active or not. The script is attached to a command button and the message is displayed in a text box:

```
dbTotals = ActiveDocument.Sections["OLAPQuery"].DatabaseTotals
if (dbTotals )
{
TextBox1.Text = "Database Totals are on"
}
else
{
    TextBox1.Text = "DB Totals are off"
}
```

Constants:

The DataFunction property uses the BqDataFunction constant group, which consists of these values:

- bqDataFunctionAverage
- bqDataFunctionColumnName
- bqDataFunctionCount
- bqDataFunctionCountDistinct
- bqDataFunctionIncrease (Pivot Totals properties, not Facts)
- bqDataFunctionMaximum
- bqDataFunctionMinimum
- bqDataFunctionNone
- bqDataFunctionNonNullAverage
- bqDataFunctionNonNullCount
- bqDataFunctionPercentOfCategory
- bqDataFunctionPercentOfColumn
- bqDataFunctionPercentOfGrand
- bqDataFunctionPercentOfIncrease

- bqDataFunctionPercentOfRow
- bqDataFunctionSum

DataFunction (Property)

Applies To:

Chart and Pivot Facts objects, TargetFact object

Description:

Returns aggregate values that summarize groupings of data when applied to Chart, Pivot, gauges and (Live) Chart facts. From the user interface, data functions are available from the short cut menu and Chart and Pivot menus only if fact values are selected.

Use data functions to show the kind of values represented in Chart and Pivot reports, such as total, average, and maximum product sale by Quarter. You can use these data functions for Pivot and Chart Facts:

- Sum (default)
- Average
- Count
- Maximum
- Minimum
- Percent Grand
- Percent Column
- Percent Row
- Null Count
- Non-Null Count

Action:

Read-only

Constants:

The DataFunction property uses the BqDataFunction constant group, which consists of these values:

- bqDataFunctionAverage
- bqDataFunctionCount
- bqDataFunctionIncrease (Pivot Totals properties, not Facts)
- bqDataFunctionMaximum
- bqDataFunctionMinimum

- bqDataFunctionNone
- bqDataFunctionNonNullAverage
- bqDataFunctionNonNullCount
- bqDataFunctionNullCount
- bqDataFunctionPercentOfColumn
- bqDataFunctionPercentOfRow
- bqDataFunctionPercentofGrand (For Totals, not Facts)
- bqDataFunctionSum

Example:

This example shows how to set the *Product Line* TopLabels column in the Pivot section to the average data function:

```
ActiveDocument.Sections["SalesPivot"].TopLabels["Product Line"].Totals[2].
DataFunction=bqDataFunctionAverage
```

DataType (Property)

Applies To:

Column object, Requests collection

Description:

Returns the data type associated with an object.

Action:

Read-only

Constants:

The BqDataType constant group consists of these values:

- bqDataTypeDate
- bqDataTypeInteger
- bqDataTypeNone
- bqDataTypeNumber
- bqDataTypeString

Example:

This script example shows how to return the data type associated with all columns in a Results section:

```
var ColCount = ActiveDocument.Sections["Results"].Columns.Count
```

```
for (j = 1 ; j <= ColCount ; j++)
{
Console.Writeln(ActiveDocument.Sections["Results"].Columns[j].DataType)
}
```

DBLibAllowChangeDatabase (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib Only. The DBLibAllowChangeData (Property) enables you to change the database during login.

Note: Do not use DBLibAllowChangeDatabase in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DBLibAllowChangeDatabase = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibApiSeverity (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib only. The DBLibApiSeverity (Property) enables you to change the API error level severity.

Note: Do not use DBLibApiSeverity in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Long

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DBLibApiSeverity = 2
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in  a datamodel
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibDatabaseCancel (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib only. The DBLibDatabaseCancel (Property) enables you to change the Database cancel options.

Note: Do not use DBLibApiDatabaseCancel in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write

Constants:

The BqDbLibCancelMode constant group consists of these values:

- bqDbLibCancel
- bqDbLibLogoff
- bqDbLibPrompt

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DBLibDatabaseCancel = bqDbLibPrompt
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibPacketSize (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib only. The DBLibPacketSize (Property) enables you to change the packet size of the query.

Note: Do not use DBLibPacketSize in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Numeric

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DBLibPacketSize = 200
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibServerSeverity (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib Only. The DBLibServerSeverity (Property) enables you to changes the error level severity on the server.

Note: Do not use DBLibServerSeverity in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Numeric

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.DBLibServerSeverity = 2
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibUseQuotedIdentifiers (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib Only. The DBLibUseQuotedIdentifiers (Property) enables you to use of quoted indentures when connecting by using DB-Lib.

Note: Do not use DBLibUseQuotedIdentifiers in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to create a connection (OCE) using JavaScript:

```
Var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
MyCon.HostName ="PlutoSQLSVR"
MyCon.DBLibUseQuotedIdentifiers = true
MyCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DBLibUseSQLTable (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

DB-Lib Only. The DBLibUseSQLTable (Property) enables the connection to use Structured Query Language (SQL) to get tables.

Note: Do not use DBLibUseSQLTable inInteractive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to create a connection (OCE) that uses SQL to get tables:

```
var myCon
myCon = Application.CreateConnection()
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
MyCon.HostName ="PlutoSQLSVR"
MyCon.DBLibUseSQLTable = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

DecimalPlaces (Property)

Applies To:

DBSpecific (Oracle Essbase) object

Description:

Sets the number of decimal places that the server return. The DecimalPlaces (Property) supports only numeric values from zero through five (0-5).

Action:

Read-write, Number

Example:

This example shows how to set the number of decimal places to three:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.DecimalPlaces=3
```

DefaultDrillOperation (Property)

Applies To:

QueryOptions object

Description:

Sets the default drill method when you drill on an item in the CubeQuery section.

Actions:

Read-write, enumerator

Constants:

BqOLAPDrill, which has the following constant values:

- bqOLAPDrillALLDescendents = 6
- bqOLAPDrillBottom = 5
- bqOLAPDrillDefault = 1
- bqOLAPDrillDown = 3
- bqOLAPDrilNext = 4
- bqOLAPDrillNone = 0
- bqOLAPDrillSameGeneration = 9
- bqOLAPDrillSameLevel = 8
- bqOLAPDrillSibling = 7

- bqOLAPDrillUp = 2

Example

This example shows to how to retrieve the selected drill option:

```
//Get Selected Drill Option
var intSelDrillOption=
ActiveDocument.Sections["QueryOptions"].Shapes["DropDown1"].SelectedIndex
var intbqOLAPDrill

switch (intSelDrillOption)
{
    case 1:     intbqOLAPDrill = 3;    //Next
        break;
    case 2:    intbqOLAPDrill = 4;    //Bottom
        break;
    case 3:    intbqOLAPDrill = 5;    //All Descendants
        break;
    case 4:    intbqOLAPDrill = 6;    //Sibling
        break;
    case 5:    intbqOLAPDrill =7;    //Same Level
        break;
    case 6:    intbqOLAPDrill = 8;    //Same Generation
        break;
    default :     intbqOLAPDrill =
ActiveDocument.Sections["Query"].QueryOptions.DefaultDrillOperation
        break;
}

ActiveDocument.Sections["Query"].QueryOptions.DefaultDrillOperation = intbqOLAPDrill
```

DefaultSortOrderLang (Property)

Applies To:

Application object

Description:

Sets the default sort order language format that is applied to an Interactive Reporting document when it is opened in a Unicode enabled application. If no default language is set (the check box is not enabled) in the user interface, this property returns 0, which does not correspond to any valid language. The sort order language of a Interactive Reporting document may be changed to start sorting the data according to the user interface language preferences. The Application .ini file stores this property in the DefDocLanguage property of [Regional Settings]. It is interchangeable between all applications including Services.

Action:

Read/Write

Example:

This example shows how to read and write the value of a default sort order language in an Interactive Reporting document.

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name
var ActionName = "DefaultSortOrderLang"
TextBox1.Text ="Start " + ActionName
if (TextBox5.Text == "")
{
TextBox1.Text ="Step 1"
try
{
TextBox3.Text =  "DefaultSortOrderLang is: " + Application.DefaultSortOrderLang
switch(Application.DefaultSortOrderLang)
{
case 0:
TextBox3.Text = "No Language specified"
break;
case 1:
TextBox3.Text = "Language is bqLangAfrikaans"
break;
case 2:
TextBox3.Text = "Language is bqLangAlbanian"
break;
case 3:
TextBox3.Text = "Language is bqLangArabic"
break;
case 4:
TextBox3.Text = "Language is bqLangBasque"
break;
case 5:
TextBox3.Text = "Language is bqLangCatalan"
break;
case 6:
TextBox3.Text = "Language is bqLangChinese_Simp"
break;
case 7:
TextBox3.Text = "bqLangChinese_Trad"
break;
case 8:
TextBox3.Text = "Language is bqLangChinese_GB2312_Sort"
break;
case 9:
TextBox3.Text = "Language is bqLangChinese_Trad"
break;
case 10:
TextBox3.Text = "Language is bqLangCroatian"
break;
case 11:
TextBox3.Text = "Language is bqLangCzech"
break;
case 12:
TextBox3.Text = "Language is bqLangDanish"
```

```
break;
case 13:
TextBox3.Text = "Language is bqLangDutch"
break;
case 14:
TextBox3.Text = "Language is bqLangEnglish"
break;
case 15:
TextBox3.Text = "Language is bqLangUKEnglish"
break;
case 16:
TextBox3.Text = "Language is bqLangEstonian"
break;
case 17:
TextBox3.Text = "Language is bqLangFinnish"
break;
case 18:
TextBox3.Text = "Language is bqLangFrench"
break;
case 19:
TextBox3.Text = "Language is bqLangGerman"
break;
case 20:
TextBox3.Text = "Language is bqLangGerman_Phonebook_Sort"
break;
case 21:
TextBox3.Text = "Language is bqLangGreek"
break;
case 22:
TextBox3.Text = "Language is bqLangHebrew"
break;
case 23:
TextBox3.Text = "Language is bqLangHungarian"
break;
case 24:
TextBox3.Text = "Language is bqLangIcelandic"
break;
case 25:
TextBox3.Text = "Language is bqLangIndonesian"
break;
case 26:
TextBox3.Text = "Language is bqLangItalian"
break;
case 27:
TextBox3.Text = "Language is bqLangJapanese"
break;
case 28:
TextBox3.Text = "Language is bqLangKorean"
break;
case 29:
TextBox3.Text = "Language is bqLangLatvian"
break;
case 30:
TextBox3.Text = "Language is bqLangLithuanian"
break;
case 31:
TextBox3.Text = "Language is bqLangMacedonian"
```

```
break;
case 32:
TextBox3.Text = "Language is bqLangMalay"
break;
case 33:
TextBox3.Text = "Language is bqLangNorwegian_Bokmal"
break;
case 34:
TextBox3.Text = "Language is bqLangNorwegian_Nynorsk"
break;
case 35:
TextBox3.Text = "Language is bqLangPolish"
break;
case 36:
TextBox3.Text = "Language is bqLangPortuguese"
break;
case 37:
TextBox3.Text = "Language is bqLangRomanian"
break;
case 38:
TextBox3.Text = "Language is bqLangRussian"
break;
case 39:
TextBox3.Text = "Language is bqLangSerbian_Cyrillic"
break;
case 40:
TextBox3.Text = "Language is bqLangSerbian_Latin"
break;
case 41:
TextBox3.Text = "Language is bqLangSlovak"
break;
case 42:
TextBox3.Text = "Language is bqLangSlovenian"
break;
case 43:
TextBox3.Text = "Language is bqLangSpanish"
break;
case 44:
TextBox3.Text = "Language is bqLangSpanish_Trad_Sort"
break;
case 45:
TextBox3.Text = "Language is bqLangSwedish"
break;
case 46:
TextBox3.Text = "Language is bqLangThai"
break;
case 47:
TextBox3.Text = "Language is bqLangTurkish"
break;
case 48:
TextBox3.Text = "Language is bqLangUkrainian"
break;
case 49:
TextBox3.Text = "Language is bqLangVietnamese"
break;
default:
TextBox3.Text = "Language Value is Not Available"
```

```
}
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
}
else
{
TextBox1.Text ="Step 2"
try
{
Application.DefaultSortOrderLang = eval(TextBox5.Text)
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
TextBox1.Text ="Step 3"
try
{
TextBox3.Text =  "DefaultSortOrderLang is: " + Application.DefaultSortOrderLang
switch(Application.DefaultSortOrderLang)
{
case 0:
TextBox3.Text = "No Language specified"
break;
case 1:
TextBox3.Text = "Language is bqLangAfrikaans"
break;
case 2:
TextBox3.Text = "Language is bqLangAlbanian"
break;
case 3:
TextBox3.Text = "Language is bqLangArabic"
break;
case 4:
TextBox3.Text = "Language is bqLangBasque"
break;
case 5:
TextBox3.Text = "Language is bqLangCatalan"
break;
case 6:
TextBox3.Text = "Language is bqLangChinese_Simp"
break;
case 7:
TextBox3.Text = "bqLangChinese_Trad"
break;
case 8:
TextBox3.Text = "Language is bqLangChinese_GB2312_Sort"
break;
case 9:
TextBox3.Text = "Language is bqLangChinese_Trad"
break;
case 10:
TextBox3.Text = "Language is bqLangCroatian"
break;
```

```
case 11:
TextBox3.Text = "Language is bqLangCzech"
break;
case 12:
TextBox3.Text = "Language is bqLangDanish"
break;
case 13:
TextBox3.Text = "Language is bqLangDutch"
break;
case 14:
TextBox3.Text = "Language is bqLangEnglish"
break;
case 15:
TextBox3.Text = "Language is bqLangUKEnglish"
break;
case 16:
TextBox3.Text = "Language is bqLangEstonian"
break;
case 17:
TextBox3.Text = "Language is bqLangFinnish"
break;
case 18:
TextBox3.Text = "Language is bqLangFrench"
break;
case 19:
TextBox3.Text = "Language is bqLangGerman"
break;
case 20:
TextBox3.Text = "Language is bqLangGerman_Phonebook_Sort"
break;
case 21:
TextBox3.Text = "Language is bqLangGreek"
break;
case 22:
TextBox3.Text = "Language is bqLangHebrew"
break;
case 23:
TextBox3.Text = "Language is bqLangHungarian"
break;
case 24:
TextBox3.Text = "Language is bqLangIcelandic"
break;
case 25:
TextBox3.Text = "Language is bqLangIndonesian"
break;
case 26:
TextBox3.Text = "Language is bqLangItalian"
break;
case 27:
TextBox3.Text = "Language is bqLangJapanese"
break;
case 28:
TextBox3.Text = "Language is bqLangKorean"
break;
case 29:
TextBox3.Text = "Language is bqLangLatvian"
break;
```

```
case 30:
TextBox3.Text = "Language is bqLangLithuanian"
break;
case 31:
TextBox3.Text = "Language is bqLangMacedonian"
break;
case 32:
TextBox3.Text = "Language is bqLangMalay"
break;
case 33:
TextBox3.Text = "Language is bqLangNorwegian_Bokmal"
break;
case 34:
TextBox3.Text = "Language is bqLangNorwegian_Nynorsk"
break;
case 35:
TextBox3.Text = "Language is bqLangPolish"
break;
case 36:
TextBox3.Text = "Language is bqLangPortuguese"
break;
case 37:
TextBox3.Text = "Language is bqLangRomanian"
break;
case 38:
TextBox3.Text = "Language is bqLangRussian"
break;
case 39:
TextBox3.Text = "Language is bqLangSerbian_Cyrillic"
break;
case 40:
TextBox3.Text = "Language is bqLangSerbian_Latin"
break;
case 41:
TextBox3.Text = "Language is bqLangSlovak"
break;
case 42:
TextBox3.Text = "Language is bqLangSlovenian"
break;
case 43:
TextBox3.Text = "Language is bqLangSpanish"
break;
case 44:
TextBox3.Text = "Language is bqLangSpanish_Trad_Sort"
break;
case 45:
TextBox3.Text = "Language is bqLangSwedish"
break;
case 46:
TextBox3.Text = "Language is bqLangThai"
break;
case 47:
TextBox3.Text = "Language is bqLangTurkish"
break;
case 48:
TextBox3.Text = "Language is bqLangUkrainian"
break;
```

```
case 49:
TextBox3.Text = "Language is bqLangVietnamese"
break;
default:
TextBox3.Text = "Language Value is Not Available"
}
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
}
TextBox1.Text ="End " + ActionName
```

Description (Property)

Applies To:

Connection object

Description:

Returns or sets the description associated with an Interactive Reporting database connection file (.oce).

Action:

Read-write, String

Example:

This example creates a connection file and apply it to the current document:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

Dimension (Property)

Applies To:

OLAPFilter object

Description:

Returns the filter to which the filter has been applied.

Action:

Read only, String

Example:

This example shows how to return the dimension to which the filter has been applied.

```
if (Sections["Query"].Filters[1].Dimension == "Product") {
    // There is a filter on the Product dimension
}
```

DisableSelection (Property)

Applies To:

OLAPMemberSelector object, OLAPFilter object

Description:

In CubeQuery, specifies whether a member selector should be ignored or not.

Action:

Read-write, Boolean

Example:

This example shows how to disable a member selector in the CubeQuery section:

```
ActiveDocument.Sections["Query"].Rows["Market"].MemberSelectors["West"].DisableSelection
```

Display (Property)

Applies To:

CornerLabels object, DataLabels object

Description:

Returns the display value of a corner or data label. The Display (Property) uses the BqPivotLabelDisplay constant. Valid options for displaying the label are side, top, both or none. The default corner label value is none.

Action:

Read-write, String

Constants:

The BqPivotLabelDisplay constant group consists of these values:

- BqPivotLabelDisplayBoth
- BqPivotLabelDisplayNone
- BqPivotLabelDisplaySide
- BqPivotLabelDisplayTop

Example:

This example shows how to return a corner label at the top of the Pivot report:

```
ActiveDocument.Sections["SalesPivot"].CornerLabels. Display=bqPivotLabelDisplayBoth
```

DisplayAllTextWithSmartScaling (Property)

Applies To:

ChartSection object

Description:

If the "SmartScaling (Property)" on page 208is enabled, use this property to prevent text from being removed if an object is resized or repositioned to a minimum file sized. In the user interface, this property is enabled for new Chart sections, and disabled for 8.3 releases and earlier that have Smart Scaling enabled. Inserted text and legends are not affected by this property. Smart Scaling may shrink these objects, but they are never scaled to the point where they are eliminated entirely.

Action:

Read-write, Boolean

Example:

This example shows how to retain text when Smart Scaling is enabled.

```
ActiveDocument.Sections["Chart"].DisplayAllTextWithSmartScaling = true
```

DisplayMode (Property)

Applies To:

HyperLink object

Description:

Enables you to select where to display the contents of the hyperlink control from the Display In drop-down list.

Valid options are:

- **New Window** — A new pop-up window is created every time the hyperlink is pressed. This is the default setting. All browser menus and toolbars are hidden. This maps to the OpenURL (Method) "_blank" target.
- **Current Window** — If the execution application is Interactive Reporting Web Client, or EPM Workspace, then the content replaces the entire EPM Workspace content area (i.e. excluding surrounding browse application frames). In Designer a pop-up window displays.
- **Top Window**— If the execution application is theInteractive Reporting Web Client , or EPM Workspace, the content replaces the top-most HTML window. This maps to the OpenURL (Method) "_top" target.
- **Named Window** — A "named" pop-up window is created. If the named window is already present, then the URL replaces the existing content of the named window. When this option is selected, a Target window edit control displays below the dropdown, in which the user can enter the desired window name. This maps to the OpenURL (Method) user specified target name option.
- **No Window**— No window is displayed. This option may be used in cases where the URL causes an action (such as executing a job) that does not require any content to be displayed. Returns the display value of a corner or data label. The Display property uses the BqPivotLabelDisplay constant group. Valid options for displaying the label are side, top, both or none. The default corner label value is none.

Action:

Read only, BqOpenURLTarget constant group

Constants:

The BqOpenURLTarget constant group consists of these values:

bqOpenURLTargetBlank = 0

bqOpenURLTargetCustom =1

bqOpenURLTargetNone = 2

bqOpenURLTargetSelf = 3

bqOpenURLTargetTop =4

Example:

This example shows how to return the Display Mode setting for a hyperlink object in an Alert box:

```
Alert(HyperLink.DisplayMode=bqOpenURLTargetTop)
```

DisplayName (Property)

Applies To:

DerivedItem object, Limit object, Requests collection, TopicItem object

Description:

Returns or sets the display name of the DerivedItem (Object), Limit (Object), Requests (Collection), and TopicItem (Object). The DisplayName (Property) is the name of the object as viewed by the user when working with the Interactive Reporting product suite.

There are three "name" type properties used in the Object Model, including the:

- DisplayName (Property)
- "PhysicalName (Property)" on page 155
- "Name (Property)" on page 143

To determine which name property to use in Object Models, consider this example.

A database table is entitled MyColumn. The *PhysicalName* of the column is MyColumn. To display another name for MyColumn, such as MySpecialColumn, use the *DisplayName* (Property). This enables you to modify the column name shown in Interactive Reporting but retain the original name at the source. Use the *Name* (Property) to reference objects in the Object Model. For example, to determine the name of a TextBox object in Dashboards, use this script:

```
TextBox1.Name
```

Action:

Read-write, String

Example:

This example writes the names of all the topics and topic items to the Console window:

```
var Tcount = ActiveDocument.Sections["Query"].DataModel.Topics.Count
for (j = 1; j <= Tcount ; j ++)
{
var myTopic = ActiveDocument.Sections["Query"].DataModel.Topics[j]
Console.Writeln("Topic : "+myTopic.PhysicalName)
var TICount = ActiveDocument.Sections["Query"].DataModel.Topics[j].TopicItems.Count
    for (k = 1 ; k <= TICount ; k ++)
    {
        var myItem =     ActiveDocument.Sections["Query"].DataModel.
                    Topics[j].TopicItems[k]
        Console.Writeln(" Item: "+ myItem.DisplayName)
    }
}
```

DrawingOrder (Property)

Applies To:

TrendLine object, ReferenceLine object

Description:

Sets how trend lines are positioned relative to other chart objects. Trend lines can be layered on top of the chart graphics (or Z axis for 3 D charts), or positioned to the background.

Determines reference line drawing order relative to other chart objects

Action:

Read-write, BqDrawingOrder

Constants:

The DrawingOrder (Property) uses the BqDrawingOrder constant group, which consists of the following constants:

- bqDrawingOrderToBack = 2
- bqDrawingOrderToFront = 1
- bqDrawingOrderUndefined =0

Example:

This example shows how to position the trend line to the front of the other chart elements:

```
ActiveDocument.Sections["Chart6"].TrendLines[1].DrawingOrder = 1
```

DrillDownDisplay (Property)

Applies To:

OLAPQuery object

Description:

Enables you to define what level of data is the next level displayed when you drill down in an OLAPQuery. Available drill-down options include:

- **Drill to Next Level**— Automatically displays data for the next level below the selected member. For example, in a dimension with levels of Year, Quarter, Month, and Date, double-clicking on a Year level member name automatically displays all the data for the Quarter level belonging to that year.
- **Drill to All Levels**— Automatically displays all possible levels of data below the selected member. For example, in a dimension with levels of Year, Quarter, Month, and Date, double-

clicking on a Year level member name automatically displays all the data for the Quarter, Month, and Date levels belonging to that year.

- **Drill to Lowest Level**— Automatically displays data for only the lowest level belonging to the selected member (intermediate member levels are not shown). For example, in a dimension with levels of Year, Quarter, Month, and Date, double-clicking on a Year level member name automatically displays all the data for the Date level belonging to that year.

Action:

Read-write, BqDrillDownDisplay

Constants:

The BqDrillDownDisplay constant group consists of these values:

- bqDrillDownDisplayAllLevels
- bqDrillDownDisplayLowestLevel
- bqDrillDownDisplayNextLevel

Example:

This example writes the names of all the topics and topic items to the Console window:

```
ActiveDocument.Sections["OLAPQuery"].DrillDownDisplay = bqDrillDownDisplayAllLevels
```

Effect (Property)

Applies To:

Font object, (Live) BarChart object, (Live) BlockChart object, (Live) FunnelChart object (Live) LineChart object, (Live) PieChart object, (Live) RadarChart object

Description:

For a font object, returns or sets the font effect of text associated with a font object.

For (Live) LineChart, or (Live) PieChart objects, returns or sets a 2D, 3D, or gradient effect. A 2D or 3D effect refers to the perspective of the rendered chart. A gradient defines the direction of the blend pattern from one color to another, and simulates depths of color from light to dark. Set the effect by using the BQLiveChartEffect (only the bqLiveChartEffect2D and bqLiveChartEffect3D constant values apply to the (Live) PieChart object.

For a (Live) LineChart objects, returns or sets a 2D and horizontal or vertical gradient effect.

For (Live) BlockChart and (Live) FunnelChart objects, returns or sets a 2D or 3D effect.

For a (Live) RadarChart, returns or sets an outlined or filled effect.

Action:

Read-write, BqFontEffect (Font object), BqLiveChartEffect (Live Chart objects)

Constants:

The BqFontEffect constant group consists of these values:

- bqFontEffectNone
- bqFontEffectOverDouble (not supported in EPM Workspace)
- bqFontEffectOverline
- bqFontEffectStrikeThrough (not supported in EPM Workspace)
- bqFontEffectSubScript (not supported in EPM Workspace)
- bqFontEffectSuperScript (not supported in EPM Workspace)
- bqFontEffectUnderline

  The BqLiveChartEffect constant group consists of these values:

  - bqLiveChartEffect2D
  - bqLiveChartEffect3D
  - bqLiveChartEffectFill
  - bqLiveChartEffectGradient_H
  - bqLiveChartEffectGradient_V
  - bqLiveChartEffectNone
  - bqLiveChartEffectOutline

Example:

This example changes the font effect of the text in a text label named *Description*:

```
ActiveDocument.Sections["Dashboard2"].Shapes["Description"].Font.Effect=
bqFontEffectUnderline
```

EnableAsyncProcess (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables asynchronous processing of a query associated with the connection object.

Action:

Read-write, Boolean

Example:

This example creates a connection file and applies it to the current document:

```
var myCon = Application.CreateConnection()
```

```
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

Enabled (Property)

Applies To:

Control object, ControlsCheckBox object, ControlsCommandButton object, ControlsDropDown object, ControlsListBox object, ControlsRadioButton object, ControlsTextBox object, EmbeddedBrowser object

Description:

Returns or sets the current state of a control object. If a control is disabled, you cannot access it using the Dashboard section. The control is shown in the Dashboard section in a grayed out or disabled state.

Action:

Read-write, Boolean

Example:

This examples enables every shape and control object in an Dashboard section named Dashboard:

```
var DashboardSection = ActiveDocument.Sections["Dashboard"]
var ShapeCount = DashboardSection.Shapes.Count
for (j=1;j <= ShapeCount ;j++)
{
DashboardSection.Shapes[j].Enable = true
}
```

EnableForHybridAnalysis (Property)

Applies To:

OLAPQuerySection object (DBSpecific)

Description:

Enables Hybrid Analysis value retrieval. Hybrid Analysis enables the lowest levels of an Essbase cube to reside in an external relational database. These levels are not reflected in the cube

structure (metadata) that reside on the Essbase Server. Instead, their existence and retrieval is performed by the separate Essbase Integration Server product. This option is only available for connections to Essbase 6.5+ or IBM DB2 OLAP 8.1+ databases.

Action:

Read-write, Boolean

Example:

This example shows how to enable Hybrid Analysis:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.EnableForHybridAnalysis = true
```

EnableNullFactsInComputedItems (Property)

Applies To:

Pivot Section object

Description:

Enables Interactive Reporting to evaluate a null fact value (an empty cell value) as a zero fact value for non-Moving functions. For Moving Function calculations, where the presence of all displayed Fact cell must be considered in calculations, the number of instances of null values are subtracted from the *Window* divisor.

Exponential Moving Averages treat missing source fact values as having a value of zero.

When this option is disabled, the Pivot Sections that receive their data from a relational Query Section, ignore fact cells with null data when calculating Computed Items. That is, the calculation is simply skipped for the null cell.

Action:

Read-write, Boolean

Example:

This examples shows how to enable null fact values for computed items:

```
ActiveDocument.Sections["Pivot"].EnableNullFactsInComputedItems = true
```

EnableTransActionMode (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables transaction mode for the Interactive Reporting database connection file or current connection.

Action:

Read-write, Boolean

Example:

This example creates an Interactive Reporting connection file and applies it to the current document:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
MyCon.EnableTransAction = true
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

EndLimitName (Property)

Applies To:

Parentheses object

Description:

When the Parentheses (Collections) is invoked, the EndLimitName (Property) sets the limit value and inserts the ending (closing) parentheses. This property is often used with the "BeginLimitName (Property)" on page 43.

Action:

Read-only, EndLimitName as String

Example:

This example shows how to display the name of the ending limit value enclosed in a parenthetical expression on the Limit Line:

```
Alert(ActiveDocument.Sections["Query"].Limits.Parentheses["State
Province,City"].EndLimitName)
```

ExecuteOnPostProcess (Property)

Applies To:

DesktopClient object, PlugInClient object, ThinClient object and Scheduler object.

Description:

Enables the execution of an OnStartup document event for the DesktopClient (Object) (Interactive Reporting), PlugInClient (Object) (web client), ThinClient (Object) (Hyperion System 9 BI + Workspace) and Scheduler (Object) in the 8.x client environment. The OnPostProcess document event is executed after the query is processed.

Action:

Read-write, Boolean

Example:

This example shows how to disable the OnPostProcess document events when running in the web client environment:

```
Documents["Sample1.bqy"].Events["PlugInClient"].ExecuteOnPostProcess=false
```

ExecuteOnPreProcess (Property)

Applies To:

DesktopClient object, PlugInClient object, ThinClient object and Scheduler object.

Description:

Enables the execution of an OnStartup document event for the DesktopClient (Object) (Interactive Reporting, PlugInClient (Object) (web client), ThinClient (Object) (EPM Workspace) and Scheduler (Object) in the 8.0 client environment. The OnStartup document event is executed immediately before query processing. To discontinue processing for a defined condition in the PreProcess() event, see the InterruptQueryProcess (Method).

Action:

Read-write, Boolean

Example:

This example shows how to enable the OnPreProcess document events when running in the Scheduler environment:

```
Documents["Sample1.bqy"].Events["Scheduler"].ExecuteOnPreProcess=false
```

ExecuteOnShutDown (Property)

Applies To:

DesktopClient object, PlugInClient object, ThinClient object and Scheduler object.

Description:

Enables the execution of an OnStartup document event for the DesktopClient (Object) (Interactive Reporting Studio), PlugInClient (Object) (Interactive Reporting Web Client), ThinClient (Object) (EPM Workspace) and Scheduler (Object) in the 8.0 client environment. The OnShutDown document event is executed when the document is closed.

Action:

Read-write, Boolean

Example:

This example shows how to disable the OnShutDown document event when running in the EPM Workspaceenvironment:

```
Documents["Sample1.bqy"].Events["ThinClient"].ExecuteOnShutDown=false
```

ExecuteOnStartup (Property)

Applies To:

DesktopClient object, PlugInClient object, ThinClient object and Scheduler object.

Description:

Enables the execution of an OnStartup document event for the DesktopClient (Object) (Interactive Reporting Studio), PlugInClient (Object) (Interactive Reporting Web Client), ThinClient (Object) (EPM Workspace) and Scheduler (Object) in the 8.0 client environment. The OnStartup document event is executed when the document is opened.

Action:

Read-write, Boolean

Example:

This example shows how to enable an OnStartup document events when running in the Interactive Reporting environment:

```
Documents["Sample1.bqy"].Events["DesktopClient"].ExecuteOnStartUp=true
```

ExpandLabelBox (Property)

Applies To:

XAxislLabel object

Description:

Increases the label box size for an X category label (Z and Y category labels are unaffected) in a Chart. It is only active when the "ShowLabel (Property)" on page 195is enabled and the "AutoFrequency (Property)" on page 32 is disabled. It is unavailable for Pie Charts. This property is used primarily for sequential labels, for example, time, day or number labels. For regular name-based labels, this property should be disabled so that the user can view each label name.

Action:

Read-write, Boolean

Example:

This example shows how to enable the ExpandLabelBox property:

```
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.ExpandLabelBox=true
```

Explode (Property)

Applies To:

(Live) PieChart object

Description:

Returns or sets the property to display the relationship of each data value to a whole and extract individual values in a (Live) Pie Chart.

Action:

Read-write, Boolean

Example:

This example shows to explode slices in a (Live) Pie Chart:

```
PieChart.Explode = true
```

ExportWithoutQuotes (Property)

Applies To:

OLAPQuerySection, PivotSection object, ResultsSection object, TableSection object

Description:

When exporting section data, the ExportWithoutQuotes (Property) enables double quotes surrounding column/cell values containing real values. The default value is disabled.

Action:

Read-write, Boolean

Example 1:

This example exports a Results set to a tab delimited text file that retains double quotes surrounding the Results column data:

```
ActiveDocument.Sections["Results"].ExportWithoutQuotes=false
ActiveDocument.Sections["Results"].Export("C:\Temp\ExportTest\ MyFile",
bqExportFormatText)
```

Example 2:

This example exports Results set to a tab delimited text file without double quotes surrounding the Results column data:

```
ActiveDocument.Sections["Results"].ExportWithoutQuotes=true
ActiveDocument.Sections["Results"].Export("C:\Temp\ExportTest\ MyFile",
bqExportFormatText)
```

FactIndex (Property)

Applies To:

TrendLine object, ReferenceLine object

Description:

Index of facts in the Fact collection for which the trend line is built. A trend line added for the sum of all facts has an index value of zero. This property is the equivalent to the Fact dropdown which contains the list of facts available for assigning to the trend line. In current version it is not possible to bind several trend lines to a single fact.

Action:

Read-write

Example:

This example shows how to display the fact index numbers for two trend lines in a Line chart:

```
Alert(ActiveDocument.Sections["Chart6"].TrendLines[1].FactIndex)
Alert(ActiveDocument.Sections["Chart6"].TrendLines[2].FactIndex)
```

FactName (Property)

Applies To:

Embedded Pivot Objects

Description:

Returns the name of the fact to which the value applies. For example, if the value in CellValue corresponds to the name “Amount Sales” in the Facts section of the Pivot Outliner, then FactName is set to “Amount Sales”. When no Fact exists and the property value is retrieved, the property return an empty string.

The FactName property is accessible for the Dashboard Pivot Embedded Sections Objects (ESOs). This property is only available when a Dashboard section containing a Pivot ESO is included in the BI document. It is only functional when the Pivot is in Active mode.

The FactName Property is read-only. Value returned are a string representing the unformatted JavaScript value of the cell. Default value of the property is the name of the fact associated with the cell selected. .

The FactName (Property) is available in the Interactive Reporting Studio and Interactive Reporting Web Client. It is unavailable for the EPM Workspace or the Jobs/Scheduler.

Action:

Read only, String

Example:

This example shows how to return the value of the fact that has been double-clicked. It includes a try-catch statements, and a call to the *tostring* value. In this example, “FactName is: Amount” is returned.

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name
var ActionName = "FactName"
TextBox1.Text ="Start " + ActionName
if (TextBox5.Text == "")
{
TextBox1.Text ="Step 1"
try
{
```

```
PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nFactName is: " +
ActiveSection.Shapes["Pivot1"].FactName
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
}
else
{
TextBox1.Text ="Step 2"
try
{
ActiveDocument.Sections["Pivot"].Facts.FactName = TextBox5.Text
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
TextBox1.Text ="Step 3"
try
{
TextBox3.Text = "Facts is: " + ActiveDocument.Sections["Pivot"].Facts.FactName
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
}
TextBox1.Text ="End " + ActionName
```

Filename (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns the full name and path of the Interactive Reporting database connection file (.oce) associated with the connection object.

Action:

Read-only, String

Example:

This example creates an Interactive Reportingdatabase connection file and associates it with the current Interactive Reporting document:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto.".Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
var OCEFilename = ActiveDocument.Sections["Query"].DataModel.Connection.Filename
Console.Write ("Successfully opened the OCE named: "+OCEFilename)
```

FilePath (Property)

Applies To:

Picture object

Description:

Sets the file name of a picture object.

Note: You can change the FilePath (Property), but this change has no effect on the picture that is displayed. This is because the picture is no longer referenced from the file system, and it is embedded in theInteractive Reporting document file (.bqy) .

Action:

Read-write, Name

Example:

This example shows how to set the file path name for the picture entitled Report:

```
ActiveDocument.Sections["Report"].Body.Shapes["Picture"].FilePath = "c:\\hyperion\
\report.bmp"
```

FillUnderRibbon (Property)

Applies To:

Area Chart

Description:

Enables the area under the ribbon to be filled in an Area Chart.

Action:

Read-write, Boolean

Example:

This example enables the FillUnderRibbion attribute of an Area Chart for the section named *Sales Chart*:

```
var MyChart = ActiveDocument.Sections["Sales Chart"]
MyChart.AreaChart.FillUnderRibbon = true
```

Focus (Property)

Applies To:

Legend Collection

Description:

Returns or sets the focus of the legend on a selected chart axis type (X-axis, Y-axis, or Z axis.

Action:

Read-only, BqChartAxisType constant

Constants:

The BqChartAxisType constant group consists of these values:

- BqChartXAxis
- BqChartYAxis
- BqXhartZAxis

Example:

This example shows how to change the Chart axis type to the X-axis category:

```
ActiveDocument.Sections["Chart"].Legend.Focus=bqChartXAxis
```

Formula (Property)

Applies To:

Fields collection

Description:

Sets a computable value for a Field item in the Report section. This property is analogous to editing or entering a formula for a selected field in the Expression bar. The value specified instructs the application from where to retrieve information.

Action:

Read-write, String

Example:

This example shows how to change the chart axis type to the X-axis category:

```
ActiveDocument.Sections["Sales Report"].ReportHeader.Fields["ReportName Field"].Formula
= "ReportName() + '    '  + new Date()"
```

FullName (Property)

Applies To:

Limit object

Description:

Returns or sets the value of limits full name. The full name may include the topic. (Query and Data Model Limits only)

Action:

Read-write, String

Example:

This example prints out the full names of all the limits in a query section named *SalesQuery*:

```
var MyQuery = ActiveDocument.Sections["SalesQuery"]
var LimitCount = MyQuery.Limits.Count
for (j =1 ; j <= LimitCount ; j++)
    Console.Writeln("Limit fullname is " + MyQuery.Limits[j].FullName)
```

Function (Property)

Applies To:

Reference Line object

Description:

Sets the data function applied to the reference line fact value.

Action:

BqDataFunction (Read only) constant group, which consists of the following values:

- bqDataFunctionAverage
- bqDataFunctionColumnName
- bqDataFunctionCount
- bqDataFunctionCountDistinct
- bqDataFunctionIncrease (Pivot Totals properties, not Facts)
- bqDataFunctionMaximum
- bqDataFunctionMinimum
- bqDataFunctionNone
- bqDataFunctionNonNullAverage
- bqDataFunctionNonNullCount
- bqDataFunctionPercentOfCategory
- bqDataFunctionPercentOfColumn
- bqDataFunctionPercentOfGrand
- bqDataFunctionPercentOfIncrease
- bqDataFunctionPercentOfRow
- bqDataFunctionSum

Example:

This example shows how to read the data function value applied to the fact item on the reference line, and display it in the Console window.

```
Console.Write(ActiveDocument.Sections["Chart8"].ReferenceLines[1].Function)
```

GraphicsFileType (Property)

Applies To:

Picture object

Description:

Sets the graphic file type of a picture graphics object.

Action:

Read-write, BqGraphicsFileType

Constants:

The GraphicsFileType property uses the BqGraphicsFileType constant group, which consists of these values:

- bqGraphicsFileTypeBMP
- bqGraphicsFileTypeGIF
- bqGraphicsFileTypeJPEG

Example 1:

This example shows how to set the GraphicFileType (Property) to a .BMP format in the Dashboard section:

```
ActiveDocument.Sections["Dashboard2"].Shapes["Picture2"].GrapicsFileType=bqGraphicsFileTypeBMP
```

Example 2:

This example shows how to set the GraphicFileType (Property) to a .JPEG format in the Report section:

```
ActiveDocument.Sections["Report"].Body.Shapes["Picture"].GraphicsFileType=bqGraphicsFileTypeJPEG
```

Group (Property)

Applies To:

ControlsRadioButton object

Description:

Enables you to join together two or more Radio buttons in a group.

Action:

Read-only, String

Example:

This example shows how to assign a group name to radio buttons:

```
RadioButton1.Group="Sales"
RadioButton2.Group="Sales"
```

```
RadioButton3.Group="Sales"
```

HardwireMode (Property)

Applies To:

OLAPQuerySection object

Description:

Enables the automatic processing of an OLAP query every time there is a change to it. Hardwire mode allows the instantaneous retrieval of cube slice and eliminates having to manually click Process when a change (an item is added or removed from the Outliner) is made.

Before using hardwire or process mode, determine if you will drag items to the Outliner before initiating requests. To drag multiple items to the Outliner, use Process mode.

Action:

Read-write, Boolean

Example:

This example shows to activate hardwire mode:

```
ActiveDocument.Sections["OLAPQuery"].HardwireMode = true
```

Height (Property)

Applies To:

PieChart object, Shapes collection (Placement node), Image object

Description:

For a Pie chart object, returns or sets the height property.

When used for placing objects, the Height (Property) sets or returns the height of a shape in pixels.

For an Image object, this property determines the height in pixels of the specific image in the Resource Manager. The property is read-only, and the value returned is a non-string number between zero and the maximum height of the image. The default value of the property is the maximum height of the image.

Action:

Read-write for Pie chart and Shape objects, **Read only** for Image object. Numeric

Example 1:

This example shows how to change the height of a pie chart in the chart section named *Sales Pie Chart*:

```
var MyChart = ActiveDocument.Sections["Pie Chart"]
MyChart.PieChart.Height = 10
```

Example 2:

This example shows how to determine the height of an image and to show the results in a textbox:

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name

var ActionName = "RM Height"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{

TextBox1.Text ="Step 1"

try
{
TextBox3.Text =  "Height is: " +
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Height
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}
else
{
TextBox1.Text ="Step 2"

try
{
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Height =
TextBox5.Text
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
{
TextBox3.Text =  "Height is: " +
```

```
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Height
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

TextBox1.Text ="End " + ActionName
```

HomeDashboard (Property)

Applies To:

ActiveDocument object

Description:

Sets the default Dashboard Home section at publishing time, which instructs the Interactive Reporting Service to open a Dashboard section with at least one embedded browser object instead of the last opened section in the Interactive Reporting document file (.bqy). The HomeDashboard (Property) is equivalent to selecting a section on the Dashboard Home dialog. If the Dashboard Home section has been selected in the user interface, it overrides any selection made in the HomeDashboard (Property).

You must use this property with the PreloadHomeSection (Property), which pre-loades the active Dashboard home section to increase loading performance in the EPM Workspace. If the HomeDashboard (Property) is not used with preloadHomeSection (), the default Dashboard section is used.

Note: To enable the Dashboard Home toolbar button when an Interactive Reporting document file is deployed in the EPM Workspace, Dashboard home must be explicitly set using the HomeDashboard (Property) or the Dashboard Home feature in Interactive Reporting Studio or Interactive Reporting Web Client.

Action:

Read-write, String

Example:

This example show how to preload the active Dashboard home section:

```
ActiveDocument.PreloadHomeSection = true
ActiveDocument.HomeDashboard= "Dashboard"
```

HorizontalAlignment (Property)

Applies To:

CellFormat object, TableFacts object

Description:

Returns or sets the horizontal alignment of text in a table column. This property corresponds to the features on the Alignment Properties dialog box.

Action:

Read-write, BqHorizontalAlignment

Constants:

The HorizontalAlignment property uses the BqHorizontalAlignment constant group, which consists of these values:

- bqAlignCenter
- bqAlignLeft
- bqAlignRight

Example:

This example shows how to align left the horizontal text in the *Unit Sales* column:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts["Unit
Sales"].HorizontalAlignment=bqAlignLeft
```

HostName (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the name of the data source.

Action:

Read-write, String

Example 1:

This example creates a connection file and applies it to the current document. The data source name in this example is *PlutoSQLSVR* which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
```

```
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto.".Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

Example 2:

This example shows how to derive the host name for the connection:

```
Console.Writeln(ActiveDocument.Sections["Query"].DataModel.Connection.HostName)
```

HTMLBoundaryHeight (Property)

Applies To:

ChartSection object

Description:

Enables you to fix the height of a Chart boundary. Adjust the height of a Chart boundary to publish/republish the chart section of Interactive Reporting document files (.bqy) files EPM Workspace. The boundary consists of pixilated vertical and horizontal parameters that surround the chart, legend and label.

Note: If you make dynamic HTML changes using the Export Properties dialog or the Object Model, publish or republish the Interactive Reporting document (.bqy) file.

Action:

Read-Write, Number

Example:

This example shows how to enable the boundary and to set both the boundary height and width to 300 pixels:

```
ActiveDocument.Sections["Chart"].HTMLBoundaryMode = true
ActiveDocument.Sections["Chart"].HTMLBoundaryHeight = 300
ActiveDocument.Sections["Chart"].HTMLBoundaryWidth = 300
```

HTMLBoundaryMode (Property)

Applies To:

ChartSection object

Description:

To publish/republish the Chart section of an Interactive Reporting document file(.bqy) file for use in the EPM Workspace, use this property to enable the Chart boundary. The boundary consists of the rectangular parameters measured in pixels which surround the actual chart, legend and label(s).

Note: If you make dynamic HTML changes using the Export Properties dialog or the Object Model, publish or republish the Interactive Reporting document (.bqy) file.

Action:

Read-Write, Boolean

Example:

This example shows how to enable the boundary and to set both the boundary height and width to 300 pixels:

```
ActiveDocument.Sections["Chart"].HTMLBoundaryMode = true
ActiveDocument.Sections["Chart"].HTMLBoundaryHeight = 300
ActiveDocument.Sections["Chart"].HTMLBoundaryWidth = 300
```

HTMLBoundaryWidth (Property)

Applies To:

ChartSection object

Description:

If you publish/republish the Chart section of an Interactive Reporting document (.bqy) file for use in the EPM Workspace, use this property to fix the width of a Chart boundary. The boundary consists of the rectangular parameters measured in pixels which surround the actual chart, legend and label.

Note: If you make dynamic HTML changes using the Export Properties dialog or the Object Model, publish or republish the Interactive Reporting document file (.bqy).

Action:

Read-Write, Number

Example:

This example shows how to enable. the boundary and to set both the boundary height and width to three hundred pixels:

```
ActiveDocument.Sections["Chart"].HTMLBoundaryMode = true
```

```
ActiveDocument.Sections["Chart"].HTMLBoundaryHeight = 300
ActiveDocument.Sections["Chart"].HTMLBoundaryWidth = 300
```

HTMLDisplayViews (Property)

Applies To:

ChartSection object

Description:

Includes the HTMLDisplayXViews and HTMLDisplayZViews properties, and enables multiple views of a HTML rendered charts (static and EPM Workspace). This property is used with the "HTMLMaxBarsDisplayed (Property)" on page 115, to define the number of bars to show per view.

The HTMLDisplayViews (Property) is unavailable for Pie charts.

If the "HTMLSyncScrollingProps (Property) " on page 115 has been set to true, then the "HTMLMaxBarsDisplayed (Property)" on page 115 properties is disabled. The HTMLSyncScollingProp (Property) specifies that the maximum number of bars displayed (X and Z directions) for HTML renderings will match the corresponding values in the Chart Properties dialog.

Action:

Read-write, Boolean

Example:

This example shows how to enable multiple views of a rendered Chart for the X and Y axis labels, and sets the number of bars per view at ten for the X axis label and twelve for the Z axis label:

```
ActiveDocument.Sections["Chart"].HTMLSyncScrollingProps = false
ActiveDocument.Sections["Chart"].HTMLDisplayXViews = true
ActiveDocument.Sections["Chart"].HTMLDisplayZViews = true
ActiveDocument.Sections["Chart"].HTMLMaxXBarsDisplayed = 10
ActiveDocument.Sections["Chart"].HTMLMaxZBarsDisplayed = 12
```

HTMLExportBreakColumnCount (Property)

Applies To:

OLAPQuerySection object, PivotSection object

Description:

Enables the number of column per exported HTML page feature in the Pivot and OLAP section. The default is one hundred. Setting the value to zero (0) causes the HTML pages not to break based on a column position.

Action:

Read-write, Number

Example 1:

This example retrieves the value of HTMLExportBreakColumnCount (Property):

```
var breakVal=ActiveDocument.Sections["Pivot"]. HTMLExportBreakColumnCount;
```

Example 2:

This example sets the number of columns per HTML page to five hundred:

```
ActiveDocument.Sections["Pivot"].HTMLExportBreakColumnCount = 500
```

HTMLExportBreakRowCount (Property)

Applies To:

OLAPQuerySection object, PivotSection object, ResultsSection object, TableSection object

Description:

Enables the number of rows per exported HTML page feature in the Pivot, OLAP, Results and Table sections. The default is one hundred. Setting the value to zero (0) causes the HTML pages not to break based on a row position.

Action:

Read-write, Number

Example 1:

This example retrieves the value of HTMLExportBreakRowCount (Property):

```
var breakVal=ActiveDocument.Sections["Pivot"]. HTMLExportBreakRowCount;
```

Example 2:

This example sets the number of rows per HTML page to five hundred:

```
ActiveDocument.Sections["Results"].HTMLExportBreakRowCount = 500
```

HTMLHorizontalPageBreakEnabled (Property)

Applies To:

OLAPQuerySection object, PivotSection object

Description:

Enables the horizontal page break control on the Export Property dialog for the Pivots section, Results and Tables sections.

Action:

Read-write, Boolean

Example 1:

This example shows how to enable the Horizontal Page Break control:

```
Documents["Sales.bqy"].Sections["Results"].HTMLHorizontalPageBreakEnabled = true
```

HTMLHorizontalPageBreakUnits (Property)

Applies To:

PivotSection object, OLAPQuerySection object

Description:

Sets the type of unit (column, pixel or none) to use for horizontal page break on a HTML page. This property is used with the BqHTMLPageBreakUnits constant group and the "HTMLExportBreakColumnCount (Property)" on page 112.

Constants:

This property references the BqHTMLPageBreakUnits constants group, which consists of these values:

- bqHTMLPageBreakUnitsCols
- bqHTMLPageBreakUnitsPixels
- bqHTMLPageBreakUnitsRows

Example:

This example shows how to set the horizontal page break for columns to two hundred columns per HTML page:

```
ActiveDocument.Sections["OLAPQuery"].HTMLHorizontalPageBreakEnabled = true
ActiveDocument.Sections["OLAPQuery"].HTMLExportBreakColCount = 200
ActiveDocument.Sections["OLAPQuery"].HTMLHorizontalPageBreakUnits =
bqHTMLPageBreakUnitsCols
```

HTMLMaxBarsDisplayed (Property)

Applies To:

ChartSection object

Description:

This property, that includes the HTMLMaxXBarsDisplayed and HTMLZBarsDisplayed, sets the number of bars to display per view if multiple views are allowed for a HTML rendered Chart (static and EPM Workspace). These properties are used with the "HTMLDisplayViews (Property) " on page 112.

The HTMLDisplayViews (Property) is unavailable for Pie charts.

If the "HTMLSyncScrollingProps (Property) " on page 115 has been set to true, then the HTMLMaxBarsDisplayed properties will be disabled. The HTMLSyncScollingProp (Property) specifies that the maximum number of bars displayed (X and Z directions) for HTML renderings match the corresponding values in the Chart Properties dialog.

Action:

Read-write, Number

Example:

This example shows how to enable multiple views of a rendered chart for the X and Y axis labels, and sets the number of bars per view at ten for the X axis label and twelve for the Z axis label:

```
ActiveDocument.Sections["Chart"].HTMLSyncScrollingProps = false
ActiveDocument.Sections["Chart"].HTMLDisplayXViews = true
ActiveDocument.Sections["Chart"].HTMLDisplayZViews = true
ActiveDocument.Sections["Chart"].HTMLMaxXBarsDisplayed = 10
ActiveDocument.Sections["Chart"].HTMLMaxZBarsDisplayed = 12
```

HTMLSyncScrollingProps (Property)

Applies To:

ChartSection object

Description:

Attaches the maximum number of bars displayed (X and Z directions) for HTML renderings (static or EPM Workspace) with the corresponding values on the Label Axis tab of Chart Properties. The default setting for this control is checked. When this control is checked, the "HTMLDisplayViews (Property) " on page 112 and "HTMLMaxBarsDisplayed (Property)" on page 115 are disabled.

Action:

Read-write, Boolean

Example:

This example shows how to enable the HTMLSyncScrollingProps (Property) and disable the HTMLDisplaysViews (Property):

```
ActiveDocument.Sections["Chart"].HTMLSyncScrollingProps = false
ActiveDocument.Sections["Chart"].HTMLDisplayXViews = false
```

HTMLVerticalPageBreakEnabled (Property)

Applies To:

OLAPQuerySection object, PivotSection object, ResultsSection object, TableSection object

Description:

Enables the vertical page break control on the Export Property dialog in the Pivots, Results and Tables sections.

Action:

Read-write, Boolean

Example:

This example shows how to enable the Vertical Page Break control:

```
Documents["Sales.bqy"].Sections["Results"].HTMLVerticalPageBreakEnabled = true
```

HTMLVerticalPageBreakUnits (Property)

Applies To:

PivotSection object, ResultsSection object, TableSection object, OLAPQuerySection object

Description:

Sets the type of unit (row, pixel or none) to use for the vertical page break on a HTML page. This property is used with the BqHTMLPageBreakUnits constant group and the "HTMLExportBreakRowCount (Property)" on page 113.

Constants:

This property references the BqHTMLPageBreakUnits constants group, which consists of these values:

- bqHTMLPageBreakUnitsCols
- bqHTMLPageBreakUnitsPixels
- bqHTMLPageBreakUnitsRows

Example:

This example shows how to print the names of all the columns in the Results section to the Console window; export the section in HTML format; enable the vertical page break unit to the row option; and set the number of rows per HTML page to one hundred:

```
MyResults = ActiveDocument.Sections["Results"]
ColumnCount = MyResults.Columns.Count
for (I=1;I<= ColumnCount;I++)
Console.Write("Column#"+I+":"+MyResults.Columns[I].Name+"\r\n")
MyResults.Export("c:\\temp\\Sample.htm", bqExportFormatHTML)
MyResults.HTMLVerticalPageBreakUnits = bqHTMLPageBreakUnitsRows
MyResults.HTMLExportBreakRowCount = 100
```

Ignore (Property)

Applies To:

Limit object

Description:

In the Query section, enables a Query to ignore limits. Consequently, the Results section is not recalculated. In the CubeQuery section, this property specifies whether a member selector should be ignored or not

Action:

Read-write, Boolean

Example:

This example shows how to temporarily ignore all Data Model limits before processing the query:

```
var MyQuery = ActiveDocument.Sections["Query" MyDM = MyQuery.DataModel
var DMLimitCount = MyDM.Limits.Count
for (j = 1 ; j <= DMLimitCount ; j++)
    MyDM.Limits[j].Ignore = true
//Assumes you are already connected
```

IncludeConsolidationInfo (Property)

Applies To:

QueryOptions object

Description:

Enabling this property includes consolidation type/unary operator information of all members from the Data Layout when Download To Results is executed in the CubeQuery section.

Disabling the option excludes the consolidation type/unary operator information for each member. By default this option is disabled for newly inserted CubeQuery sections.

Action:

Read-write, Boolean

Example:

This example shows how to enable the IncludeConsolidationInfo property:

```
ActiveDocument.Sections["Query3"].QueryOptions.IncludeConsolidationInfo = true
```

IncludeInProcessAll (Property)

Applies To:

(CubeQuery)QuerySection object, OLAPQuerySection object, QuerySection object

Description:

When read, this property indicates whether the section is included when a Process All operation is performed (value = true). If the value is false then the section is not included any Process All operations. When this property is set, a value of true ncludes the section in future Process All operations, and a value of false excludes the section from future Process All operations.

When a job is published or scheduled, all queries are listed in the Connecting to Data Sources section in order by the ordinal value of the ProcessSequence Number (Property), regardless of the setting for the IncludeInProcessAll (Property). Queries with the IncludeInProcessAll (Property) set to true are not listed in the Query Connections and Processing section. When the script is generated for the job, the processing actions is in ProcessSequenceNumber order, excluding queries where the IncludeInProcessAll (Property) is false.

When scheduling/running a job, if a query has IncludeInProcessAll (Property) set to false, but it is used in other queries as a derived table, and the excluded query has variable limits, its parameters are requested for each query that uses it. For example, Query 1 is excluded from the processing order, but Query 2 and Query 3 use it as a derived table. When collected parameters for Query 2 and Query 3, Query 1 should be included. Different parameter values could be selected for each of these queries.

When a custom process order is selected from the Process Custom dialog, all queries belonging to a given node of the processing tree (Queries, OLAPQueries, and Imported Files being the nodes) are arranged in order by the ordinal value of ProcessSequenceNumber (Property), regardless of the setting of IncludeInProcessAll (Property). Queries that are checked in that dialog are processed in ProcessSequenceNumber order, regardless of the Query type.

For example, assume a document with Query2, OLAPQuery1, authorsout.csv, Query1, and OLAPQuery2, are assigned sequence numbers 1 to 5. They appear in the tree as Query2 and Query1, in that order under Queries, and OLAPQuery1 and OLAPQuery2 under the OLAP Queries branch, and authorsout.csv under Imported Files. If users checked and five and press OK, this processing order is used:

- Query2
- OLAPQuery1
- authorsout.csv
- Query1
- OLAPQuery2

If a Query section has the IncludeInProcessAll (Property) set to false, clicking the Process button when viewing that section or any of its dependent sections, or executing the Process (Method) associated with that section still processes that query.

When a Query section is duplicated, the new section gets the next highest ProcessSequenceNumber, and it appears last in the Query Processing Order dialog list box if it is opened immediately following the duplicate section action. The IncludeInProcessAll (Property) Boolean value is inherited from the source section during duplication.

Action:

Read-write, Boolean

Example 1:

This example shows how to display the number of queries in theInteractive Reporting document file(.bqy) in an Alert box, set the processing the *Query* section to the second position in the Query Processing Order dialog, includes the *Query* section in a Process All command, and then executes the Process All command for the document:

```
Alert("Number of Query Sections "   +  ActiveDocument.Sections.QueryCount)
ActiveDocument.Sections["Query"].ProcessSequenceNum = 2
ActiveDocument.Sections["Query"].IncludeInProcessAll = true
ActiveDocument.ProcessAll()
```

Example 2:

This example show how to read and set a section in a Process All operation:

```
if (Sections[“Query”].IncludeInProcessAll == true) {
    // the OLAP query is included in the process all operation
} else {
    // Include the OLAP query in the process all operation
    Sections[“Query”].IncludeInProcessAll = true;
}
```

IncludeNulls (Property)

Applies To:

Limit object

Description:

Enables you to include null values as part of a limit.

Action:

Read-write, Boolean

Example:

This example shows how to set all the limits in the Data Model to support null values:

```
var MyQuery = ActiveDocument.Sections["Query"]
var MyDM = MyQuery.DataModel
var DMLimitCount = MyDM.Limits.Count
for (j = 1 ; j <= DMLimitCount ; j++)
    MyDM.Limits[j].IncludeNulls = true
//Assumes you are already connected
MyQuery.Process()
```

IncludeSelectedMember (Property)

Applies To:

QueryOptions object

Description:

Retains the selected member along with the other members retrieved as a result of a drill down.

Action:

Read-write, Boolean

Example:

This example shows how to enable the IncludeSelectedMember (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox7"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.IncludeSelectedMember = true}
else {ActiveDocument.Sections["Query"].QueryOptions.IncludeSelectedMember = false}
```

IncludeWithinSelectedGroup (Property)

Applies To:

QueryOptions object

Descriptions:

Applies drilling only to the group of members in which the selection is made. Enabling this property is only meaningful when the report contains two or more dimensions of data down a report as rows or across a report as columns. This type of report is considered asymmetric, which is characterized by groups of nested members that differ by at least one member.

Action:

Read-write, Boolean

Example:

This example shows how to enable the IncludeWithinSelectedGroup (Property) when a check box is checked:

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox8"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.IncludeWithinSelectedGroup = true}
else {ActiveDocument.Sections["Query"].QueryOptions.IncludeWithinSelectedGroup = false}
```

Indent (Property)

Applies To:

QueryOptions object

Description:

Sets and reads the number of characters by which each generation in the hierarchy is indented in a CubeQuery section.

Action:

Read-write, Number

Example:

This example shows how to set the indentation to 3 for each generation in the hierarchy:

```
ActiveDocument.Sections["Query"].QueryOptions.Indent = 3
```

Index (Property)

Applies To:

PivotLabel object, PivotFact object, Column object

Description:

Returns or sets the number in which values display. For example if two Pivot top labels, Units and Item ID, display in the Outliner in that order, then the Units top label has an index of 1,

and the Item ID top label has an index of 2. If you assign Units an index of 2, then it is the equivalent of dragging Units into the Outliner so that it is the second in order, after Item ID.

Action:

Read-write, PivotLabel and PivotFact

Read-only, Column

Example 1:

This example shows how to change the position of a Pivot fact:

```
ActiveDocument.Sections["SalesPivot"].Facts["Unit Sales"].Index=3
```

Example 2:

This example shows how to change the position of a column:

```
ActiveDocument.Sections["SalesResults"].Columns["Unit Sales"].Index=3
```

IntervalFrequency (Property)

Applies To:

LeftAxis object

Description:

Sets the interval frequency value of a Chart left axis. Interval frequency is a Chart property that determines how the left axis scale should display. This property is available also on the General Properties dialog, Values Axis tab. When the Interval Auto check box is disable, you can specify how refined you want the left axis displayed.

Action:

Read-write, Number

Example:

This example shows how to change the left axis to display the data in intervals of twenty thousand:

```
ActiveDocument.Sections["AllChart"].ValuesAxis.LeftAxis.IntervalFrequency=20000
```

KeepWithNext (Property)

Applies To:

PageHeader object, PageFooter object, ReportHeader object, ReportFooter object, Body object

Description:

Enables you to keep bands within a group together when paginating a report. If the lower band cannot fit on the page when the report is paginated, both bands are moved to the following page.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true (which it is by default), use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Boolean

Example:

This example shows how to keep the body band together when a page is paginated:

```
ActiveDocument.Sections["Report"].Body.KeepWithNext = true
Recalculate()
```

KeepTogether (Property)

Applies To:

PageHeader object, PageFooter object, ReportHeader object, ReportFooter object, Body object

Description:

Enables you not to split a band when a break is encountered. When a break is encountered, the entire band is moved to the next page.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true (which it is by default), use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Boolean

Example:

This example shows how not to split the page header bade when a break is encountered in a report:

```
ActiveDocument.Sections["Report"].PageHeader.KeepTogether
Recalculate()
```

LabelFormat (Property)

Applies To:

TrendLine object, ReferenceLine object

Description:

Returns or sets the text of the trend line label.

Action:

Read-write, String

Example:

This example shows how to name the trend line label to “Trend Line for Sales”:

```
ActiveDocument.Sections["Chart6"].TrendLines[1].LabelFormat = "Trend Line for Sales"
```

LabelFrequency (Property)

Applies To:

XAxis object

Description:

Returns or sets the frequency of labels displayed on a Chart X-axis.

Action:

Read-write, Number

Example:

This example shows how to change the frequency of labels to display on the X-axis to a value of three:

```
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.LabelFrequency = 3
```

LabelText (Property)

Applies To:

LeftAxis object, RightAxis object, XAxisLabel object, ZaxisLabel object

Description:

Returns or sets the value of the text associated with a Chart axis or label.

Action:

Read-write, String

Example:

This example shows how to set the text for the different labels:

```
var MyChart = ActiveDocument.Sections["Chart"]
MyChart.ValuesAxis.LeftAxis.LabelText = "Left Axis"
MyChart.ValuesAxis.RightAxis.LabelText = "Left Axis"
MyChart.LabelsAxis.XAxis.LabelText = "Xaxis"
MyChart.LabelsAxis.ZAxis.LabelText = "Zaxis"
```

LastPrinted (Property)

Applies To:

QuerySection object,

Description:

Returns a data object corresponding to the last date a section was printed. To get the date value, use the methods and properties of the Date object.

Action:

Read-only: Date Object

Example:

This example shows how to print the date that the document was last printed in the Console window:

```
Console.Writeln(ActiveDocument.Sections["Query"].LastPrinted.toString())
Thu Jun 03 13:56:13 GMT-0700 (Pacific Daylight Time) 2001
```

LastSQLStatement (Property)

Applies To:

QuerySection, Document, PluginDocument object

Description:

Returns the last Structured Query Language (SQL) statement generated by a query.

Action:

Read-only

Example

This example shows how to display the last SQL statement generated by a query in an Alert box:

```
Alert (ActiveDocument.Sections["Query"].LastSQLStatement)
```

LeftMargin (Property)

Applies To:

ReportSection object

Description:

Sets the left margin tab of the report. Margins are set for the entire report.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true (which it is by default), use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Number

Example:

This example shows how to set the left margin of the report to .25 inches:

```
ActiveDocument.Sections["Report"].LeftMargin = .25
```

LegendFormat (Property)

Applies to:

TrendLine object

Description:

Returns or sets the text of the trend line legend.

Action:

Read-write, String

Example:

This example shows how to name the trend line legend text to "Trend Line for North Eastern Sales":

```
ActiveDocument.Sections["Chart6"].TrendLines[1].LegendFormat = "Trend Line for North
Eastern Sales"
```

LimitValueType (Property)

Applies To:

Limit collection

Description:

Returns or sets the value of the selected filter value set. That is, you can select in advance to use the Available values (Show values) or Custom values on the Filter dialog box.

Action:

Read-write

Constants:

The LimitValueType (Property) uses the BqLimitValueType constant group, which consists of these values:

- bqLimitValueTypeAvailable
- bqLimitValueTypeCustom
- bqLimitValueTypeSQL

Example:

This example shows how to select the custom values for the second filter item on the Filter dialog box:

```
ActiveDocument.Sections["Query"].Limits[2].LimitValueType=bqLimitValueTypeCustom
```

Locked (Property)

Applies To:

Bullet object, (Live) BarChart object, (Live) BlockChart object, ControlsCheckBox object, ControlsCommandButton object, ControlsDropDown object, ControlsListBox object, ControlsRadioButton object, ControlsTextBox object, Dashboard objects, EmbeddedSection object, Graphics object, HyperlLink object, LineChart object, (Live) PieChart object, (Live) RadarChart object, Shapes collection, Slider object, Speedometer object, Thermometer object, TrafficlLight object

Description:

Sets whether an object can be moved or deleted in dashboard design mode. In locked mode, scripts and properties can still be added or changed.

Action:

Read-write, Boolean

Example:

This example shows how to lock a (Live) pie chart:

```
PieChart.Locked = true
```

LogicalOperator (Property)

Applies To:

Limit collection

Description:

Sets the value of the filter logical operator of each filter object. The LogicalOperator (Property) is ignored when only one filter value displays for a section. The filter LogicalOperator (Property) is always ignored for the first filter value when multiple filter values exist. If multiple filter values are in a section, the LogicalOperator of the second filter applies to the relationship between the first and second filter values. The LogicalOperator of the third filter applies to the relationship between the second and third filter values, and so on.

Action:

Read-write, BqLogicalOperator

Constants:

The LogicalOperator (Property) uses the BqLogicalOperator constant group, which consists of the bqLogicalOperatorAND (default value) and bqLogicalOperatorOR values.

Example:

This example shows how to set the *OR* logical operator on a filter object:

```
ActiveDocument.Sections["SalesQuery"].Limits["Year"].LogicalOperator=bqLogicalOperatorOR
```

MajorInterval (Property)

Applies To:

NumericRange object

Description:

Returns or sets the major incremental value to use when the indicator is moved along the major scale of a gauge.

Action:

Read-write: Number

Example:

This example shows how to set the major interval for a Speedometer object to 20:

```
Speedometer.NumericRange.MajorInterval = "20"
```

MarkerBorderColor (Property)

Applies To:

Legend Collection

Description:

Returns or sets the color of a marker border. A marker depicts a data value or point that emerges in a cell.

Action:

Read-write

Constants:

The following values are some of those in the BqColorType constant group. For a complete list, see the BqColorType constant group in the Script Editor:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937
- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119
- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120
- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410

- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232
- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637
- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215
- bqYellow = 16776960

Example:

This example shows how to set the marker border color to blue:

```
ActiveDocument.Sections["AllChart"].Legend.Items["Unit Sales"].Line.
MarkerBorderColor=bqBlue
```

MarkerFillColor (Property)

Applies To:

Legend Collection

Description:

Returns or sets the fill color property of a marker. A marker depicts a data value or point that emerges in a cell.

Action:

Read-write

Constants:

The following values are some of those in the BqColorType constant group. For a complete list, see the BqColorType constant group in the Script Editor:

- bqAqua = 65535
- bqBlack = 0
- bqBlue = 255
- bqBlueGray = 6710937
- bqBrightGreen = 65280
- bqBrown = 10824234
- bqDarkBlue = 119
- bqDarkGreen = 13056
- bqDarkRed = 9109504
- bqDarkTeal = 18504
- bqDarkYellow = 14329120
- bqGold = 16763904
- bqGray40 = 10066329
- bqGray50 = 8355711
- bqGray80 = 3355443
- bqGreen = 30464
- bqIndigo = 4915330
- bqLavender = 15132410
- bqLightBlue = 11393254
- bqLightGreen = 13434828
- bqLightOrange = 16750848
- bqLightTurquoise = 11529966
- bqLightYellow = 16777113
- bqLime = 10079232
- bqOliveGreen = 3355392
- bqOrange =16737792
- bqPaleBlue = 10079487
- bqPink = 16711936
- bqPlum = 14524637

- bqRed = 16711680
- bqRose = 16751052
- bqSeaGreen = 3050327
- bqSkyBlue = 8900331
- bqTan = 13808780
- bqTeal = 30840
- bqTransparent = 16711422
- bqTurquoise = 4251856
- bqViolet = 7864440
- bqWhite = 16777215
- bqYellow = 16776960

This example shows how to set the marker fill color to green:

```
ActiveDocument.Sections["AllCart"].Legend.Items["Unit Sales"].Line.
MarkerFillColor=bqGreen
```

MarkerSize (Property)

Applies To:

Legend Collection

Description:

Returns or sets the size property of a marker. A marker depicts a data value or point that emerges in a cell.

Action:

Read-write, Number

Example:

This example shows how to set the marker size property to six points:

```
ActiveDocument.Sections["AllChart"].Legend.Items["Unit Sales"].Line.MarkerSize=6
```

MarkerStyle (Property)

Applies To:

Legend Collection

Description:

Returns or sets the style property of a marker, such as diamond-shaped, circular, rectangular or triangular. A marker depicts a data value or point that emerges in a cell.

Action:

Read-write

Constants:

The BqMarkerStyle constant group consists of these values:

- bqMarkerStyleCircle
- bqMarkerStyleDiamond
- bqMarkerStyleRectangle
- bqMarkerStyleTriangle

Example:

This example shows how to set the marker style to triangular:

```
ActiveDocument.Sections["AllChart"].Legend.Items["Unit Sales"].Line.
MarkerStyle=bqMarkerStyleTriangle
```

Max (Property)

Applies To:

ColorRange object

Description:

Sets or returns the maximum color range value to which a color is applied. If you are adding multiple color ranges, each consecutive maximum color range value must be greater than the preceding maximum color range value. The maximum color range value cannot be greater than the maximum value defined for the numeric scale of the gauge. To use the numeric maximum scale range, do not set this property.

Action:

Read-write, Number

Example:

This example shows how to set the maximum value for the first color range object to 100:

```
Speedometer.ColorRanges[1].Max = "100"
```

MaxBubbleSize (Property)

Applies To:

BubbleChart object

Description:

Sets the maximum size of the bubble in the grid area. This size is measured as a percentage value. The default value is 25%

Example:

This example shows how to set the maximum bubble size to 20%.

```
ActiveDocument.Sections["UnitMonthChart"].BubbleChart.MaxBubbleSize= 20
```

MaximumBarsDisplayed (Property)

Applies To:

XAxisLabel object, ZAxisLabel object

Description:

Sets the number of bars to display per view if multiple views are allowed in the Chart section. This property is used with the "MaximumBarsEnabled (Property) " on page 134, which enables the multiple views option. The MaximumBarsDisplay (Property) is unavailable for Pie charts.

Action:

Read-write, Numeric

Example:

This example shows how to enable multiple views of a chart in the Chart section, and how to set the maximum bars displayed value to 10 bars:

```
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.MaximumDisplayViews = true
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.MaximumBarsEnabled = 10
```

MaximumBarsEnabled (Property)

Applies To:

XAxisLabel object, ZAxisLabel object

Description:

Enables multiple views of a rendered chart in the Chart section by scrolling. This property is used with the "MaximumBarsDisplayed (Property) " on page 134, which enables you to define the bars shown per view. The MaximumBarsEnabled (Property) is unavailable for Pie charts.

Action:

Read-write, Boolean

Example:

This example shows how to enable multiple views of a rendered chart and set the number of bars per view at 10:

```
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.DisplayViews = true
ActiveDocument.Sections["Chart"].LabelsAxis.XAxis.MaxBarsDisplayed = 10
```

MaxScale (Property)

Applies To:

NumericRange object

Description:

Returns or sets the maximum data value of the numeric value scale for a gauge. To use this property, the AutoScale (Property) must be set to false.

Action:

Read-write, Number

Example:

This example shows how to set the maximum value for a Speedometer object to 100.

```
Speedometer.NumericRange.MaxScale = "100"
```

Member (Property)

Applies To:

OLAPFilter object

Description:

Returns the member to which a filter is applied.

Action:

Read only, String

Example:

This example shows how to return the member to which the filter has been applied:

```
if (Sections["Query"].Filters["Product"].Member == "100-20") {
    // There is a filter on the 100-20 member of the Product dimension
}
```

MetadataPassword (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the password used in the metadata connection.

Action:

Read-write, String

Example:

This example creates a connection file and applies it to the current document. The data source name is *PlutoSQLSVR* which is a user DSN using the SQL Server 6.5 drive:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto.
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.MetadataUsername = "hyperion"
myCon.MetadataPassword = "hyperionhyperion"
myCon.UseAlternateMetadataLocation(true,c:\\OCEs\\PlutoMeta.OCE)
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

MetadataUser (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the value of the user name used to connect to the metadata data source.

Action:

Read-write, String

Example:

This example creates a connection file and applies it to the current document. The data source name is *PlutoSQLSVR* which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto.
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.MetadataUsername = "hyperion"
myCon.MetadataPassword = "hyperionhyperion"
myCon.UseAlternateMetadataLocation(true,c:\\OCEs\\PlutoMeta.OCE)
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

MetaFileChoice (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the value of the MetaData source from the Bqmeta0.ini file. The metadata source is the name of the predefined metadata vendor.

Action:

Read-write, String

Example:

This example creates a connection file and applies it to the current document. The data source name is *PlutoSQLSVR* which is a user DSN using the SQL Server 6.5 driver:

Note: Do not use MetaFileChoice in Interactive Reporting document files (.bqy).

Example:

This example shows how to set the value of the MetaData source.

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto.
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.MetadataUsername"hyperion"
myCon.MetadataPassword = "hyperionhyperion"
myCon.MetaFileChoice = "Broadbase"
myCon.UseAlternateMetadataLocation(true,c:\\OCEs\\PlutoMeta.OCE)
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

Min (Property)

Applies To:

ColorRange object

Description:

Sets or returns the minimum value to which a color is applied. The minimum number cannot be less than the minimum value defined for the numeric scale of the gauge. If you are adding multiple color ranges, each consecutive minimum color range value must be no less than the preceding maximum color range value. To use the numeric minimum scale range, do not use set this property.

Action:

Read-write, Number

Example:

This example shows how to set the maximum value for the first color range object to 10:

```
Speedometer.ColorRanges[1].Min = "10"
```

MinFontSize (Property)

Applies To:

ChartSection object

Description:

Sets the minimum font size applied to text labels when the overall Chart size is changed. Normally, Interactive Reporting tries to draw a Chart component using the requested font size of the component. If it is not possible to fit it in the available space, the font size is reduced by one point and a redraw is attempted. This font size reduction process is repeated until the component can fit within the component's boundary or until it reaches the value specified here or the corresponding Min Font Size field on the General tab of Chart properties.

The components to which the MinFontSize property is applied are:

- X, Y, Z axis values
- pie slice labels
- chart legend

When the component is reduced, the selection of the component continues to show the original font size and not the reduced font size in the Min Font Size on the General tab during the reduction process. When the minimum font size is reached for the supported components, and the component is still too large to fit without obstructing or being obstructed, then the component is removed from the display.

The default minimum font size is 8.

Action:

Read-write, Number

Example:

This example shows how to set the font size twelve points:

```
ActiveDocument.Sections["Chart"].MinFontSize = 7
```

MinorInterval (Property)

Applies To:

NumericRange object

Description:

Returns or sets the minor incremental value to use when the indicator is moved along the minor scale of the gauge.

Action:

Read-write: Number

Example:

This example shows how to set the major interval for a Speedometer object to 10:

```
Speedometer.NumericRange.MajorInterval = "10"
```

MinScale (Property)

Applies To:

NumericRange object

Description:

Returns or sets the minimum data value of the numeric value scale for a gauge. To use this property, the AutoScale (Property) must be set to false. By default the minimum data value is set to 0.

Action:

Read-write, Number

Example:

This example shows how to set the maximum value for a Speedometer object to 5.

```
Speedometer.NumericRange.MinScale = "5"
```

Modified (Property)

Applies To:

Document object

Description:

The Modified (Property) returns the boolean value if an application has been modified.

Note: An Interactive Reporting document file deployed in the EPM Workspace dirties documents slightly differently from one deployed inInteractive Reporting Web Client. Switching sections will not dirty an Interactive Reporting document, but will in Interactive Reporting Web Client. Any action that makes a server round trip will dirty a document, including an OM view-only property.

Action:

Read-only as Boolean

Example:

This example shows how to indicate that the application has been modified:

```
Alert(ActiveDocument.Modified)
```

ModifiedAppType (Property)

Applies To:

Document object

Description:

Returns the application name in which the Interactive Reporting document was modified.

Action:

Read-only

Constants:

The ModifiedAppType (Property) uses the BqAppType constant group, which consists of these values:

- bqAppTypeDesktopClient = 1
- bqApTypePlugInClient = 2
- bqAppTypeScheduler = 4
- bqAppTypeSmartViewClient = 5
- bqAppTypeThinClient = 3
- byAppTypeUnknown = 0

Example:

This example shows how to use and application type switch statement:

```
switch(ActiveDocument.ModifiedAppType)

{

case 0:

TextBox3.Text = ".ModifiedAppType Type is: bqAppTypeUnknown"

break;

case 1:

TextBox3.Text = ".ModifiedAppType Type is: bqAppTypeDesktopClient"

break;

case 2:

TextBox3.Text = ".ModifiedAppType Type is: bqAppTypePlugInClient"

break;

case 3:

TextBox3.Text = ".ModifiedAppType Type is: bqAppTypeThinClient"
```

```
break;
case 4:
TextBox3.Text = ".ModifiedAppType Type is: bqAppTypeScheduler"
break;
case 5:
TextBox3.Text = ".ModifiedAppType Type is: bqAppTypeScheduler"
break;
default:
TextBox3.Text = "No Section Value Available"
}
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
```

ModifiedAppVersion (Property)

Applies To:

Document object

Description:

Returns the application version in which a document was created.

Action:

Read-only as String

Example:

This example shows how to display the application type for the Chart.bqy in an Alert box:

```
Alert(Documents["Sample - charts.bqy"].ModifiedAppVersion)
```

MultiSelect (Property)

Applies To:

ControlsListBox object

Description:

Enables multiple items to be selected from a list box control.

Action:

Read-write, Boolean

Example:

This example shows how to configure a list box to support multiple user selections:

```
var MyDashboard = ActiveDocument.Sections["Dashboard"]
MyDashboard.Shapes"Listbox1"].MultiSelect = true
```

Name (Property)

Applies To:

AliasTableName object, Bullet object, Application object, Association object, (Live) BarChart object, (Live) BlockChart object, ChartSection object, Column object, Control object, ControlsCheckBox object, ControlsCommandButton object, ControlsDropDown object, ControlsListBox object, ControlsRadioButton object, ControlsTextBox object, (CubeQuery)QuerySection object, DataModelSection object, DerivableQuery object, DerivedItem object, DerivedTable object, DMCatalogItem object, DMResult object, Document object, DashboardSection object, EmbeddedBrowser object, EventScript object, Fact object, HyperLink object, LineChart object, OLAPLevelOrHierarchyNew collection, OLAPMemberSelector object, OLAPFilter object, OLAPQuerySection object, (Live) PieChart object, PivotLabelValue object, PivotSection object, PluginDocument object, (Live) RadarChart object, QuerySection object, ReportSection object, RepositoryItem object, ResultsSection object, Section object, Theme object, Slider object, SortItem object, Speedometer object, StoredProcedure object, TableSection object, TargetFact object, Thermometer object, Toolbar object, TrafficLight object

Description:

Returns or sets the name of an object listed above.

There are three "name" type properties used in the Object Model, including:

- "DisplayName (Property)" on page 88
- "PhysicalName (Property)" on page 155
- Name (Property)

To determine which name property to use in the Object Model, consider this example.

A database table contains a column titled MyColumn. The *PhysicalName* of the column is MyColumn. To show another column name, such as MySpecialColumn, use the *DisplayName* (Property). This property enables you to modify the name shown in the Interactive Reporting product suite although the source column retains the original name (MyColumn). Use the

Name (Property) to reference Object Model objects. For example, to determine the name of a TextBox object in Dashboards, write this script:

```
TextBox1.Name
```

Note: Do not use Toolbars["Standard"].Name, Toolbars["Formatting"].Name, Toolbars["Sections"].Name, and Toolbars["Navigation"].Name in Interactive Reporting document files s to be deployed in the EPM Workspace.

Action:

Read-only, String

Application object, Column object, Control object, ControlsCheckBox object, ControlsCommandButton object, ControlsDropDown object, ControlsListBox object, ControlsRadioButton object, ControlsTextBox object, Dashboard objects, PivotLabelValue object, Toolbar object

Read-write, String

ChartSection object, DataModelSection object, DMCatalogItem object, DMResult object, Document object, DashboardSection object, OLAPQuerySection object, PivotSection object, PluginDocument object, QuerySection object, Section object, TableSection object

Example:

This example prints a list of all the sections in a document to the Console window:

```
for (j = 1 ; j <= ActiveDocument.Sections.Count ; j ++)
Console.Writeln(ActiveDocument.Sections[j].Name)
```

Negate (Property)

Applies To:

Limit object

Description:

Enables negation to be applied to the filter operator. For example, if a filter operator is set to select only the values equal to a criteria and the negate property is true, then the values returned from the query are not equal to the criteria.

Action:

Read-write, Boolean

Example:

This example shows how to set the negate property of a limit:

```
var MyLimit = ActiveDocument.Sections["Query"].Limits["State"]
```

```
MyLimit.Negate = true
```

NumberFormat (Property)

Applies To:

Column object, FactsAxis object, MemberSelector object, Trendline object, ValueLabels object

Description:

Enables you to format the numeric values for a Results or Table column, Chart facts axis, OLAPQuery, CubeQuery, and Pivot facts, trend lines and reference line objects (Chart section).

Setting the NumberFormat (Property) corresponds to setting the number format on the Number tab of the Default Formats dialog box.

In the OLAPQuery section, the NumberFormat property is available for objects in the Facts collection.

In the CubeQuery sections, setting the NumberFormat property updates all the fact cells for the selected MemberSelector object in the entire column/row in the native CubeQuery grid and any CubeQuery embedded section objects upon script execution. You can apply number formats by Members of the Measures Dimension as well as Members in any other Dimension. If the MemberSelector object has advanced member selection types, the NumberFormat is applied only to member itself. If members are excluded from selection, the number format is not be applied to any cells. In this case, you should explicitly add all the members to which you want to apply a number format.

For the trend line object, the NumberFormat affects the look of R-Squared value of trend line. The R-Squared parameter (also know as the coefficient of determination) is a value in a [0, 1] range. It shows how good the trend line conforms to data.

For the reference line, the NumberFormat affects the appearance of the values shown on the line.

Interactive Reporting does not validate whether the NumberFormat string is recognized as a valid number format.

NumberFormat property defaults to no value.

Number formats can be applied to particular cell in various ways (GUI with various selection methods, Object Model, and defaults). The order of priority for applying the number format is as follow from lowest to highest priority:

- number formats defined in the Number tab of the Default Formats dialog box
- number formats defined in the user interface by selecting the column which contain the cell
- number formats defined in the user interface by selecting the row which contain the cell
- number formats defined in the user interface by selecting the cell
- number formats defined in the Object Model for row dimensions and members which correspond to the cell. The settings of posterior dimensions override settings of previous ones.

- number formats defined in the Object Model for column dimensions and members which correspond to the cell. The settings of posterior dimensions override settings of previous ones.

Action:

Read-write, String

Example 1:

This example shows how to apply currency number formatting to data in the Results section:

```
ActiveDocument.Sections["SalesResults"].Columns["Amount Sales"].NumberFormat="$#,##0.00"
```

Example 2:

This example shows how to apply a number format to the 100 Column in the CubeQuery section:

```
ActiveDocument.Sections[“Query”].Columns[“Product”].MemberSelectors[“100”].NumberFormat
= “$#,##0.00;[$#,##0.00]
```

ODBCDatabasePrompt (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

ODBC Only. The ODBCDatabasePrompt (Property) enables a prompt so that users have to enter the name of the ODBC database.

Action:

Read-write, Boolean

Example:

This example creates a connection file and applies it to the current document. The data source name in this example is *PlutoSQLSVR* which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
myCon.Description"This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.ODBCDatabasePrompt = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

ODBCEnableLargeBufferMode (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

ODBC Only. The ODBCEnableLargeBufferMode (Property) enables ODBC connections to use a larger buffer mode.

Action:

Read-write, Boolean

Example:

This examples shows how to create an Interactive Reporting database connection file and applies it to the current document. The data source name in this example is *PlutoSQLSVR*, which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()myCon.Description = "This OCE configures the
connection via ODBC, to a SQLServer 6.5 database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.ODBCEnableLargeBufferMode = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

ODSUsername (Property)

Applies To:

WebClientDocument object (Quickview and FreeView Only)

Description:

Returns or sets the value of the user name when connecting to the OnDemand Server. This property only applies if an Interactive Reporting document deployed in the Interactive Reporting Web Clienthas been saved to a local file system. This property can be used to reconnect without prompting to logon to the ODS.

Action:

Read-write, String

Example:

This example shows how to connect to the OnDemand server from a script.

Note: This script only applies to Interactive Reporting documents files that have already been registered to the OnDemand server and saved locally.

```
ActiveDocument.ODSUsername = "Hyperion"
ActiveDocument.SetODSPassword("HyperionHyperion")
```

OfficeHTMLFormulasEnabled (Property)

Applies To:

ResultsSection object, TableSection object

Description:

Enables generation of Microsoft Excel formulas when exporting Results and Tables sections to Microsoft Office HTML and MHTML formats. Adding formulas allows for automatic recalculation of Computed Items and Break/Grand Total in an Excel worksheet.

Action:

Read-write, Boolean

Example:

This example shows how to disable the formula generation:

```
Documents["Sales.bqy"].Sections["Results"].OfficeHTMLFormulasEnabled = false
```

OlapCalculationType (Property)

Applies To:

QueryOptions object

Description:

Returns the method for how database totals are calculated in a CubeQuery section. This property corresponds to the settings in the Results group of Query-> Query Options >Global.

Action:

Read-write, BqOlapCalculationType

Consants:

The OlapCalculationType (Property) uses the BqOlapCalculationType constant group, which consists of these values:

- bqOlapCubeCalculator = 1: disables both Download Database Totals and Allow Relational Aggregation
- bqOlapDatabaseAndLocalTotals = 3: enables Download Database Totals and Allow Relational Aggregation
- bqOlapDatabaseTotals = 2: enables Download Database Totals and disables Allow Relational Aggregation

Example:

This examples shows how to return the calculation type for a CubeQuery section in an Alert box:

```
Alert(ActiveDocument.Sections["Query"].QueryOptions.OlapCalculationType)
```

Operator (Property)

Applies To:

Limit object

Description:

Returns or sets the value of a filter operator. The operator is applied to the filter criteria when executing a query or recalculating a Results set. If the operator is set to Equal, only the values which are exactly equal to the filter criteria, are returned or displayed.

Action:

Read write, BqLimitOperator

Constants:

The Operator (Property) uses the BqLimitOperator constant group, which consists of these values:

- bqLimitOperatorBeginsWith
- bqLimitOperatorBetween
- bqLimitOperatorContains
- bqLimitOperatorCustomSQL
- bqLimitOperatoEndsWith
- bqLimitOperatorEqual
- bqLimitOperatorGreaterThan
- bqLimitOperatorGreaterThanOrEqual
- bqLimitOperatorIsNull
- bqLimitOperatorLessThan

- bqLimitOperatorLessThanOrEqual
- bqLimitOperatorLike
- bqLimitOperatorNotEqual

Example:

This example shows how to modify values of an existing results limit:

```
MyLimit = ActiveDocument.Sections["Results"].Limits[1]
//Clear all the values which are currently set
MyLimit.SelectedValues.RemoveAll()
// add new values to the selectedvalues collection
MyLimit.SelectedValues.Add(2000)
//Change the limit criteria
MyLimit.Operator = bqLimitOperatorLessThan
```

Orientation (Property)

Applies To:

ReportSection object, (Live) BarChart object, (Live) BlockChart object

Description:

For a ReportSection object, returns the value of portrait (vertical) or landscape (horizontal) for the page orientation of the printed report.

For a (Live) BarChart object, sets the position of the (Live) Chart object defined by the BqObjectOrientation constant group, which includes the following values:

- bqObjectOrientationDown
- bqObjectOrientationHorizontal
- bqObjectOrientationLeft
- bqObjectOrientationRight
- bqObjectOrientationUp
- bqObjectOrientationVertical

The (Live) FunnelChart object uses the following values of the BqObjectOrientation constant group:

- bqObjectOrientationDown
- bqObjectOrientationLeft
- bqObjectOrientationRight
- bqObjectOrientationUp

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true, use the Recalculate (Method) to recalculate the Report section.

Action:

Read-only, String

Constants:

The Orientation (Property) uses the BqOrientation constant group, which consists of the bqOrientationPortrait and bqOrientationPortrait values. It also uses the BqObjectOrientation constant group.

Example:

This example shows how to set the page orientation to landscape:

```
ActiveDocument.Sections["Report"].Orientation = bqOrientationLandscape
```

Owner (Property)

Applies To:

DMCatalogItem object

Description:

Returns the value of the database owner name associated with a Table in the Table Catalog.

Action:

Read-only, String

Example:

This example shows how to write all the information about the tables in the Table Catalog to the Console window:

```
with (ActiveDocument.Sections["Query"].DataModel)
{
var NumTables = Catalog.CatalogItems.Count
    for (I = 1 ; I <= NumTables ;I++)
    {
OutputString = "Database Name =" + Catalog.CatalogItems[I].DatabaseName
OutputString = OutputString +":Database Owner=" + Catalog.CatalogItems[I].Owner
OutputString = OutputString +":Table Name=" + Catalog.CatalogItems[I].Name
Console.Writeln(OutputString)
    }
}
```

PageBreak (Property)

Applies To:

PageHeader object, PageFooter object, ReportHeader object, ReportHeader object, Body object

Description:

Returns or sets the value on where to page break in a report. A page break cannot be inserted before a Report Header Group, Page Header, Body or Page Footer.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true, use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Boolean

Constants:

The PageBreak (Property) uses the BqPageBreak constant group which consists of these values:

- bqPageBreakBoth
- bqPageBreakAfter
- bqPageBreakBefore
- bqPageBreakNone

Example:

This example shows how to insert a page break after the Report Header group:

```
ActiveDocument.Sections["Report"].ReportHeader.PageBreak = bqPageBreakAfter
```

ParentName (Property)

Applies To:

OLAPLevelorHierarchy collection, (CubeQuery) OLAPLevelorHierarchyNew collection, Measures collection

Description:

Returns the parent name value of the OLAP level or hierarchy.

Action:

Read only, String

Example 1:

This example shows how to display the parent name of an OLAP level or hierarchy in an Alert box:

```
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Customers"]
["Country"].ParentName)
```

Example 2:

This example shows how to return the parent name and unique name for the product dimension in a Console window:

```
var products = Sections["Query"].Catalog.Dimensions["Product"]["100"];
for (var i = 1; i <= products.Count; i++) {
    Console.Writeln(products[i].Name);
    Console.Writeln(products.Item(i).UniqueName);
    Console.Writeln(products.Item(i).ParentName);
}
```

Password (Property)

Applies To:

ControlsTextBox object

Description:

Returns or sets the value of a text box password setting. If this property is true, the text in the text box will be replaced with ****.

Action:

Read-only, String

Example:

This example shows how to set the password property on a text box:

```
ActiveDocument.Sections["Dashboard"].Shapes["TextBox1"].Password = true
```

Path (Property)

Applies To:

Document object, PluginDocument object

Description:

Returns a string containing the full path and name of the document.

Note: A plugin document name will be the temporary name and path of the document on the local file system. For information about Web server path, refer to the URL property.

Action:

Read-only, String

Example:

This example prints out the path information for all the open documents to the Console window:

```
for ( j = 1 ; j < = Documents.Count ; j++)\
     Console.Writeln( Documents[j].Name + is located on +Documents[j].Path)
```

PathSeparator (Property)

Applies To:

Application object

Description:

Returns the separator used by the operating systems file system as shown below.

Windows – "\

UNIX – "/

Macintosh – ":

Action:

Read-only, String

Example:

This example shows how to use the path separator to build a path:

```
var PS = Application.PathSeparator
Alert(PS)
```

Pattern (Property)

Applies To:

FillFormat object

Description:

Returns or sets the background fill pattern of an object. The fill pattern refers to the level of shading used in the background object.

Action:

Read-only

Constants:

The Pattern (Property) uses the BqFillPattern constant group, which consists of these values:

- bqFillPattern100
- bqFillPattern25
- bqFillPattern50
- bqFillPattern75
- bqFillPatternNone

Example:

This example shows how to use the path separator to build a path:

```
var PS = Application.PathSeparatorvar myDir = "c:"+PS+"Documents"+PS+"Hyperion Docs"+PS
+"Sales Reports"Alert(myDir)
```

PhysicalName (Property)

Applies To:

Topic object, TopicItem object

Description:

Returns the actual name of the topic or topic item. This name cannot be changed through scripting or through the user interface.

There are three "name" type properties used in the Object Model, including the:

- "DisplayName (Property)" on page 88
- PhysicalName (Property)
- "Name (Property)" on page 143

To determine which "name" property to use in Object Models, consider this example.

A database table is called MyColumn. The *PhysicalName* of the column is also MyColumn. To display a different column name, such as MySpecialColumn, use the *DisplayName* (Property). This enables you to modify the name shown in the Interactive Reporting product suite although at the source, the column retains its original name (MyColumn). Use the *Name* (Property) to reference Object Model objects. For example, to determine the name of a TextBox object in Dashboards, write this script:

```
TextBox1.Name
```

Action:

Read-write, String

Example:

This example writes the names of all the topics and topic items to the Console window:

```
var Tcount = ActiveDocument.Sections"Query"].DataModel.Topics.Count
```

```
for (j = 1; j <= Tcount ; j ++)
{
var myTopic = ActiveDocument.Sections["Query"].DataModel.Topics[j]
Console.Writeln("Topic : "+myTopic.PhysicalName)
var TICount = ActiveDocument.Sections["Query"].DataModel.Topics[j].TopicItems.Count
for (k = 1 ; k <= TICount ; k ++)
{
var myItem = ActiveDocument.Sections["Query"].DataModel.Topics[j].TopicItems[k]
Console.Writeln("Topic Item: "+ myItem.PhysicalName)
}
}
```

PreloadHomeSection (Property)

Applies to:

ActiveDocument object

Description:

Enables the pre-loading of the active Dashboard home section with at least one embedded browser object to increase loading performance in the EPM Workspace. This method is useful when there are little or no startup scripts associated with the Dashboard section. When the Interactive Reporting Service encounters an embedded browser object, it generates and sends the empty Dashboard HTML with the embedded browser object to the browser, which contains only iframes of the embedded browser object. When the HTML is rendered, the browser processes the URLs of the iframes automatically. At the same time, the Interactive Reporting Service resumes the processing of the Dashboard section and executes any Object Model script if it exist. Finally the browser is refreshed with the actual Dashboard HTML. This method preloads the section named in the argument. By default, if no section name is specified, the active Dashboard section is preloaded.

Use PreloadHomeSection (Property) with the HomeDashboard (Property) . This property sets the default Dashboard Home section at publishing time, which instructs the Interactive Reporting Service to open this section instead of the last-opened section set in the Interactive Reporting document (.bqy). The HomeDashboard (Property) is equivalent to selecting a section on the Dashboard Home dialog. If the Dashboard Home section has been selected in the user interface, it overrides any selection made in the HomeDashboard (Property).

Action:

Read-write, Boolean

Example:

This example show how to preload the active Dashboard home section:

```
ActiveDocument.PreloadHomeSection = true
ActiveDocument.HomeDashboard= "Dashboard"
```

PrintAllViews (Property)

Applies To:

ChartSection object

Description:

Prints prints the entire rendered chart (all bars are printed on a page).

The setting you specify here is not applied to the Export to PDF option in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to add a title to a chart:

```
ActiveDocument.Sections["Chart"].PrintAllViews = true
```

ProcessEventOrigin (Property)

Applies To:

Document object

Description:

Identifies how the Process() event was initiated.

Action:

Read-only

Constants:

The ProcessEventOrigin (Property) uses the BqRequestEventOriginType constant group, which consists of these values:

- bqRequestEventOriginScript
- bqRequestEventOriginMenu
- bqRequestEventOriginToolbar

Example:

This example shows how to identify the origin of the Process event:

```
Console.Writeln("Start OnPreProcess")
//determine process event origin
Console.Writeln("Process Event Origin is: " + ActiveDocument.ProcessEventOrigin)
```

```
//write process event origin to the selected console technique
switch(ActiveDocument.ProcessEventOrigin)
{
case 0:
Console.Writeln("Switch: Process Event Origin is 0, Menu")
break;
case 1:
Console.Writeln("Switch: Process Event Origin is 1, Toolbar")
break;
case 2:
Console.Writeln("Switch: Process Event Origin is 2, Script")
break;
default:
break;
}
Console.Writeln("End OnPreProcess")
```

ProcessSequenceNum (Property)

Applies To:

(CubeQuery) QuerySection object, OLAPQuerySection Object QuerySection object

Description:

Sets the ordinal equivalent to a query position in the Query Processing Order dialog. That is, when the query is listed first in the list box, the ProcessSequenceNum is one (1). When it is listed second, this property is two (2) and so on.

This property may be used in association with the "IncludeInProcessAll (Property)" on page 118. The value selected here is relevant even if the IncludeInProcessAll (Property) is false.

A catchable exception is thrown if the value to be assigned is less than 1. If the user-assigned value is higher than the count of Query sections in the document, it is silently reset to be equal to the count assigned by the "QueryCount (Property)" on page 161.

When ProcessSequenceNum (Property) is assigned to a query section, the Query section that previously had this value and all subsequent or preceding Query section, have their ProcessSequenceNum (Property) incremented or decremented by one (1). This recalculation fills in the gap left by the old sequence number of the selected Query section. For example, if QueryA, OLAPQueryB, and Query C have the property values set to 1, 2, and 3 respectively, these recalculations occur:

- If QueryA is assigned a ProcessSequenceNum (Property) of 1, no change occurs because the ProcessSequenceNum (Property) of QueryA already has that value.
- If QueryC is assigned a ProcessSequenceNum (Property) of 1, QueryA has its ProcessSequenceNum (Property) incremented by 1 to 2, and OLAPQueryB is incremented by 1 to 3.
- If the second example is revered and OLAPQueryB is assigned a ProcessSequenceNum (Property) of 3, QueryC, consequently, has its ProcessSequenceNum (Property)

decremented by 1 to 2. There would be no change to the ProcessSequenceNum (Property) for QueryA.

The Boolean value selected for this property is the equivalent of the asterisk (*) in the Query Processing Order dialog. Use this property to return or set the value of the IncludeInProcessAll (Property) .

If this property is true, the query is processed in the order specified by the *ProcessSequenceNum* (Property) or as specified on the Query Processing Order dialog when a *Process All* action is requested.

If this property is false, the query is not included in the Query Processing Order.

When a job is published or scheduled, all queries are listed in the Connecting to Data Sources section in order by the ordinal value of the ProcessSequence Num (Property), regardless of the setting for the IncludeInProcessAll (Property). Queries with the IncludeInProcessAll property set to true are not listed in the Query Connections and Processing section. When the script is generated for the job, the processing actions is in ProcessSequenceNum order, excluding queries where IncludeInProcessAll (Property) is false.

When scheduling/running a job, if a query has IncludeInProcessAll (Property) set to false, but it is used in other queries as a derived table, and the excluded query has variable limits, its parameters are requested for each query that uses it. For example, Query 1 is excluded from the processing order, but Query 2 and Query 3 use it as a derived table. When collected parameters for Query 2 and Query 3, Query 1 should be included. Different parameter values could be selected for each of these queries.

When a custom process order is selected from the Process Custom dialog, all queries belonging to a given node of the processing tree (Queries, OLAPQueries, and Imported Files being the nodes) are arranged in order by the ordinal value of ProcessSequenceNum (Property), regardless of the setting of IncludeInProcessAll. Queries that are checked in that dialog box are processed in ProcessSequenceNum order, regardless of the Query type. For example, assume a document with Query2, OLAPQuery1, authorsout.csv, Query1, and OLAPQuery2, are assigned sequence numbers 1 to 5. They appear in the tree as Query2 and Query1, in that order under Queries, and OLAPQuery1 and OLAPQuery2 under the OLAP Queries branch, and authorsout.csv under Imported Files.

If a user checked all five and clicks OK, this is the process order

- Query2
- OLAPQuery1
- authorsout.csv
- Query1
- OLAPQuery2

If a query section has the IncludeInProcessAll (Property) set to false, clicking the *Process* button when viewing that section or any of its dependent sections or execution of the Process (Method) associated with that section still processes that query.

When a query section is duplicated, the new section gets the next highest ProcessSequenceNum, and it appears last in the Query Processing Order dialog list box if it is opened immediately following the duplicate section action. The IncludeInProcessAll boolean value is inherited from the source section during duplication.

Action:

Read-write, numeric

Example 1:

This example shows how to display the number of queries in the Interactive Reporting Reporting document (.bqy) file in an Alert box, set the processing the *Query* section to the second position in the Query Processing Order dialog, includes the *Query* section in a *Process All* command, and then execute the Process All command for the document:

```
Alert("Number of Query Sections "   +  ActiveDocument.Sections.QueryCount)
ActiveDocument.Sections["Query"].ProcessSequenceNum = 2
ActiveDocument.Sections["Query"].IncludeInProcessAll = true
ActiveDocument.ProcessAll()
```

Example 2:

This example shows how to set a query to process first in the query processing order:

```
var procSeqNum = Sections[“Query”].ProcessSequenceNum;

// Make this query get processed first
Sections[“Query”].ProcessSequenceNum = 1;
```

Prompt (Property)

Applies To:

Limit object

Description:

Returns or sets the value of the text displayed on the Limit dialog box.

Action:

Read-write, String

Example:

This example shows how to change the text displayed in a variable limit:

```
var MyLimit = ActiveDocument.Sections"Query"].Limits["State"]
MyLimit.VariableLimit = true
MyLimit.Prompt = "Please select a state from the list box below."
```

PromptToSave (Property)

Applies To:

Document object

Description:

Sets the Boolean value for displaying a prompt to save dialog box when an user exists an application, If the value of this property is set to true, a prompt to save dialog box is displayed when the user exits the application. If the value of this property is set to false, then no prompt to save dialog box is displayed when the user exits the application.

These are the truth tables for this property:

Table 2 PromptToSave Truth Table for an Interactive Reporting Document Deployed in the EPM Workspace

App	Y	N
Doc		
Y	P	D
N	P	D

In other words, for an document, the Prompt to Save option is determined at the application level and not the document level. In this case, the Prompt to Save Unsaved Files option on the Preferences dialog determines if the user is prompted to save files when exiting the EPM Workspace. For an Interactive Reporting document, the "Remind To Save" setting on the Options dialog box determines whether the prompt occurs or not.

Action:

Read-write, Boolean

Example:

This example shows how to set display the prompt to save property to true for the document:

```
Documents["Sample 1.bqy"].PromptToSave = true
```

QueryCount (Property)

Applies To:

Section object

Description:

Identifies the count of all query sections in an IInteractive Reporting document file (.bqy). This count includes relational, OLAP and processable imported data file sections. The count does not include Data Model sections.

This property may be used on association with the "IncludeInProcessAll (Property)" on page 118 and "ProcessSequenceNum (Property) " on page 158.

If you use the ProcessSequenceNumber (Property) to assign a query's position on the Query Processing Order dialog and the value is higher than the count of query sections in the document, it is silently reset to be equal to the count assigned by the "QueryCount (Property)" on page 161.

Action:

Read only

Example:

This example shows how to display the number of queries in the IInteractive Reporting document file (.bqy) in an Alert box, set the processing the *Query* section to the second position in the Query Processing Order dialog, includes the *Query* section in a Process All command, and then execute the Process All command for the document:

```
Alert("Number of Query Sections "   +  ActiveDocument.Sections.QueryCount)
ActiveDocument.Sections["Query"].ProcessSequenceNum = 2
ActiveDocument.Sections["Query"].IncludeInProcessAll = true
ActiveDocument.ProcessAll()
```

QueryInProcess (Property)

Applies To:

Document object

Description:

Identifies the name of the query being processed. This property is only appropriate for use in the OnPreProcess() and OnPostProcess() events.

Action:

Read-only, String

Example:

This example shows how to display the name of the query being processed in an Alert box:

```
Console.Writeln("Start OnPreProcess")
switch(ActiveDocument.QueryInProcess)
{
case "Query":
Alert("Query");
break;
 case "Query2":
 Alert("Query2");
break;
case "OLAPQuery":
```

```
Alert("OLAPQuery");
break;
default: Alert("Default");
break;
}
```

QuerySize (Property)

Applies To:

QuerySection object

Description:

Returns the estimated number of rows the current query will return if processed.

Action:

Read-only, Integer

Example:

This example shows how to check the size of the query before processing and ask the user if they want to process the query given the size:

```
var MyCon = ActiveDocument.Sections"Query"].DataModel.Connection
MyCon.Username = "Hyperion"
MyCon.SetPassword("HyperionHyperion")
MyCon.Connect()
var QS = ActiveDocument.Sections["Query].QuerySize
if (QS > 5000)
{
      var Msg = "The query you are about to run, returns "+QS+ rows. "Are you sure you
want to continue?"
      var retVal = Alert(Msg,Alert,Yes,No)
      if (retVal == 1)
            ActiveDocument.Sections["Query"].Process()
}
```

RefreshData (Property)

Applies To:

PivotSection object, ChartSection object

Description:

Sets a separate refresh frequency for each Pivot and Chart in a document. When the Query is processed, Reports are populated with data according to their refresh frequencies. There are three methods available for refreshing reports: After Process, OnActivate and Manually. These

options are mutually exclusive. An additional option, the RefreshDataNow (Method). This method is only available when *Manually* is the selected option.

Note: Refresh options are set on a per-report basis. For example, to refresh ten Pivot reports when they are activated, set the When Section Displayed option for each report.

Action:

Read-Write

Constants:

The BqRefreshData constant group consists of these values:

- bqRefreshDataAfterProcess
- bqRefreshDataManually
- bqRefreshDataOnActivate

Example 1:

In this example, a request is made to manually refresh the Pivot section, after which an immediate refresh to the current section is invoked:

```
//Manual Refresh of Data
ActiveDocument.Sections["Pivot"].RefreshData=bqRefreshDataManually
ActiveDocument.Sections["Pivot"].RefreshDataNow()
```

Example 2:

In this example, a request is made to establish an automatic link to the Results section to update the report whenever the query is processed:

```
//Refresh Data After Processing
ActiveDocument.Sections["Pivot"].RefreshData=bqRefreshDataAfterProcess
```

Example 3:

In this example, a request is made to refresh when the section is accessed and displayed:

```
//Refresh Data When Section is Displayed
ActiveDocument.Sections["Pivot"].RefreshData=bqRefreshDataOnActivate
```

RemoveUnselectedGroup (Property)

Applies To:

QueryOptions object

Description:

Removes all dimension groups that are not in the selected group.

Action:

Read-write, Boolean

Example:

This example shows how to enable the RemoveUnselectedGroup (Property) when a check box is checked:

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox9"].Checked)
    {ActiveDocument.Sections["Query"].QueryOptions.RemoveUnselectedGroup = true}
else
{ActiveDocActiveDocument.Sections["Query"].QueryOptions.RemoveUnselectedGroupument.Secti
ons["Query"].QueryOptions.RemoveUnselectedGroup = false}
```

ReplaceMissing (Property)

Applies To:

QueryOptions object

Description:

Sets or returns a label for missing values.

Action:

Read-write, String

Example:

This example shows how to set the label for missing values to “missing”:

```
ActiveDocument.Sections["Query"].QueryOptions.ReplaceMissing = "missing"
```

ReplaceNoAccess (Property)

Applies To:

QueryOptions object

Description:

Sets or returns a label for values from the Essbase cube to which a user does not have security access.

Action:

Read-write, String

Example:

This example shows how to set the label for no access items to “No Access”:

```
ActiveDocument.Sections["Query"].QueryOptions.ReplaceNoAccess = "No Access"
```

ReplaceZeros (Property)

Applies To:

QueryOptions object

Description:

Sets or returns a label for zero values.

Action:

Read-write, String

Example:

This example shows how to set the label for zero values to “0”.

```
ActiveDocument.Sections["Query"].QueryOptions.ReplaceZeros = "0"
```

Repository (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the Repository radio button setting on the Object tab of the Embedded Browser control and the HyperLink control. If this property is true, the Repository radio button has been checked.

Action:

Read only, Boolean

Example:

This example shows how to show if the Repository radio button has been checked in an Alert box:

```
Alert(EmbeddedBrowser1.Repository)
```

RepositoryBQYSection (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the section name set for the Interactive Reporting document (.bqy) repository object on the Document Options dialog. The default section name matches the Interactive Reporting document (.bqy) = default (that is, whatever section would display if the user opened the Interactive Reporting document file (.bqy) using the EPM Workspace icon in the Browse application).

Action:

Read only, String

Example:

This example shows how to display the name of the IInteractive Reporting document file (.bqy) section in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryBQYSection)
```

RepositoryBQYToolbarType (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the Interactive Reportingdocument file (.bqy) toolbar type set for a Interactive Reporting document (.bqy) repository object on the Document Options dialog (corresponds to the Toolbar field). The toolbar type indicates which toolbar is associated with the Interactive Reporting document file (.bqy) when it is viewed in the EPM Workspace

Action:

Read only, BqRepositoryBQYToolbarType

Constants:

The BqRepositoryBQYToolbarType constant group consists of these values:

- bqRepositoryBQYToolbarNone = 0
- bqRepositoryBQYToolbarPaging = 1
- bqRepositoryBQYToolbarStandard =2

Example:

This example shows how to display the type of toolbar shown for the Interactive Reporting document (.bqy) repository object in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryBQYToolbarType)
```

RepositoryDocument (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the name of the object selected from the Repository on the Document Options dialog.

Action:

Read only, String

Example:

This example shows how to display the name of the object selected from the Repository in an Alert box (only the object name displays, and not the path name):

```
Alert(EmbeddedBrowser1.RepositoryDocument)
```

RepositoryFileType (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the type of file associated with the repository object on the Document Options dialog. There are six repository file types for external content in the EPM Workspace, including:

- Interactive Reporting document (.bqy)
- Interactive Reporting document Job
- Oracle's Hyperion® SQR® Production Reporting Job
- Oracle's Hyperion® Web Analysis
- Oracle Hyperion Financial Reporting, Fusion Edition
- Other

Action:

Read only, BqRepositoryFiletype constant group

Constants:

The BqRepositoryFileType constant group consists of these values:

- bqRepositoryFileTypeAnalyzer = 5
- bqRepositoryFiletypeBQY = 2
- bqRepositoryFiletypeBQYJob = 3
- bqRepositoryFiletypeOther = 1
- bqRepositoryFileTypeReport = 6
- bqRepositoryFiletypeSQRJob = 4

Example:

This example shows how to display the file type of the repository object in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryFiletype)
```

RepositoryJobFilename (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the name of the job file displayed from a job output repository object. The job file name represents theInteractive Reporting or Production Reporting job to be viewed in the EPM Workspace. This property is only available when the Document Type is an Interactive Reporting or a Production Reporting Job on the Document Options dialog.

Action:

Read only, String

Example:

This example shows how to display the job file name of the repository object in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryJobFilename)
```

RepositoryJobRun (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the value of the Run Job value on the Document Options dialog. This property is only available for Interactive Reporting Job or Oracle's Hyperion® SQR® Production Reporting Job document types.

Action:

Read only, Boolean

Example:

This example shows how to return the value in Run Job check box in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryJobRun)
```

RepositoryParams (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns any parameters appended to the URL (used as a Smartcut for accessing the Interactive Reporting document file (.bqy) in the EPM Workspace) through the *Other Parameters* control of the Document options dialog.

Action:

Read only, String

Example:

This example shows how to return additional parameters appended to the URL in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryParams)
```

RepositoryReportsDisplayFormat (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the display format to used for a Hyperion Reports document embedded or hyperlinked in the Dashboards section. Valid options are PDF format or HTML format.

Action:

Read only, BqExportFileFormat

Constants:

The BqExportFileFormat constant group contains two values available with the RepositoryReportsDisplayFormat (Property): bqExportFileFormatHTML (1) and bqExportFormatPDF (8). If other values are selected from this constant group, an exception is thrown.

Example:

This example shows how to return additional parameters appended to the URL in an Alert box:

```
Alert(EmbeddedBrowser1.RepositoryReportsDisplayFormat = bqExportFormatPDF)
```

RepositorySmartcut (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the Smartcut generated using the Repository options on the Document Options dialog. A SmartCut is a link in the form of a special URL to an item in the Hyperion Foundation Repository.

Action:

Read only, String

Example:

This example shows how to return the Smartcut path generated for the embedded repository object in an Alert box:

```
Alert(EmbeddedBrowser1.RepositorySmartcut)
```

RepositorySmartcutParams (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns any additional parameters specified in the Smartcut Parameters edit box on the Document Options dialog. When multiple parameters exist, a single string is returned with

parameter values delimited by an ampersand (&). A SmartCut is a link in a URL to an item in the repository.

Action:

Read only, String

Example:

This example shows how to return additional Smartcut parameter generated for embedded repository objects in an Alert box:

```
Alert(EmbeddedBrowser1.RepositorySmartcutParams)
```

RepositoryToolbarType (Property)

Applies To:

EmbeddedBrowser object, HyperLink object

Description:

Returns the type of toolbar displayed with an embedded browser object or hyperlink object . The values available for this property are none, paging (Interactive Reporting document file (.bqy) or I (Interactive Reporting Job only) or standard.

If you are working with an Interactive Reporting document file (.bqy) or Interactive Reporting (Job and the paging toolbar type is returned, then this property represents an abbreviated version of the Hyperion System 9 BI + Standard toolbar and consists of the section paging controls (First Page, Previous Page, Next Page, Last Page) and the current page indicator text (Page X of Y). If you try to display the Paging Toolbar in Interactive Reporting Studio or Interactive Reporting Web Client, the scrip command is ignore, no exception is thrown, and the script continues.

In a Oracle's Hyperion Reporting and Analysis document in which the standard toolbar type is returned, this property represents the third Web Analysis toolbar or the Point of View (POV) toolbar. The first and second Web Analysis toolbars (Help/Preferences/LogOff/Help and Repository->MyReport) are hidden for Reports populating a Interactive Reporting document (.bqy) external content control. The third Web Analysistoolbar, Point of View (POV), is an optional Dashboard display, and is hidden by default.

The HTML/PDF radio buttons display in Web Analysis based on the third POV toolbar setting. If the POV toolbar is shown, the HTML/PDF radio buttons are shown. Otherwise, they are hidden.

In a Web Analysis document, you can use the second and third Oracle's Hyperion® Web Analysis toolbars in the Dashboard section by choosing the Standard toolbar. However, both toolbars are treated as a single unit. Either both are shown, or hidden.

Action:

Read only, BqRepositoryToolbarType

Constants:

The BqRepositoryToolbarType constant group contains these values:

- bqRepositoryToolbarNavigation = 3
- bqRepositoryToolbarPaging = 2
- bqRepositoryToolbarPagingNavigation = 4
- bqRepositoryToolbarStandard =1

Example:

This example shows how to return the standard toolbar type for the embedded object:

```
EmbeddedBrowser1.RepositoryToolbarType = bqRepositoryToolbarStandard
```

ResetDefaultSortOrderLang (Property)

Applies To:

Application object

Description:

Clears the current sort order language. This property is equivalent to disabling the Use this choice option on the International tab of the Properties dialog box.

Example:

This example shows how to use the ResetDefaultSortOrderLang (Method) to clear the default sort order language.

```
//Clear all textboxes except TextBox5
TextBox1.Text = ""
TextBox1.Text = ""
TextBox1.Text = ""
TextBox4.Text = ActiveSection.Name
var ActionName = "ResetDefaultSortOrderLang new"
TextBox1.Text ="Start " + ActionName
try
{
Application.ResetDefaultSortOrderLang()
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
TextBox1.Text ="End " + ActionName
```

ResetPrintProperties (Property)

Applies To:

Application object

Description:

Enables you to use custom or default document print settings. When false (default), the original default print settings are used for all document sections. When true, the current default print settings are used.

Note: Unexpected print behavior may occur when this option is enabled in the user interface and disabled through the object model in a document OnStartup script.

Action:

Read-write, Boolean

Example:

This example shows how to set the SetPrintProperties to true:

```
Application.ResetPrintProperties=true
```

RightMargin (Property)

Applies To:

ReportSection object

Description:

Sets the right margin of the report. Margins are set for the entire report.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true, use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Number

Example:

This example shows how to set the right margin of the report to .25 inches:

```
ActiveDocument.Sections["Report"].RightMargin = .25
```

Rotation (Property)

Applies To:

PieChart object

Description:

Returns or sets the rotation value of a Pie Charts to change the visual perspective.

Action:

Read-write, Numeric

Example:

This example shows how to change the rotation of a pie chart:

```
ActiveDocument.Sections["AllChart"].PieChart.Rotation=45
```

RowCount (Property)

Applies To:

ResultsSection object, TableSection object

Description:

Returns the number of rows in a Results or Table section.

Note: Use the "QuerySize (Property)" on page 163 to determine the number of rows returned by queries.

Action:

Read-only, Integer

Example:

This example shows how to transfer a list of values from a table column to a list box in a Dashboard section:

```
var RC =  ActiveDocument.Sections["Table"].RowCount
  for ( j = 1; j <= RC ; j++)
  {
     var MyVal = ActiveDocument.Sections["Table"].Column["State"].GetCell(j)
     ActiveDocument.Sections["Dashboard"].Shapes["ListBox1"].Add(MyVal)
  }
```

RowLimit (Property)

Applies To:

QuerySection object, DataModel object

Description:

Defines the maximum number of rows retrieved by queries against the Data Model. Use this property with "RowLimitActive (Property)" on page 176 to define the maximum number of rows to retrieve. This property corresponds to the Rows field on the General tab of the Data Model Options dialog.

Action:

Read-Write, Number

Example:

This example shows how to define a 100 row limit and process the query:

```
ActiveDocument.Sections["Query2"].DataModel.RowLimitActive = true
ActiveDocument.Sections["Query2"].DataModel.RowLimit = 100
ActiveDocument.Sections["Query2"].Process()
```

RowLimitActive (Property)

Applies To:

QuerySection object, DataModelSection object

Description:

Enables you to activate the query governor feature when retrieving rows against a Data Model. Use this property with "RowLimit (Property)" on page 176 to specify the number of rows to retrieve.

Explorer and Interactive Reporting Web Client users can set a query governor, but Data Model options override those set at the query level. If row limits are set at the query level, the lower number is enforced.

Action:

Read-only, Boolean

Example:

This example enables the Row Filter setting, sets the maximum number of rows to retrieve, and processes the query:

```
ActiveDocument.Sections["Query2"].DataModel.RowLimitActive = true
ActiveDocument.Sections["Query2"].DataModel.RowLimit = 200
```

```
ActiveDocument.Sections["Query2"].Process()
```

RowNumber (Property)

Applies To:

ResultsSection object, TableSection object

Description:

Returns the selected row in a Results/Table section. You can call Renumber (Property) from the OnRowDoubleClick event any other event, including those in the Dashboard section, Startup/Shutdown, and Custom Menu items.

RowNumber (Property) is determined by the row selected in the Row/Table section. This property applies to a Results/Table section that is actively embedded in an Dashboard section when you select a row from the embedded Results/Table. Selecting a Results/Table section sets the RowNumber (Property) to a number that represents the nth row in the section. If a row is not selected, this property resets to zero.

Action:

Read-only, Number

Example:

This example shows how to display the row number:

```
Alert (ActiveDocument.Sections["Results"].RowNumber)
```

SaveResults (Property)

Applies To:

QuerySection object

Description:

Returns or sets the value of the Save Results with document option. To save the Results set of a query and computed columns as a snapshot in a document, set this property to true.

Save the Results set with a query to analyze and generate reports without having a database connection. Results are saved for any queries for which results exist.

Note: Saving Results in documents is performed on a query-by-query basis.

Action:

Read-write, Boolean

Example:

This example shows how to save the results with a sample Query section, *SalesQuery*:

ActiveDocument.Sections["SalesQuery"].SaveResults=true

SaveWithoutUsername (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables you not to save the database user name with the Interactive Reporting database connection file (.oce).

Action:

Read-write, Boolean

Example:

This example creates a connection file and applies it to the current document. The data source name is *PlutoSQLSVR*, which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.EnableAsyncProcess = true
myCon.SaveWithoutUsername = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

ScaleMax (Property)

Applies To:

LeftAxis object, RightAxis object, FactsAxis object

Description:

Returns or sets the maximum Chart scale values for the right, left or facts axis.

Action:

Read-write, Numeric

Example:

This example shows how to change the maximum scale of left and right chart axes:

```
ActiveDocument.Sections["AllChart"].ValuesAxis.LeftAxis.ScaleMax=2000000
ActiveDocument.Sections["AllChart"].ValuesAxis.RightAxis.ScaleMax=2000000
```

ScaleMin (Property)

Applies To:

LeftAxis object, RightAxis object

Description:

Returns or sets the minimum scale values for the right or left Chart axes.

Action:

Read-write, Numeric

Example:

This example shows how to change the minimum scale of a left and right Chart axes:

```
var MyChart = ActiveDocument.Sections["Chart"]
MyChart.ValuesAxis.LeftAxis.ScaleMin  = 25
MyChart.ValuesAxis.RightAxis.ScaleMin  = 25
```

ScaleX (Property)

Applies To:

Picture object

Description:

Sets the horizontal scale of a picture object.

This property corresponds to the Percent Scale Width field on the Picture Properties dialog.

Action:

Read-write, Numeric

Example:

This example shows how to reduce the width of the picture by 50%:

```
ActiveDocument.Sections["Report"].Body.Shapes["Picture"].ScaleX = 50
```

ScaleY (Property)

Applies To:

Picture object

Description:

Sets the vertical scale of a picture object.

This property corresponds to the Percent Scale Height field on the Picture Properties screen.

Action:

Read-write, Numeric

Example:

This example shows how to increase the width of the picture by 50%:

```
ActiveDocument.Sections["Report"].Body.Shapes["Picture"].ScaleY = 150
```

Script (Property)

Applies To:

EventScript object

Description:

The Script property is a read/write string containing the event handler's JavaScript source code, as it is seen in the Script editing dialog.

You have to alternate single and double quotes to avoid problems with string parsing, for example, write:

```
 'Console.Writeln("Step 1")'
or
"Console.Writeln('Step 1')"
```

and not,

```
"Console.Writeln("Step 1")"
or
'Console.Writeln('Step 1')'
```

Action:

Read-write, String

Example:

This example shows how to create a command button named "cbcreated" on an OnClick event. The script also writes the results of the script to the Console Window:

```
Console.Writeln("Start Dynamically Add CommandButton")

Console.Writeln("Step1")
var oCBShape = ActiveDocument.Sections["Dashboard"].Shapes.CreateShape(bqButton)

Console.Writeln("Step2")
oCBShape.Name = "CBCreated"
oCBShape.Text = "CBCreated"
oCBShape.Visible = true
oCBShape.Enabled = true
oCBShape.Locked = false
//Font color not enable for CommandButtons
//oCBShape.Font.Color = bqBlue
oCBShape.Font.Effect = bqFontEffectUnderline
oCBShape.Font.Name = "Ariel"
oCBShape.Font.Size = "12"
oCBShape.Font.Style = bqFontStyleItalic

Console.Writeln("Step3")

Console.Writeln("Width: " + oCBShape.Placement.Width)
Console.Writeln("Height: " + oCBShape.Placement.Height)

oCBShape.Placement.Modify(25, 25, oCBShape.Placement.Width, oCBShape.Placement.Height)

Console.Writeln("Step4")

if (Application.Type != bqAppTypeThinClient)
{
oCBShape.EventScripts["OnClick"].Script = 'Console.Writeln("OnClick")'
oCBShape.EventScripts["OnClientClick"].Script = 'Console.Writeln("OnClientClick")'
}
else
{
oCBShape.EventScripts["OnClick"].Script = 'Console.Writeln("OnClick")'

oCBShape.EventScripts["OnClientClick"].Script = 'alert("OnClientClick")'
}

Console.Writeln("Step5")

oCBShape.OnClick()
oCBShape.OnClientClick()

Console.Writeln("End Add CommandButton")
```

Scrollable (Property)

Applies To:

ControlsTextBox object

Description:

Returns or sets the value of the textbox scrollable property. Setting this property to "true" enables vertical scrolling of text in the Text box control.

Action:

Read-write, Boolean

Example:

This example shows how to change the properties of a text box:

```
ActiveDocument.Sections["Dashboard"].Shapes["TextBox1"].Scrollable = true
```

ScrollbarsAlwaysShown (Property)

Applies To:

DashboardSection object

Description:

Enables you to always display scrollbars for embedded section objects. This property does not apply to hyperlinked embedded section objects or view-only embedded sections with auto-sizing enabled.

The default setting, show scrollbars after the embedded section is selected, is false.

Action:

Read-write, Boolean

Example:

This example shows how to enable embedded section objects to always show scrollbars:

```
ActiveDocument.Sections["Dashboard"].Shapes["Chart1"].ScrollbarsAlwaysShown = true
```

SelectedIndex (Property)

Applies To:

ControlsDropDown object, Slider object

Description:

Returns or sets the selections index in a drop-down control. Setting this value causes the dropdown to change its selection.

Action:

Read-write, Integer

Example:

This example shows how to display the number of the selected item in an Alert dialog box:

```
Index=ActiveDocument.Sections["Dashboard2"].Shapes["DropDown1"].SelectedIndex=3
Alert("The user selected " + String(Index))
```

SelectedTheme (Property)

Applies To:

Themes collection

Description:

Returns or sets the theme type of a gauge. The theme refers to the parts of the gauge that can be altered to change the look of the gauge without changing its functionality. The standard Interactive Reporting theme types are either realistic (three dimensional appearance) or simplistic (one dimensional appearance).

Action:

Read-write, String

Example:

This example shows how to change the theme type of a speedometer from simplistic to realistic:

```
ActiveDocument.Sections["Dashboard"].Shapes["Speedometer"].Theme.SelectedTheme =
"realistic"
```

SelectedValue (Property)

Applies To:

Slider object

Description:

Returns the value selected by dragging or selecting the slider handle. The value returned is a string representing VALUE C. The Default value of the property is VALUE X.

Action:

Read only, String

Example:

This example shows how to read the value selected by moving the slider handle and writes it to an Alert box:

```
Alert(Slider.SelectedValue)
```

ServerAddress (Property)

Applies To:

Document object

Description:

Sets the server address of an Interactive Reporting document.

Action:

Read-write, String

Example:

This example shows how to set the server address for the Sample 1.bqy:

```
Documents["Sample 1.bqy"].ServerAddress = " http://edsel:19000/workspace"
```

Shadow (Property)

Applies To:

Picture object

Description:

Displays shadows for lines or shapes so that these objects appear three-dimensional.

This property corresponds to the Shadow field on the Borders and Background screen in the user interface.

Action:

Read-write, Boolean

Example:

This example shows how to set the shadow property to the picture object:

```
ActiveDocument.Sections["Report"].Body.Shapes["Picture"].Shadow = true
```

ShiftPoints (Property)

Applies To:

BarLineChart object

Description:

Determines where Line Chart plot points are placed in relation to the bar.

Action:

Read-write, BqBarLineShift constant group

Constants:

The BqBarLineShift constant group consists of the bqShiftCenter and bqShiftLeft values.

Example:

This example shows how to change a Bar Line charts shift points:

```
ActiveDocument.Sections["AllChart"].BarLineChart.ShiftPoints=bqShiftLeft
```

Show3DObjects (Property)

Applies To:

ChartSection object

Description:

Displays Charts as 3D objects. When disabled, Charts are displayed as 2D objects.

Action:

Read-write, Boolean

Example:

This example shows how to change a chart to display 3D objects:

```
ActiveDocument.Sections"Chart"].Show3DObjects = true
```

ShowAdvanced (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables the display of the Show Advanced Properties dialog in the OCE wizard.

Action:

Read-write, Boolean

Example:

This example shows how to set the ShowAdvanced (Property):

```
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
ActiveDocument.Sections["Query"].DataModel.Connection.ShowAdvanced = true
ActiveDocument.Sections["Query"].DataModel.Connection.Save()
```

ShowAllPositive (Property)

Applies To:

PieChart object

Description:

Enables the display of all values (both positive and negative) as positive when displaying a Pie Chart.

Action:

Read-write, Boolean

Example:

This example shows how to display all the values as positive values in a Pie Chart:

```
var MyChart = ActiveDocument.Sections["Sales Pie Chart"]
MyChart.PieChart.ShowAllPositive = true
```

ShowBackPlane (Property)

Applies To:

ChartSection object

Description:

Displays the back plane of Charts.

Action:

Read-write, Boolean

Example:

This example shows how to display the back plane in a Chart section:

```
var MyChart = ActiveDocument.Sections["Sales Chart"]
MyChart.ShowBackPlane = true
```

ShowBarOutline (Property)

Applies To:

BarChart object

Description:

Displays bar borders.

Action:

Read-write, Boolean

Example:

This example show to display the border of the bar.

```
ActiveDocument.Sections["Chart"].BarChart.ShowBarOutline = true
```

ShowBarValues (Property)

Applies To:

BarChart object, BarLineChart object

Description:

Display data values on the tops of individual bars in Bar and Bar Line Charts.

Action:

Read-write, Boolean

Example:

This example shows how to display the true data values on top of the bars in Bar and Bar line charts:

```
var MyChart = ActiveDocument.Sections["AllChart"]
MyChart.BarChart.ShowBarValues = true
```

ShowBorder (Property)

Applies To:

ChartSection object

Description:

Enables a border to be displayed around a Chart.

Action:

Read-write, Boolean

Example:

This example shows how to display the Chart border:

```
var MyChart = ActiveDocument.Sections["Sales Chart"]
MyChart.ShowBorder = true
```

ShowBrioRepositoryTables (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Displays the Document Repository Tables in the Table Catalog associated with Interactive Reporting database connection files (.oce).

Action:

Read-write, Boolean

Example:

This example shows how to create an Interactive Reporting database connection file and applies it to the current Interactive Reporting document. The data source name in this example is *PlutoSQLSVR*, which is a user DSN using the SQL Server 6.5 driver:

```
Var myCon = Application.CreateConnection()
MyCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
MyCon.HostName ="PlutoSQLSVR"
MyCon.EnableAsyncProcess = true
MyCon.ShowBrioRepositoryTables = true
MyCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
```

```
("d:\\OCEs\\PlutoSQL.oce")
```

ShowCatalog (Property)

Applies To:

Document object, PluginDocument object

Description:

Displays the Section/Catalog pane. This property is the equivalent to selecting/deselecting the Section/Catalog item from the View menu.

Note: Do not use ActiveDocument.ShowCatalog in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to hide and show various user interface elements in Interactive Reporting based on the application they are running:

```
if (Application.Name == "BrioQuery")
{
   ActiveDocument.ShowCatalog = true
   ActiveDocument.ShowMenuBar = true
}
else
{
//Save space in plugin by hiding catalog and turning off menu bar
   ActiveDocument.ShowCatalog = false
   Application.ShowMenuBar = false
}
```

ShowColumnTitles (Property)

Applies To:

ReportTable object

Description:

Displays the Table column titles.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true, use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Boolean

Example:

This example shows how to hide table column titles:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].ShowColumnTitles = false
```

ShowColumnTotal (Property)

Applies To:

TableFact object

Description:

Displays column total (break total) on a Table Fact column in the Report section.

Note: When using this property and the "SuspendCalculation (Property)" on page 226 is true, use the Recalculate (Method) to recalculate the Report section.

Action:

Read-write, Boolean

Example:

This example shows how to display the column total for the *Amount Sales* Table column:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts["Amount
Sales"].ShowColumnTotal = true
```

ShowCustomMenu (Property)

Applies To:

Application object

Description:

Manages custom menu display. This property enables users to disable the Custom menu option during security-based processing in OnPreProcess events. The Boolean value specified is used for as long as theInteractive Reporting Studio runs. For Interactive Reporting Web Client users,

the application restarts when new documents are opened. Consequently, write a script for every document for which you must disable Custom menu options.

Action:

Read-write, Boolean

Example:

This example shows how to display the column total for the *Amount Sales* column:

```
Application.ShowCustomMenu = false;
```

ShowDerivableQueries (Property)

Applies To:

DMCatalog object

Description:

Displays derivable queries in the Catalog pane. A derived table is a statement-local temporary table created by a sub-query in the FROM clause of a SQL SELECT statement. It exists only in memory and behaves like a standard view or table.

Action:

Read-write, Boolean

Example:

This example shows how to display derivable queries:

```
Documents["Mapping.bqy"].Sections["Query"].DataModel.Catalog.ShowDerivableQueries = true
```

ShowDrillpathInLabels (Property)

Applies To:

XAxisLabel object, ZAxisLabel object

Description:

Displays the drill path value in the label and legend.

Action:

Read-write, Boolean

Example:

This example shows how to not show the drill path value in the label and legend:

```
ActiveDocument.Sections["BooksChart"].LabelsAxis.XAxis.ShowDrillpathInLabels=false
```

ShowFilter (Property)

Applies To:

OLAPFilter object

Description:

Sets or returns whether the Filters pane is visible in the data layout.

Action:

Read-write, Boolean

Example:

This example shows how to read and write the ShowFilter (Property)

```
if (Sections[“Query”].ShowFilters == true)
    // The filters pane is visible

// Hide the filters pane
Sections[“Query”].ShowFilters = false;
```

ShowFullNames (Property)

Applies To:

DMCatalog object

Description:

Displays the full names of tables in the Table Catalog.

Action:

Read-write, Boolean

Example:

This example shows how to display the full names of tables in a Table Catalog:

```
var myQuery = ActiveDocument.Sections["Query"]
myQuery.DataModel.Catalog.ShowFullNames = true
```

ShowHorizontalPlane (Property)

Applies To:

ChartSection object

Description:

Displays he horizontal plane of a Chart.

Action:

Read-write, Boolean

Example:

This example shows how to display the Chart border:

```
var MyChart = ActiveDocument.Sections"Sales Chart"]
MyChart.ShowBorder = true
MyChart.ShowHorizontalPlane = true
```

ShowIconJoins (Property)

Applies To:

DataModel object

Description:

Displays joins between topics that were made into icons in the Data Model.

Action:

Read-write, Boolean

Example:

This example shows how to display icon joins in a Data Model:

```
ActiveDocument.Sections["Query"].DataModel.ShowIconJoins = true
```

ShowInLegend (Property)

Applies To:

TrendLine object

Description:

Enable trend line information in the Chart legend

Action:

Read-write, Boolean

Example:

This example show how to display trend line information in the Chart legend:

```
ActiveDocument.Sections["Chart"].TrendLines[1].ShowInLegend = true
```

ShowIntervalTickmarks (Property)

Applies To:

ValuesAxis object

Description:

Displays tick marks on a Chart Values axis.

Action:

Read-write, Boolean

Example:

This example shows how to display interval tick marks in a chart:

```
ActiveDocument.Sections["Chart"].ValuesAxis.ShowIntervalTickmarks = true
```

ShowIntervalValues (Property)

Applies To:

ValueAxis object

Description:

Enables the display of interval values on a Chart values axis.

Action:

Read-write, Boolean

Example:

This example shows how to enable interval values for a chart:

```
ActiveDocument.Sections"Chart"].ValuesAxis.ShowIntervalValues = true
```

ShowLabel (Property)

Applies To:

LeftAxis object, RightAxis object, XAxisLabel object, ZAxisLabel object, TrendLine object

Description:

Displays labels associated with an axis, and trend line object.

Action:

Read-write, Boolean

Example:

This example shows how to show all the labels for the various chart objects:

```
ActiveDocument.Sections["Chart"].Activate()
ActiveSection.ValuesAxis.RightAxis.ShowLabel = true
ActiveSection.LabelsAxis.XAxis.ShowLabel = true
ActiveSection.ValuesAxis.LeftAxis.ShowLabel = true
ActiveSection.LabelsAxis.ZAxis.ShowLabel = true
```

ShowLabels (Property)

Applies To:

PieChart object

Description:

Displays Pie Chart labels.

Action:

Read-write, Boolean

Example:

This example shows how to display Pie Chart labels and the percentages of value:

```
ActiveDocument.Sections["Chart"].PieChart.ShowLabels = true
ActiveDocument.Sections["Chart"].PieChart. ShowPercentages = true
```

ShowLegend (Property)

Applies To:

ChartSection object

Description:

Displays the Chart legend.

Action:

Read-write, Boolean

Example:

This example shows how to enable the chart legend:

```
ActiveDocument.Sections["Chart"].ShowLegend = true
```

ShowLevelProperties (Property)

Applies To:

OLAPConnection object (OLE DB only)

Description:

Enables you to show any description association with a level.

Action:

Read only, Boolean

Example:

This example shows how to display in a text box if level properties are shown in the query:

```
var xxx=ActiveDocument.Sections["OLAPQuery"].Connection.ShowLevelProperties
if (xxx==true)
{
TextBox1.Text = "Levels are shown"
}
else (xxx==false)
{
TextBox1.Text = "Levels are not shown"
}
```

ShowLocalResults (Property)

Applies To:

DMCatalog object

Description:

Displays local results in the Table Catalog.

Action:

Read-write, Boolean

Example:

This example shows how to search a document for multiple Results sets and display local results in the Table Catalog:

```
var ResultsCount = 0
for (j =1 ; j <= ActiveDocument.Sections.Count ; j++)
      if (ActiveDocument.Sections[j].Type == bqQuery)
                ResultsCount++
if (ResultsCount > 1 )
      ActiveDocument.Sections["Query"].DataModel.ShowLocalResults = true
```

ShowMarkerOutline (Property)

Applies To:

BarLineChart object

Description:

Displays the outline of a maker for a BarLineChart (object).

Action:

Read-write, Boolean

Example:

This example shows how to display the outline of a marker:

```
ActiveDocument.Sections["Chart2"].BarLineChart.ShowMarkerOutline = true
```

ShowMenuBar (Property)

Applies To:

Application object

Description:

Displays the Application Menu toolbar. The default value for this property is true.

Note: Do not use Application.ShowMenuBar object in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to hide and show various user interface elements in Interactive Reporting based on the application they are running:

```
if (Application.Name == "BrioQuery Designer")
{
   ActiveDocument.ShowCatalog = true
 Application.ShowMenuBar = true
}
else
{
//Save space in plugin by hiding catalog and turning off menu bar
   ActiveDocument.ShowCatalog = false
   Application.ShowMenuBar = false
}
```

ShowMetadata (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables the display of metadata settings in the Open Catalog Extensions Wizard.

Action:

Read-write, Boolean

Example:

This example creates an Interactive Reporting database connection file and applies it to the current document. The data source name in this example is *PlutoSQLSVR*, which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
myCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
mmyCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.EnableAsyncProcess = true
myCon.ShowMetaData = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

ShowNegativeValues (Property)

Applies To:

BubbleChart object

Description:

Displays bubbles with negative values. These type of values are derived from their real absolute value, and the real negative value is depicted in the data label (although based on the positive value).

Action:

Read-write, Boolean

Example:

This example shows how to display negative values in a bubble chart:

```
ActiveDocument.Sections["Chart2"].BubbleChart.ShowNegativeValues = true
```

ShowOutline (Property)

Applies To:

Legend object

Description:

Displays the border of the chart legend.

Example:

This example shows how to display the border of the legend:

```
ActiveDocument.Sections["Chart2"].Legend.ShowOutline = true
```

ShowOutliner (Property)

Applies To:

ChartSection object, (CubeQuery)QuerySection object, OLAPQuerySection object, PivotSection object, QuerySection object, ResultsSection, TableSection object, ReportSection object

Description:

Enables the display of the Outliner association with section.

Note: Do not use Application.ShowOutliner (Property) in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to display the Chart Outliner:

```
ActiveDocument.Sections["Chart"].ShowOutliner = true
```

ShowPartialViewIndicator (Property)

Applies To:

ChartSection object

Description:

Displays the text "Partial View" as an indicator in the top left corner of the Contents pane if not all bars of a chart can be displayed in one view. You can show or hide this indicator on a per chart basis by enabling or disabling this property. This property corresponds to the "Show partial view indicator" field on the General tab of Chart Properties.=

Action:

Read-write, Boolean

Example:

This example shows how to hide the Partial View Indicator on the Chart:

```
ActiveDocument.Sections["Chart"].ShowPartialViewIndicator = false
```

ShowPercentages (Property)

Applies To:

PieChart object

Description:

Enables the display of Pie slice percentages in a Pie Chart.

Action:

Read-write, Boolean

Example:

This example shows how to set Pie Chart specific properties:

```
ActiveDocument.Sections["Chart"].PieChart.ShowLabels = true
ActiveDocument.Sections["Chart"].PieChart. ShowPercentages = true
```

ShowPieOutline (Property)

Applies To:

PieChart object

Description:

Displays the border around the entire pie chart.

Action:

Read-write, Boolean

Example:

This example shows how to sets the ShowPieOutline (Property) to true:

```
ActiveDocument.Sections["AllChart"].PieChart.ShowPieOutline = true
```

ShowRowNumbers (Property)

Applies To:

ResultsSection object, TableSection object

Description:

Enables the display of row numbers in the left most region of a Table Section.

Action:

Read-write, Boolean

Example:

This example displays the row numbers:

```
ActiveDocument.Sections"Results"].ShowRowNumbers = true
```

ShowSectionTitleBar (Property)

Applies To:

Document object, PluginDocument object

Description:

Displays a section specific title bar. Changing this property is equivalent to showing/hiding the section title bar from the View menu.

Note: Do not use ActiveDocument.ShowSectionTitleBar inInteractive Reporting document file (.bqy) to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to hide and show various user interface elements inInteractive Reporting based on the application you are running:

```
if (Application.Name == "BrioQuery Designer")
{
   ActiveDocument.ShowCatalog = true
    ActiveDocument.ShowSectionTitleBar = true
    Application.ShowStatusBar = true
   Application.ShowMenuBar = true
}
else
{
//Save space in plugin by turning off various user interface elements
   ActiveDocument.ShowCatalog = false
   ActiveDocument.ShowSectionTitleBar = false
    Application.ShowStatusBar = false
   Application.ShowMenuBar = false
}
```

ShowScrollbar (Property)

Applies To:

Chart Embedded Object, EmbeddedBrowser object

Description:

Sets the scroll bar option to indicate when scroll bars are displayed for an Embedded Content control.

Available scroll bar options are:

- Always
- Never
- Automatic

Automatic is the default scroll bar option.

Action:

Read write, BqScrollbar constant group

Constants:

The BqScrollbar constant group consists of these values:

- bqScrollbarTypeAlways
- bqScrollbarTypeNever
- bqScrollbarTypeAutomatic

Example:

This example shows how to always display the Scrollbar for an embedded browser control:

```
EmbeddedBrowser1.ShowScrollbar = bqScrollbarTypeAlways
```

ShowSlicer (Property)

Applies To:

OLAPQuerySection object

Description:

Displays the slicer in the OLAPQuery Outliner. The slicer defines a logical slice of the server cube by instructing the server to ignore all values not part of the slice. The default value of this property is true.

Action:

Read-write, Boolean

Example:

This example shows how to hide the slicer in an OLAPQuery:

```
ActiveDocument.Sections["OLAPQuery"].ShowSlicer=false
```

ShowSortLine (Property)

Applies To:

Section object

Description:

Displays the Sort line in each section (with the exception of the Dashboard section). The default value of this property is true.

Action:

Read-write, Boolean

Example:

This example shows how to hide the sort line in an OLAPQuery:

```
ActiveDocument.Sections["OLAPQuery"].ShowSortLine=false
```

ShowStatusBar (Property)

Applies To:

Application object

Description:

Enables the display of the Status bar. Changing this property is equivalent to showing/hiding the Status bar from the View menu.

Note: Do not use Application.ShowStatusBar and Application.ShowStatusText in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example shows how to hide and show various user interface elements in Interactive Reporting based on the application being executed:

```
if (Application.Name == "BrioQuery Designer")
{
   ActiveDocument.ShowCatalog = true
    ActiveDocument.ShowSectionTitleBar = true
    Application.ShowStatusBar = true
   Application.ShowMenuBar = true
}
```

```
else
{
//Save space in plugin by hiding by various user interface elements
   ActiveDocument.ShowCatalog = false
   ActiveDocument.ShowSectionTitleBar = false
   Application.ShowStatusBar = false
   Application.ShowMenuBar = false
}
```

ShowSubTitle (Property)

Applies To:

ChartSection object

Description:

Displays the Chart sub-title.

Action:

Read-write, Boolean

Example:

This example shows how to add a sub-title to a chart:

```
var MyChart=ActiveDocument.Sections["Chart"]
MyChart.SubTitle="This is the Sub Title"
MyChart.ShowSubTitle=true
```

ShowTickmarks (Property)

Applies To:

XAxis object, ZAxis object

Description:

Displays tick marks on the X or Y axis.

Action:

Read-write, Boolean

Example:

This example shows how to display tick marks on the X-axis and hide them on the Z-axis:

```
var MyChart = ActiveDocument.Sections["Chart"]
MyChart.LabelsAxis.XAxis.ShowTickmarks = true
```

```
MyChart.LabelsAxis.ZAxis.ShowTickmarks = false
```

ShowTitle (Property)

Applies To:

ChartSection object

Description:

Enables the displays of a Chart title.

Action:

Read-write, Boolean

Example:

This example shows how to add a title to a chart:

```
var MyChart=ActiveDocument.Sections["Chart"]
MyChart.Title="This is the Title"
MyChart.ShowTitle=true
```

ShowValues (Property)

Applies To:

XAxis object, ZAxis object

Description:

Enables the display of the values along the X-axis and the Z-axis.

Action:

Read-write, Boolean

Example:

This example shows how to display values only on the X-axis:

```
var MyChart = ActiveDocument.Sections["Chart"]
MyChart.LabelsAxis.XAxis.ShowValues = true
MyChart.LabelsAxis.ZAxis.ShowValues = false
```

ShowValuesAtRight (Property)

Applies To:

ValuesAxis object

Description:

Displays values to the right of the Values axis.

Action:

Read-write, Boolean

Example:

This example shows how to display the values to the right of the axis:

```
var MyChart = ActiveDocument.Sections"Chart"]
MyChart.ValuesAxis.ShowValuesAtRight = true
```

ShowVerticalPlane (Property)

Applies To:

ChartSection object

Description:

Enables the display of the vertical plane in a Chart section.

Action:

Read-write, Boolean

Example:

This example shows how to display the vertical plane on a chart:

```
ActiveDocument.Sections["Chart"].ShowVerticalPlane=true
```

ShowZeroBubbles (Property)

Applies To:

BubbleChart object

Description:

Renders bubbles of zero size using a small bubble size to prevent them from disappearing.

Action:

Read-write, Boolean

Example:

This example shows to the display a bubble with zero values:

```
ActiveDocument.Sections["Chart2"].BubbleChart.ShowZeroBubbles = true
```

Size (Property)

Applies To:

Font object

Description:

Returns or sets the size of text associated with font objects.

Action:

Read-write, Numeric

Example:

This example shows how to change the size of text associated with a text label:

```
var MyLabel = ActiveDocument.Sections"Dashboard"].Shapes["TextLabel1"]
MyLabel.Font.Size = 14
MyLabel.Font.Style = bqFontStyleBoldItalic
```

SmartScaling (Property)

Applies To:

ChartSection object

Description:

Enables the Chart Smart Scaling feature, which determines how Chart components display initially, reposition, and change during Chart resizing. Smart Scaling uses a display hierarchy or drawing order to determine what is displayed. The display hierarchy is based on the number of components and the initial size of the chart boundary. The display hierarchy for Chart sections is as follows.

Bar, Line and Area charts have this display order:

- Planes
- Bar/Lines/Ribbons

- Z axis values
- X axis values
- Y axis values
- Legend
- Titles
- Axis
- Inserted text

Pie charts have this display order:

- Slices
- Slice Labels
- Legend
- Titles
- Inserted text

Smart Scaling uses the hierarchy to determine which components are you add are incorporated or omitted.

Actions:

Read-write, Boolean

Example:

This example shows how to enable Smart Scaling for a Chart section:

```
ActiveDocument.Sections["Chart"].SmartScaling = true
```

SortAscending (Property)

Applies To:

ReportTable object

Description:

Sets the sort ascending order for items in a ReportTable column.

Action:

Read-write, Boolean

Example

This example shows how to sort items in the *Region* column in ascending order:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].SortItems["Region"].SortAscending
 = true
```

SortDescending (Property)

Applies To:

ReportTable object

Description:

Sets the sort descending order for items in a ReportTable column.

Action:

Read-write, Boolean

Example

This example shows how to sort items in the *Region* column in descending order:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].SortItems["Region"].SortDescendin
g = true
```

SortFactName (Property)

Applies To:

PivotLabels collection, XCategory object, ZCategory object

Description:

Sets the sort criteria for a Pivot fact item. Use this property with SortByFact (Method).

Actions:

Read-only, String

Example:

This example shows how to sort the side label *Product Name* by fact value:

```
ActiveDocument.Sections["Pivot3"].SideLabels["Product Name"].SortFactName="Unit Sales"
```

SortFunction (Property)

Applies To:

PivotLabels collection, XCategory object, ZCategory object

Description:

Returns or sets aggregate statistical functions. This property takes a BqSortFunction group value, which duplicates the data functions available in the Pivot and Chart sections. Use this property with SortByFact (Method) which enables you to sort by a numeric data item.

Action:

Read-only, String

Constants:

The BqSortFunction group constant has these values:

- bqSortFunctionAverage
- bySortFunctionCount
- bqSortFunctionMaximum
- bqSortFunctionMinimum
- bqSortFunctionNonNullAverage
- bqSortFunctionNonNullCount
- bqSortFunctionNullCount
- bqSortFunctionSum

Example

This example shows how to sort values based on the *average* statistical function:

```
ActiveDocument.Sections["Pivot3"].SideLabels["Product Name"].SortFunction=
bqSortFunctionAverage
```

SortOrder (Property)

Applies To:

Sort Items, XCategory object, ZCategory object

Description:

Returns or sets the ascending or descending sort order property. The SortNow (Method) is required to use the SortOrder (Property).

Action:

Read-write

Constants

The constant associated with this property is a member of the constant group called BqSortOrder, which consists of the bqSortAscend and bqSortDescend constants.

Example:

This example shows how to use the SortOrder (Property) in a Table section. The example references the Add (Method), Sorting (Method), SortOrder (Property) and BqSortOrder constant groups. It involves four command buttons and two text boxes that record the actions of the scripts as they execute.

1st Command button

```
//adds the Store Id conditon
ActiveSection.Shapes["CommandButton1"].OnClick()
TextBox1.Text = "Start Add Sort"
try
{
ActiveDocument.Sections["Table"].SortItems.Add("Store Id")
}
catch(e)
{
TextBox2.Text = e.toString()
}
TextBox1.Text = "End Add Sort"
```

2nd Command Button

```
//invokes the SortNow method
ActiveSection.Shapes["CommandButton1"].OnClick()
TextBox1.Text = "Start SortNow"
try
{
ActiveDocument.Sections["Table"].SortItems.SortNow()}
catch(e)
{
TextBox2.Text = e.toString()
}
TextBox1.Text ="End SortNow"
```

3rd Command Button

```
//invokes the SortOrder property and bqSortAscend constant
ActiveSection.Shapes["CommandButton1"].OnClick()
TextBox1.Text = "Start SortOrder"
try
{
TextBox1.Text = "Step1"
ActiveDocument.Sections["Table"].SortItems[1].SortOrder = bqSortAscend
TextBox1.Text = "Step2"
TextBox3.Text = ActiveDocument.Sections["Table"].SortItems[1].SortOrder
TextBox1.Text = "Step3"
switch(ActiveDocument.Sections["Table"].SortItems[1].SortOrder)
{
case 1:
TextBox3.Text = "Sort Order is bqSortAscend"
break;
case 2:
TextBox3.Text = "Sort Order is bqSortDescend"
break;
default:
```

```
TextBox3.Text = "No Sort Order is Available"
}
}
catch(e)
{
TextBox2.Text = e.toString()
}
TextBox1.Text = "End SortOrder"
```

4th Command Button

```
//invokes the SortOrder property and bqSortAscend constant
TextBox1.Text = "Start SortOrder"
try
{
TextBox1.Text = "Step1"
ActiveDocument.Sections["Table"].SortItems[1].SortOrder = bqSortDescend
TextBox1.Text = "Step2"
TextBox3.Text = ActiveDocument.Sections["Table"].SortItems[1].SortOrder
TextBox1.Text = "Step3"
switch(ActiveDocument.Sections["Table"].SortItems[1].SortOrder)
{
case 1:
TextBox3.Text = "Sort Order is bqSortAscend"
break;
case 2:
TextBox3.Text = "Sort Order is bqSortDescend"
break;
default:
TextBox3.Text = "No Sort Order is Available"
}
}
catch(e)
{
TextBox2.Text = e.toString()
}
TextBox1.Text = "End SortOrder"
```

SortOrderLang (Property)

Applies To:

Document object

Action:

Read/Write

Description:

Sets the properties of an Interactive Reporting document that controls how language dependent data sorting is performed in document sections. These values of bqLanguage enumeration can be used:

- bqLangAfrikaans
- bqLangAlbanian
- bqLangArabic
- bqLangBasque
- bqLangCatalan
- bqLangChinese_Simp (pinyin sort)
- bqLangChinese_Trad (big5han sort)
- bqLangChinese_GB2312_Sort
- bqLangChinese_Stroke_Sort
- bqLangCroatian
- bqLangCzech
- bqLangDanish
- bqLangDutch
- bqLangEnglish
- bqLangUKEnglish
- bqLangEstonian
- bqLangFinnish
- bqLangFrench
- bqLangGerman
- bqLangGerman_Phonebook_Sort
- bqLangGreek
- bqLangHebrew
- bqLangHungarian
- bqLangIcelandic
- bqLangIndonesian
- bqLangItalian
- bqLangItalian
- bqLangJapanese
- bqLangKorean
- bqLangLatvian
- bqLangLithuanian
- bqLangMacedonian
- bqLangMalay
- bqLangNorwegian_Bokmal
- bqLangNorwegian_Nynorsk

- bqLangPolish
- bqLangPortuguese
- bqLangRomanian
- bqLangRussian
- bqLangSerbian_Cyrillic
- bqLangSerbian_Latin
- bqLangSlovak
- bqLangSlovenian
- bqLangSpanish
- bqLangSpanish_Trad_Sort
- bqLangSwedish
- bqLangThai
- bqLangTurkish
- bqLangUkrainian
- bqLangVietnamese

Action:

Read/Write

SourceSectionName (Property)

Applies To:

Bullet object, (Live) BarChart object, (Live) BlockChart object, (Live) FunnelChart object, (Live) LineChart object, (Live) PieChart object, (Live) RadarChart object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Description:

Returns or sets the section name used for the source data set of a gauge or (Live) chart object. For each object, you can select only numeric data type values from the same section. For example, you cannot select an actual value from the Results section and a target value from the Table section. If you add more than one gauge object, you can use values from the data set of another query. Target values can be selected from the database if a column is available, or you can create a computed item reflecting the target.

Action:

Read only, String

Example:

This example shows how to set the section name to the Results section

```
TrafficLight.SourceSectionName = "Results"
```

SpecificMetadataLogin (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables the log in information specified in the default connection for the metadata connection.

Action:

Read-write, Boolean

Example:

This example creates an OCE and applies it to the current document. The data source name is *PlutoSQLSVR*, which is a user DSN using the SQL Server 6.5 driver:

```
var myCon = Application.CreateConnection()
myCon.Description"This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiOpenClient
myCon.Database = bqDatabaseSQLServer
myCon.HostName ="PlutoSQLSVR"
myCon.EnableAsyncProcess = true
myCon. SpecificMetadataLogin = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoSQL.oce")
```

SQLDecimalPositions (Property)

Applies To:

Limit object

Description:

Specifies the number of digits to the right of decimal points in SQL numeric literal statements. The SQL statement uses this setting to determine which values to recognize. For example, if 0.12345678 is the filter value, the SQL statement writes “where column = 0.12346” (the rounded value). However, if you change the spinner to 8, SQL writes “where column = 0.12345678”.

Action:

Read-write, Boolean

Example:

This example shows how to set the SQL decimal places to 6.

```
ActiveDocument.Sections["SalesQuery"].Limits["Amount Sales"].SQLDecimalPositions = 6
```

SQLName (Property)

Applies To:

Requests collection

Description:

Returns the name of the Request object used in building SQL statements.

Action:

Read-only, String

Example:

This example shows how to display all names used in the SQL statement corresponding to the Request Line items:

```
var RequestCount = ActiveDocument.Sections["Query"].Requests.Count
for (j =1 ; j <= RequestCount ; j++)
     {
    var DisplayName  = ActiveDocument.Sections["Query"].Requests[j].DisplayName
    var SQLName = ActiveDocument.Sections["Query"].Requests[j].SQLName
     Console.Writeln("The column named "+ DisplayName + "is actually known by "+ SQLName
+ "to the database.")
     }
```

SQLNetRetainDateFormats (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

SQLNet Only. This property enables the retention of date formats specified by SQLNet.

Action:

Read-write, Boolean

Example:

This example creates a connection file and applies it to the current document:

```
Var myCon = Application.CreateConnection()
MyCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
MyCon.Api = bqApiSQLNet
MyCon.Database = bqDatabaseOracle71
MyCon.HostName ="PlutoORACLE"
MyCon. SQLNetRetainDateFormats =true
MyCon.SaveAs("d:\\OCEs\\PlutoORACLE.oce")
//Now use this connection in a data model
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoORACLE.oce")
```

StackClusterType (Property)

Applies To:

BarLineChart object

Description:

Sets the value of the cluster by type. This property uses the BqClusterBarType constant.

Action:

Read-write

Constants:

The BqClusterBarType constant consists of the bqClusterByY and bqClusterByZ values.

Example:

This example shows how to change the type of Bar Line Chart:

```
var MyChart = ActiveDocument.Sections["Chart"]
MyChart.BarLineChart.StackClusterType = bqClusterByY
```

StatusText (Property)

Applies To:

Application object

Description:

Provides the status text displayed on the status bar.

Note: Do not use Application.ShowStatusBar and Application.StatusText in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, String

Example:

This example shows how to hide or only show various user interface elements in Interactive Reporting based on the application running. The status text indicates if the Catalog and Section Title Bar are shown based on the application.

```
Application.ShowStatusBar = true

if (Application.Type == bqAppTypeDesktopClient)
{
 ActiveDocument.ShowCatalog = true
 ActiveDocument.ShowSectionTitleBar = true
 Application.StatusText = "Catalog and Section Title Bar are shown"
}
else
{
//Save space in plugin by hiding by various user interface elements
   ActiveDocument.ShowCatalog = false
   ActiveDocument.ShowSectionTitleBar = false
   Application.StatusText = "Catalog and Section Title Bar are not shown"
}
```

StringRetrieval (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Enables a connection to use string retrieval. If this property is disabled, binary retrieval is used.

Action:

Read-write, Boolean

Example:

This example creates a connection and applies it to the current document:

```
Var myCon = Application.CreateConnection()
MyCon.Description = "This OCE configures the connection via ODBC, to a SQLServer 6.5
database named pluto."
myCon.Api = bqApiSQLNet
myCon.Database = bqDatabaseOracle71
myCon.HostName ="PlutoORACLE"
```

```
myCon. StringRetrieval =true
myCon.SaveAs("d:\\OCEs\\PlutoORACLE.oce")
//Now use this connection in a datamodel
ActiveDocument.Sections["Query"].DataModel.Connection.Open
("d:\\OCEs\\PlutoORACLE.oce")
```

Style (Property)

Applies To:

Font object, Themes collection

Description:

Returns or sets the value of text associated with the font object.

Action:

Read-write

Constants:

The BqFontStyle constant has these values:

- bqFontStyleBold
- bqFontStyleBoldItalic
- bqFontStyleItalic
- bqFontStyleNone
- bqFontStyleRegular

Example:

This example shows how to change the size of the text associated with a text label:

```
var MyLabel = ActiveDocument.Sections["Dashboard"].Shapes["TextLabel1"]
MyLabel.Font.Size = 14
MyLable.Font.Style = bqFontStyleBoldItalic
```

SubTitle (Property)

Applies To:

ChartSection object

Description:

Returns or sets the value of a Chart sub-title.

Action:

Read-write, String

Example:

This example shows how to add a sub-title to a Chart:

```
ActiveDocument.Sections["Chart"].SubTitle ="This is the sub title"
ActiveDocument.Sections["Chart"].ShowSubTitle=true
```

Subtype (Property)

Applies To:

Bullet object, (Live) BarChart object, (Live) BlockChart object, LineChart object, (Live) PieChart object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Description:

Returns or sets the degrees of orientation of the gauge.

For a (Live) PieChart object, the BqLivePieChartType determines the type of (Live) PieChart. It consists of these values:

- BqLivePieChartTypePie—Displays pieces (slices) of the pie drawn to represent the relative value of a measurable item to the whole. Only one category (data series) can be plotted in a Pie Chart.
- BqLivePieChartTypeDonut—Drawn like a Pie Chart, the Donut chart does not include a center hole.

For a Live BarChart object, the BqLiveBarChartype determines the type of BarChart,. It consists of the following values:

- bqLiveBarChartType100PctStack
- bqLiveBarChartTypeCluster
- bqLiveBarChartTypeStack

For a Live BlockChart object, the BqLiveBlockChartType determines the type of BlockChart. It consists of the following values:

- bqLiveBlockChartTypeCone
- bqLiveBlockChartTypePyramid

For a Live LineChart object, the BqLiveLineChartType determines the type of LineChart. It consists of the following values:

- bqLiveLineChartTypeArea
- bqLiveLineChartTypeLine

Action;

Read-write, String

The following is a list of subtype strings by gauge:

Speedo object:

- 360 degree orientation—speedo_full
- 180 degree orientation—speedo_180
- 90 degree left orientation—speedo_90left
- 90 degree right orientation—speedo_90right

Thermometer object:

- Vertical thermometer orientation—Vertical_Thermometer
- Horizontal thermometer orientation—Horizontal_Thermometer

Bullet object:

- Vertical bullet orientation—Vertical_Bullet
- Horizontal bullet orientation—Horizontal_Bullet

TrafficLight object:

- Vertical Traffic Light orientation—Vertical_Traffic_Light
- Horizontal Traffic Light orientation—Horizontal_Traffic_Light
- Single Traffic Light orientation—Single_Traffic_Light

Example:

This example shows how to set the orientation of a traffic light gauge to a single light:

```
TrafficLight.Subtype = "Single_Traffic_Light"
```

SuppressDuplicates (Property)

Applies To:

Column object

Description:

Suppresses duplicate values in columns.

Action:

Read-write, Boolean

Example:

This example shows how to suppress duplicate results in columns in a Results section:

```
var Col1 = "State"
var Col2 = "City"
ActiveDocument.Sections"Results"].Columns[Col1].SupressDuplicates = true
ActiveDocument.Sections["Results"].Columns[Col2].SupressDuplicates = true
```

SuppressEmptyRows (Property)

Applies To:

DBSpecific object (MS OLAP and SAP BW)

Description:

Enables the suppression of any empty rows for which there is no measure data.

Action:

Read-write, Boolean

Example:

This example shows how to suppress empty rows in an OLAPQuery:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.SuppressEmptyRows=true
```

SuppressMissingColumns (Property)

Applies To:

QueryOptions object

Description

Suppresses the return of data columns that contain only missing data in the CubeQuery section.

Action:

Read-write, Boolean

Example:

This example shows how to enable the SuppressMissingColumns (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox5"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.SuppressMissingColumns = true}
else {ActiveDocument.Sections["Query"].QueryOptions.SuppressMissingColumns= false}
```

SuppressMissingRows (Property)

Applies To:

Database (Essbase and DB2 OLAP) — specific object, QueryOptions object.

Description:

In the OLAPQuery section, this property suppresses missing rows in which all cells are null.

In the CubeQuery section, this property suppresses the return of data rows that contain only missing values. If one value is in the row, the row is not suppressed.

Action:

Read-write, Boolean

Example 1:

This example shows how to suppress empty rows in an OLAP query:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.SuppressMissingRows=true
```

Example 2:

This example shows how to enable the SuppressMissingRows (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox2"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.SuppressMissingRows = true}
else {ActiveDocument.Sections["Query"].QueryOptions.SuppressMissingRows= false}
```

SuppressSharedMembers (Property)

Applies To:

QueryOptions object

Description:

Suppresses the return of members tagged as shared. Shared member instructs Essbase to share the member's data with one or more other members.

Action:

Read-write, Boolean

Example:

This example shows how to enable the SuppressSharedMembers (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox4"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.SuppressSharedMembers = true}
else {ActiveDocument.Sections["Query"].QueryOptions.SuppressSharedMembers= false}
```

SuppressZeroColumns (Property)

Applies To:

QueryOptions object

Description:

Suppresses the return of data columns that contain only zeros in the CubeQuery section.

Action:

Read-write, Boolean

Example:

SuppressZeroRows (Property)

Applies To:

In the OLAPQuery section, applies to Database (Oracle Essbase and DB2 OLAP) specific object.
In the CubeQuery section, applies to the QueryOptions object.

Description:

Enables the suppression of any zero rows where all cells are null.

Action:

Read-write, Boolean

Example 1:

This example shows how to suppress empty rows in an OLAPQuery:

```
ActiveDocument.Sections["OLAPQuery"].DBSpecific.SuppressZeroRows=true
```

Example 2:

his example shows how to enable the SuppressZeroRows (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions"].Shapes["CheckBox3"].Checked)
    { ActiveDocument.Sections["Query"].QueryOptions.SuppressZeroRows = true}
else {ActiveDocument.Sections["Query"].QueryOptions.SuppressZeroRows= false}
```

SurfaceValues (Property)

Applies To:

PivotSection object

Description:

Enables Pivots to perform calculations based on the computed item formula when a break total is evaluated. Changes to the property are selected in the user interface when the property is set. Changes to the user interface are reflected when the property is read. The default value of this property is false.

Average and Count aggregation Data Functions are not evaluated in True Total mode unless the Use Surface Values property is enabled. If disabled, the Average and Count aggregation are calculated using the count of the underlying Table/Result Section data values.

Action:

Read-write, Boolean

Example 1:

This example shows how to enable surface values:

```
//Surface Values ON
ActiveDocument.Sections["Pivot"].SurfaceValues=true
```

Example 2:

This example shows how to disable surface values:

```
//Surface Values OFF
ActiveDocument.Sections["Pivot"].SurfaceValues=false
```

SuspendCalculation (Property)

Applies To:

ReportSection

Description:

Prevents graphic recalculation of the Report section when items are added through the object model. Enable this property to greatly enhance Report section performance

Note: Use the Recalculate (Method) or Process (Method) to force a recalculation when using this property.

Action:

Read-write, Boolean

Example:

This example shows how to increase the performance of the Report section using SuspendCalculation (Property):

```
ActiveDocument.Sections["Report"].SuspendCalculation = true
```

SuspendRecalculation (Property)

Applies To:

Limit (Results limits only)

Description:

Prevents the Results limit from recalculating after modification. This greatly enhances the performance of Results limit calculations.

Note: Use Recalculate (Method) or Process (Method) to force a recalculation when you use this property.

Action:

Read-write, Boolean

Example:

This example shows how to increase the performance of limits applied to a results set using Suspend Recalculation (Property):

```
var MyLimit = ActiveDocument.Sections["Results"].Limits["Units"]
MyLimit.SuspendRecalculation = true
MyLimit.SelectedValues.RemoveAll()
MyLimit.SelectedValues.Add(10)
MyLimit.SelectedValues.Add(11)
MyLimit.SelectedValues.Add(12)
MyLimit.SuspendRecalculation = false
ActiveDocument.Sections["Results].Recalculate()
```

Note: Instead of calculating the results limit four times, the script above only calculates it once.

Text (Property)

Applies To:

ControlsTextBox, EmbeddedBrowser object, HyperLink object, Title object

Description:

Returns or sets the value of the text that is displayed in a Text box control or Text label shape.

Action:

Read-write, String

Example:

This example shows how to set an initial value for a text box:

```
ActiveDocument.Sections["Dashboard2"].Shapes["TextBox1"].Text="Hello World"
Alert (ActiveDocument.Sections["Dashboard2"].Shapes["TextBox1"].Text)
```

TextWrap (Property)

Applies To:

Column object, Legend object

Description:

Enables text in a column to wrap and extend to the height of the column, or legend provided sufficient vertical space is available.

Action:

Read-write, Boolean

Example:

This example shows how to force text to wrap on specific columns within the Results section:

```
var Col1 = "State"
var Col2 = "City"
ActiveDocument.Sections["Results"].Columns[Col1].TextWrap = false
ActiveDocument.Sections["Results"].Columns[Col2]. TextWrap = true
```

Example:

This example shows to force text to wrap in a chart legend:

```
ActiveDocument.Sections["BooksChart"].Legend.TextWrap = false
```

TickmarkFrequency (Property)

Applies To:

XAxis object

Description:

Returns or sets the number of tick marks displayed on the X-axis.

Action:

Read-write, Numeric

Example 1:

This example shows how to display a tick mark for every value on the X-axis:

```
ActiveDocument.Sections["AllChart"].LabelsAxis.XAxis.TickmarkFrequency=1
ActiveDocument.Sections["AllChart"].LabelsAxis.XAxis.ShowTickmarks=true
```

Example 2:

This example shows how to display a tick mark for every other value on the X-axis:

```
ActiveDocument.Sections["AllChart"].LabelsAxis.XAxis.TickmarkFrequency=2
ActiveDocument.Sections["AllChart"].LabelsAxis.XAxis.ShowTickmarks=true
```

TimeAwareAutoRange (Property)

Applies To:

XAxisLabel (Object)

Description:

Enables the calculation of the time range for a Time Aware axis automatically. The minimum and maximum time range values are searched in the current data set. The time range is recalculated automatically when the data set is processed.

Action:

Read-write: Boolean

Example:

This example shows how to enable the Auto Range feature:

```
ActiveDocument.Sections["Chart2"].LabelsAxis.XAxis.TimeAwareAutoRange = true
```

TimeAwareLabelIntervalType (Property)

Applies To:

XAxisLabel (Object)

Description:

Enables you to manually specify the recurring intervals at which to display a limited number of labels. This property is used with the bqTimeAwareIntervalUnit constant group.

Action:

Read-write, BQTimeAwareIntervalUnit

Constants:

The BQTimeAwareIntervalUnit constant group consists of these values:

- bqTimeAwareDay
- bqTimeAwareHour
- bqTimeAwareMinute
- bqTimeAwareMonth
- bqTimeAwarePage
- bqTimeAwareSecond
- bqTimeAwareValue
- bqTimeAwareWeek
- bqTimeAwareYear

Note: The bqTimeAwarePage constant is not applicable to the TimeAwareLabelIntervalType property.

Example:

This example shows how to set the place the label interval type to every fifth day.

```
ActiveDocument.Sections["Chart2"].LabelsAxis.XAxis.TimeAwareAutoRange = true
ActiveDocument.Sections["Chart2"].LabelsAxis.XAxis.LabelFrequency = 5
ActiveDocument.Sections["Chart2"].LabelsAxis.XAxis.TimeAwareLabelIntervalType =
bqTimeAwareDay
```

TimeAwareScrollIntervalType (Property) TimeAwareScrollMaxDisplayed (Property) TimeAwareTickIntervalType (Property)

TimeAwareOn (Property)

Applies To:

XAxisLabel object

Description:

Sets the discrete X-Axis into a continuous data fact axis, and arranges and plots timestamp values on a continuous line of time. A Time Aware Axis is an axis considered as a time scale with its minimum and maximum bounds. The data is arranged along the axis according to the matching time values. The distance between adjacent axis items is proportional to their time value difference: a non-Time Aware Axis presents all the items using the same distance between them. The Time Aware feature only implements an alternative visualization of source data and does not affect the way how the data is aggregated and computed items are calculated. That is, the data processing in chart section which includes dividing data into categories and calculating fact data does not depend on whether the Time Aware feature is activated. In particular, behavior of the "Chart This Pivot", "Pivot This Chart" and "Add Computed Item" actions do not change. This property is only functional when the X-axis has a category of date/time type to display. In this case, it automatically becomes a dedicated Time Scale Category. In addition, no other categories can be on the X-Axis. Adding multiple categories on X-Axis makes the property inactive.

Action:

Read-write:, Boolean

Example:

This example shows how to enable the Time Aware axis feature.

```
ActiveDocument.Sections["Chart2"].LabelsAxis.XAxis.TimeAwareOn = true
```

TimeAwareIntervalScrollType (Property)

Applies To:

XAxisLabel (Object)

Description:

Reads or sets the value in the Labels Axis "Scrolling" region "at every n interval" dropdown on the Label Axis tab. That is, it specifies whether to use a page or a date/time value when scrolling through a fixed number of intervals for the Time Aware axis. The TimeAwareIntervalScrollType (Property) uses the BqTimeAwareIntervalUnit constant group. The property persists with the document/application.

Action:

Read-write:, BqTimeAwareIntervalUnit

Constants:

The BqTimeAwareIntervalUnit constant group consists of these values:

- bqTimeAwareDay
- bqTimeAwareHour
- bqTimeAwareMinute
- bqTimeAwareMonth
- bqTimeAwarePage
- bqTimeAwareSecond
- bqTimeAwareValue
- bqTimeAwareWeek
- bqTimeAwareYear

Note: The bqTimeAwareValue constant is not applicable to the TimeAwareIntervalScrollType property.

Example:

This example shows how to enable the scrolling for the Time Aware axis. It also shows how to enable the Time Aware axis, and to set the scrolling interval type to a maximum of five days.

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollEnabled =
true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollIntervalTyp
e = bqTimeAwareDay
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollMaxDisplaye
d = 5
```

TimeAwareMax (Property)

Applies To:

XAxisLabel (Object)

Description:

Sets the maximum time range interval when setting the time range of a Time Aware axis manually.

Action:

Read-write, Date

Example:

This example shows how to enable the time aware feature, disable the auto arrange property, and set the time aware minimum value to 4/4/96 at 1:45 and fifty-seven seconds:

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareAutoRange = false
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareMa x = new
Date("4/4/96 13:45:57")'
```

TimeAwareMin (Property)

Applies To:

XAxisLabel (Object)

Description:

Sets the minimum time range interval when setting the time range of a Time Aware axis manually.

Action:

Read-write, Date

Example:

This example shows how to enable the time aware feature, disable the auto arrange property, and set the time aware minimum value to 4/4/96 at 1:45 and fifty-seven seconds:

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareAutoRange = false
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareMin = new
Date("4/4/96 13:45:57")'
```

TimeAwareScrollEnabled (Property)

Applies To:

XAxisLabel (Object)

Description:

Enables you to select the number of pages or a date/time values to display when scrolling through a fixed number of intervals for a Time Aware axis. This is like selecting the “Display” checkbox in the “Scrolling” region in the Label Axis tab of General Properties

The Default value of the property is false, even for a chart which does not have the Time Aware feature enabled (the TimeAwareOn option is checked but displays the text “Time Aware(inactive now)”). The property persists with the document/application. When time aware inactive, the non time aware controls are displayed in "Scrolling" region of the dialog box. These controls

correspond to other Object Model properties, in particular MaximumBarsEnable and MaximumBarsDisplayed.

Action:

Read-write, Boolean

Example:

This example shows how to enable the scrolling for the Time Aware axis. It also shows how to enable the Time Aware axis and set the scrolling interval type to a maximum of five days.

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollEnabled =
true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollIntervalTyp
e = bqTimeAwareDay
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollMaxDisplaye
d = 5
```

TimeAwareScrollMaxDisplayed (Property)

Applies To:

XAxisLabel (Object)

Descriptions

Setting the TimeAware ScrollMaxDisplayed property is equivalent to entering the maximum number of values to display when scrolling through values on the Time Aware axis. The property persists with the document/application.

Action:

Read-write, Numeric

Example:

This example shows how to enable the scrolling for the Time Aware axis. It also shows how to set the scrolling interval type to a maximum of five days.

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollEnabled =
true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollIntervalTyp
e = bqTimeAwareDay
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareScrollMaxDisplaye
d = 5
```

TimeAwareTickIntervalType (Property)

Applies To:

XAxisLabel (Object)

Description:

Enables you to specify the recurring intervals at which to display tick marks. This property is used with the BQTimeAwareIntervalUnit constant group.

Action:

Read-write, BQTimeAwareIntervalUnit

Constants:

The BQTimeAwareIntervalUnit constant group consists of these values:

- bqTimeAwareDay
- bqTimeAwareHour
- bqTimeAwareMinute
- bqTimeAwareMonth
- bqTimeAwarePage
- bqTimeAwareSecond
- bqTimeAwareValue
- bqTimeAwareWeek
- bqTimeAwareYear

Note: The bqTimeAwarePage constant is not applicable to the TimeAwareTickIntervalType property.

Example:

This example shows how to set the place the tickmark interval type to every fifth day. The script enables Time Aware, disables the AutoFrequency property, enables the Show Tickmarks property and sets the tickmark interval to every fifth week.

```
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareOn = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.AutoFrequency = false
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.ShowTickmarks = true
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TickmarkFrequency = 5
ActiveDocument.Sections["UnitProdLineChart"].LabelsAxis.XAxis.TimeAwareTickIntervalType
= bqTimeAwareWeek
```

TimeLimit (Property)

Applies To:

Connection object, MetaDataConnection object, DataModel object, QuerySection object

Description:

Returns or sets the maximum time limit a query can process before timing out. It can be set on the OCE, DataModel or Connection level. The time increment is minutes.

Action:

Read-write, Numeric

Example:

This example shows how to set the TimeLimit (Property) for all the supported objects:

```
//Connections
var myCon = Application.CreateConnection()
myCon.Api = bqApiSQLNet
myCon.Database = bqDatabaseOracle71
myCon.HostName ="PlutoORACLE"
myCon.TimeLimit = 20
myCon.SaveAs("d:\\OCEs\\PlutoORACLE.oce")
//DataModel
ActiveDocument.Sections["Query].DataModel.TimeLimit = 30
//Query
ActiveDocument.Sections["Query].TimeLimit = 30
```

TimeLimitActive (Property)

Applies To:

QuerySection object, DataModel object

Description:

Enables the "TimeLimit (Property)" on page 236.

Action:

Read-only, Boolean

Example:

This example shows how to enable the TimeLimitActive (Property), set the maximum time limit to process a query before timing out, and process the query:

```
ActiveDocument.Sections["Query"].DataModel.TimeLimitActive = true
ActiveDocument.Sections["Query"].DataModel.TimeLimit = 30
```

```
ActiveDocument.Sections["Query"].Process()
```

Title (Property)

Applies To:

ChartSection object,

Description:

Returns or sets the value of the title displayed on a Chart.

Action:

Read-write, String

Example:

This example shows how to add a title to a chart.

```
var MyChart = ActiveDocument.Sections"Chart"]
MyChart.Title = "This is the Title"
MyChart.ShowTitle = true
```

Tooltips (Property)

Applies To:

ColorRange object

Description:

Returns or sets a description of the color range. For example, you might assign “poor” for the color red, or “excellent” for the color green for a TrafficLight gauge. The tool tip is specific to the color range. The default tool tip description is “Current value is xxx”

Action:

Read-write, String

Example:

This example shows how to set the tool tip text to “Excellent” for the first color ragne:

```
Speedometer.ColorRanges[1].Tooltips = "Excellent"
```

TopMargin (Property)

Applies To:

ReportSection object

Description:

Sets the top margin of the report. Margins are set for the entire report, read-write, Number

Example:

This example shows how to set the top margin of the report to .25 inches.

```
ActiveDocument.Sections["Report"].TopMargin = .25
```

TopicName (Property)

Applies To:

Join object, Local Join

Description:

Retrieves the parent of the Topic item, which is the Topic Name in a join or local join. It also enables you to retrieve the Topic Item Names of joins (and not local joins).

Action:

Read-only, String

Example 1:

This example shows how to retrieve the topic names 1 and 2 from a join.

```
//Get Join Topic Names
TextBox1.Text=ActiveDocument.Sections["Query"].DataModel.Joins["1"].Topic1Name;
TextBox2.Text=ActiveDocument.Sections["Query"].DataModel.Joins["1"].Topic2Name;
TextBox3.Text=ActiveDocument.Sections["Query"].DataModel.Joins["1"].Type;
```

Example 2:

This example shows how to retrieve the Topic Item Names from a join.

```
/Get Join Topic Item Names
TextBox4.Text=ActiveDocument.Sections["Query"].DataModel.Joins["1"].TopicItem1.DisplayNa
me
TextBox5.Text=ActiveDocument.Sections["Query"].DataModel.Joins["1"].TopicItem2.PhysicalN
ame
```

Example 3:

This example shows how to retrieve the topic names 1 and 2 from a local join.

```
//Get Local Join Topic Names
TextBox6.Text=ActiveDocument.Sections["Query"].DataModel.LocalJoins["1"].Topic1Name;
TextBox7.Text=ActiveDocument.Sections["Query"].DataModel.LocalJoins["1"].Topic2Name;
TextBox8.Text=ActiveDocument.Sections["Query"].DataModel.LocalJoins["1"].Type;
```

TrueComputedItemTotals (Property)

Applies To:

PivotSection object

Description:

Recalculates all computed item break totals to equal to the sum (or other aggregation data function) of their displayed values in the Computed item column of the Pivot Section. To calculate break totals using the computed item formula, set the Boolean value to false in arguments.

When this property is false, the value in a computed item break totals is calculated based on the formula used to create the corresponding break total in the source fact column.

Average and Count aggregation Data Functions are not evaluated in True Total mode unless the "SurfaceValues (Property)" on page 226 is also enabled. If the Surface Values (Property) is not enabled, the Average and Count aggregation are calculated using the count of the underlying Table/Result Section data values instead of the displayed Pivot values.

Action:

Read-write, Boolean

Example:

This example shows how to enable the true computed items totals option.

```
ActiveDocument.Sections["Pivot"].TrueComputedItemTotals = true
```

Type (Property)

Applies To:

Application object, Bullet object, (Live) BarChart object, (Live) BlockChart object, EventScript object,. EmbeddedBrowser object, (Live) FunnelChart object, HyperLink object, Join object, JoinsOptions object, (Live) LineChart object, (Live) PieChart object, PivotSection object ,(Live) RadarChart object, (CubeQuery) QuerySection object, QuerySection, ReportSection object, Section object, Speedometer object, Shape object, Slider object, Thermometer object, Toolbar object, Topic object, TrafficLight object

Description:

Returns the type value of these objects:

- **Section Objects**—Represents the type of section. (Chart, Pivot, Query, etc.)
- **Join**—Refers to the type of join. (Left, right, Outer, etc.)
- **Toolbar**—Rpresents the type of toolbar. (Standard, format, etc.)
- **Topic**—Represents type of topic. (Standard, Meta, etc.)
- **Shape**—Represents the type of drawing object or control in an Dashboard section. (Line, Rectangle, etc.)
- **Joins Options** —Represents the type of join option. (All Topics, Auto Join, etc.)
- **EventScript**—Represents a read-only value of the enumeration BqScriptableEvents

  allows for better performance when checking event types if this is required.
- **External Content**– This property represents the type of shape for an Embedded Browser object or HyperLink object.

> **Note:** Do not use Toolbars["Standard"].Type, Toolbars["Formatting"].Type, Toolbars["Sections"].Type, and Toolbars["Navigation"].Type in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-only

Constants:

Application object —BqAppType

- bqAppTypeDesktopClient
- bqAppTypePlugInClient
- bqAppTypeScheduler
- bqAppTypeThinClient

Section Objects —BqSectionType

- bqChart
- bqCubeQuery
- bqDataModel
- bqDashboard
- bqOLAPQuery
- bqPivot
- bqQuery
- bqReport
- bqResults
- bqTable

Join—BqJoinType

- bqDataModelJoinsOptionAllTopics
- bqDataModelJoinsOptionAutoJoin
- bqDataModelJoinsOptionDefJoin
- bqDataModelJoinsOptionMinTopics
- bqDataModelJoinsOptionRefTopics
- bqJoinLeft
- bqJoinOuter
- bqJoinRight
- bqJoinSimpleEqual
- bqJoinSimpleGreaterThan
- bqJoinSimpleGreaterThanOREqual
- bqJoinSimpleLessThan
- bqJoinSimpleLessThanOrEqual
- bqJoinSimpleNotEqual

Join Options—

Toolbar —BqToolbars

- bqToolbarFormat
- bqToolbarNavigation
- bqToolbarSections
- bqToolbarStandard

Topic—BqTopicType

- bqTopicTypeMeta
- bqTopicTypeNone
- bqTopicTypeQueryObject
- bqTopicTypeResults
- bqTopicTypeStoredProcedure

Shape—BqShapeType

- bqBullet
- bqButton
- bqCheckBox
- bqDropBox
- bqEmbeddedSection

- bqEmbeddedBrowser
- bqEmbeddedSection
- bqHorizontalLine
- bqHyperLink
- bqLine
- beListBox
- bqLiveBarChart
- bqLiveBlockChart
- bqLiveFunnelChart
- bqLiveLineChart
- bqLivePieChart
- bqLiveRadarChart
- bqOval
- bqPicture
- bqRadioButton
- bqRectangle
- bqRoundRectangle
- bqSlider
- bqSpeedometer
- bqTextBox
- bqTextLabel
- bqThermometer
- bqTrafficLight
- bqVerticalLine

JoinsOptions—BqDataModelJoinsOptions

- bqDataModelJoinsOptionAllTopics
- bqDataModelJoinsOptionAutoJoin
- bqDataModelJoinsOptionDefJoin
- bqDataModelJoinsOptionMinTopics
- bqDataModelJoinsOptionRefTopics

EventScript object—BqScriptableEvents

- bqScriptableEventsOnActivate
- bqScriptableEventsOnCalculate
- bqbqScriptableEventsOnCellDoubleClick

- bqScriptableEventsOnChange
- bqScriptableEventsOnClick
- bqScriptableEventsOnClientClick
- bqScriptableEventsOnClientEnter
- bqScriptableEventsOnClientExit
- bqScriptableEventsOnClientSelection
- bqScriptableEventsOnDeactivate
- bqScriptableEventsOnDoubleClick
- bqScriptableEventsOnEnter
- bqScriptableEventsOnExit
- bqScriptableEventsOnPostProcess
- bqScriptableEventsOnPreProcess
- bqScriptableEventsOnRowDoubleClick
- bqScriptableEventsOnSelection
- bqScriptableEventsOnShutdown
- bqScriptableEventsOnStartup

Example:

This example shows how to use the Type (Property) to determine which properties apply to a specific object. In this example, checking the Type (Property) of the Section objects allows the script to process every query in a document.

```
var SecCount = ActiveDocument.Sections.Count
for (j = 1; j <= SecCount ; j++)
     {
      if (ActiveDocument.Sections[j].Type == bqQuery)
            ActiveDocument.Sections[j].Process()
     }
```

UID (Property)

Applies To:

Image object

Description:

Setting the UID (Property) is equivalent to reading the unique identifier of a Resource Manager image. The property is read-only, and the value returned is a string representing this unique identifier. The property value is unique and there is no default value.

Action:

Read only, String

Example

This example shows how to retrieve the unique identifier of a Resource Manager image:

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name

var ActionName = "RM UID"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{

TextBox1.Text ="Step 1"

try
{
TextBox3.Text =  "UID is: " + ActiveSection.Shapes["Picture"].RMImage.UID
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

else
{
TextBox1.Text ="Step 2"

try
{
ActiveSection.Shapes["Picture"].RMImage.UID = TextBox5.Text
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
{
TextBox3.Text =  "UID is: " + ActiveSection.Shapes["Picture"].RMImage.UID
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
```

```
}

TextBox1.Text ="End " + ActionName
```

UILanguage (Property)

Applies To:

Application object

Description:

Defines the localization language of the application. This is the language of menus and messages. Its value is a member of the BqLanguage constant group.

Action:

Read/Write

Constants:

The BqLanguage constant group consists of these values:

- bqLangArabic
- bqLangBasque
- bqLangCatalan
- bqLangChinese_GB2312_Sort
- bqLangChinese_Simp
- bqLangChinese_Stroke_Sort
- bqLangChinese_Trad
- bqLangCroatian
- bqLangCzech
- bqLangDanish
- bqLangDutch
- bqLangEnglish
- bqLangEstonian
- bqLangFinnish
- bqLangFrench
- bqLangGerman
- bqLangGerman_Phonebook_Sort
- bqLangGreek
- bqLangHebrew

- bqLangHungarian
- bqLangIcelandic
- bqLangIndonesian
- bqLangItalian
- bqLangJapanese
- bqLangKorean
- bqLangLatvian
- bqLangLithuanian
- bqLangMacedonian
- bqLangMalay
- bqLangNorwegian_Bokmal
- bqLangNorwegian_Nynorsk
- bqLangPolish
- bqLangPortuguese
- bqLangRomanian
- bqLangRussian
- bqLangSerbian _Cyrillic
- bqLangSerbian_Latin
- bqLangSlovak
- bqLangSlovenian
- bqLangSpanish
- bqLangSpanish_Trad_Sort
- bqLangSwedish
- bqLangThai
- bqLangTurkish
- bqLangUKEnglish
- bqLangUkrainian
- bqLangVietnamese

Example:

This example shows how to read and write the value of the user interface language in an Interactive Reporting document.

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name
```

```
var ActionName = "UILanguage"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{
TextBox1.Text ="Step 1"

try
{
TextBox3.Text =  "UILanguage is: " + Application.UILanguage

switch(Application.UILanguage)
{
case 0:
TextBox3.Text = "No Language specified"
break;
case 1:
TextBox3.Text = "Language is bqLangAfrikaans"
break;
case 2:
TextBox3.Text = "Language is bqLangAlbanian"
break;
case 3:
TextBox3.Text = "Language is bqLangArabic"
break;
case 4:
TextBox3.Text = "Language is bqLangBasque"
break;
case 5:
TextBox3.Text = "Language is bqLangCatalan"
break;
case 6:
TextBox3.Text = "Language is bqLangChinese_Simp"
break;
case 7:
TextBox3.Text = "bqLangChinese_Trad"
break;
case 8:
TextBox3.Text = "Language is bqLangChinese_GB2312_Sort"
break;
case 9:
TextBox3.Text = "Language is bqLangChinese_Trad"
break;
case 10:
TextBox3.Text = "Language is bqLangCroatian"
break;
case 11:
TextBox3.Text = "Language is bqLangCzech"
break;
case 12:
TextBox3.Text = "Language is bqLangDanish"
break;
case 13:
TextBox3.Text = "Language is bqLangDutch"
break;
case 14:
```

```
TextBox3.Text = "Language is bqLangEnglish"
break;
case 15:
TextBox3.Text = "Language is bqLangUKEnglish"
break;
case 16:
TextBox3.Text = "Language is bqLangEstonian"
break;
case 17:
TextBox3.Text = "Language is bqLangFinnish"
break;
case 18:
TextBox3.Text = "Language is bqLangFrench"
break;
case 19:
TextBox3.Text = "Language is bqLangGerman"
break;
case 20:
TextBox3.Text = "Language is bqLangGerman_Phonebook_Sort"
break;
case 21:
TextBox3.Text = "Language is bqLangGreek"
break;
case 22:
TextBox3.Text = "Language is bqLangHebrew"
break;
case 23:
TextBox3.Text = "Language is bqLangHungarian"
break;
case 24:
TextBox3.Text = "Language is bqLangIcelandic"
break;
case 25:
TextBox3.Text = "Language is bqLangIndonesian"
break;
case 26:
TextBox3.Text = "Language is bqLangItalian"
break;
case 27:
TextBox3.Text = "Language is bqLangJapanese"
break;
case 28:
TextBox3.Text = "Language is bqLangKorean"
break;
case 29:
TextBox3.Text = "Language is bqLangLatvian"
break;
case 30:
TextBox3.Text = "Language is bqLangLithuanian"
break;
case 31:
TextBox3.Text = "Language is bqLangMacedonian"
break;
case 32:
TextBox3.Text = "Language is bqLangMalay"
break;
case 33:
```

```
TextBox3.Text = "Language is bqLangNorwegian_Bokmal"
break;
case 34:
TextBox3.Text = "Language is bqLangNorwegian_Nynorsk"
break;
case 35:
TextBox3.Text = "Language is bqLangPolish"
break;
case 36:
TextBox3.Text = "Language is bqLangPortuguese"
break;
case 37:
TextBox3.Text = "Language is bqLangRomanian"
break;
case 38:
TextBox3.Text = "Language is bqLangRussian"
break;
case 39:
TextBox3.Text = "Language is bqLangSerbian_Cyrillic"
break;
case 40:
TextBox3.Text = "Language is bqLangSerbian_Latin"
break;
case 41:
TextBox3.Text = "Language is bqLangSlovak"
break;
case 42:
TextBox3.Text = "Language is bqLangSlovenian"
break;
case 43:
TextBox3.Text = "Language is bqLangSpanish"
break;
case 44:
TextBox3.Text = "Language is bqLangSpanish_Trad_Sort"
break;
case 45:
TextBox3.Text = "Language is bqLangSwedish"
break;
case 46:
TextBox3.Text = "Language is bqLangThai"
break;
case 47:
TextBox3.Text = "Language is bqLangTurkish"
break;
case 48:
TextBox3.Text = "Language is bqLangUkrainian"
break;
case 49:
TextBox3.Text = "Language is bqLangVietnamese"
break;
default:
TextBox3.Text = "Language Value is Not Available"
}

}
catch(e)
{
```

```
TextBox2.Text = "Caught: " + e.toString()
}

}

else
{
TextBox1.Text ="Step 2"
try
{
Application.UILanguage = eval(TextBox5.Text)
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
{
TextBox3.Text =  "UILanguage is: " + Application.UILanguage

switch(Application.UILanguage)
{
case 0:
TextBox3.Text = "No Language specified"
break;
case 1:
TextBox3.Text = "Language is bqLangAfrikaans"
break;
case 2:
TextBox3.Text = "Language is bqLangAlbanian"
break;
case 3:
TextBox3.Text = "Language is bqLangArabic"
break;
case 4:
TextBox3.Text = "Language is bqLangBasque"
break;
case 5:
TextBox3.Text = "Language is bqLangCatalan"
break;
case 6:
TextBox3.Text = "Language is bqLangChinese_Simp"
break;
case 7:
TextBox3.Text = "bqLangChinese_Trad"
break;
case 8:
TextBox3.Text = "Language is bqLangChinese_GB2312_Sort"
break;
case 9:
TextBox3.Text = "Language is bqLangChinese_Trad"
break;
case 10:
```

```
TextBox3.Text = "Language is bqLangCroatian"
break;
case 11:
TextBox3.Text = "Language is bqLangCzech"
break;
case 12:
TextBox3.Text = "Language is bqLangDanish"
break;
case 13:
TextBox3.Text = "Language is bqLangDutch"
break;
case 14:
TextBox3.Text = "Language is bqLangEnglish"
break;
case 15:
TextBox3.Text = "Language is bqLangUKEnglish"
break;
case 16:
TextBox3.Text = "Language is bqLangEstonian"
break;
case 17:
TextBox3.Text = "Language is bqLangFinnish"
break;
case 18:
TextBox3.Text = "Language is bqLangFrench"
break;
case 19:
TextBox3.Text = "Language is bqLangGerman"
break;
case 20:
TextBox3.Text = "Language is bqLangGerman_Phonebook_Sort"
break;
case 21:
TextBox3.Text = "Language is bqLangGreek"
break;
case 22:
TextBox3.Text = "Language is bqLangHebrew"
break;
case 23:
TextBox3.Text = "Language is bqLangHungarian"
break;
case 24:
TextBox3.Text = "Language is bqLangIcelandic"
break;
case 25:
TextBox3.Text = "Language is bqLangIndonesian"
break;
case 26:
TextBox3.Text = "Language is bqLangItalian"
break;
case 27:
TextBox3.Text = "Language is bqLangJapanese"
break;
case 28:
TextBox3.Text = "Language is bqLangKorean"
break;
case 29:
```

```
TextBox3.Text = "Language is bqLangLatvian"
break;
case 30:
TextBox3.Text = "Language is bqLangLithuanian"
break;
case 31:
TextBox3.Text = "Language is bqLangMacedonian"
break;
case 32:
TextBox3.Text = "Language is bqLangMalay"
break;
case 33:
TextBox3.Text = "Language is bqLangNorwegian_Bokmal"
break;
case 34:
TextBox3.Text = "Language is bqLangNorwegian_Nynorsk"
break;
case 35:
TextBox3.Text = "Language is bqLangPolish"
break;
case 36:
TextBox3.Text = "Language is bqLangPortuguese"
break;
case 37:
TextBox3.Text = "Language is bqLangRomanian"
break;
case 38:
TextBox3.Text = "Language is bqLangRussian"
break;
case 39:
TextBox3.Text = "Language is bqLangSerbian_Cyrillic"
break;
case 40:
TextBox3.Text = "Language is bqLangSerbian_Latin"
break;
case 41:
TextBox3.Text = "Language is bqLangSlovak"
break;
case 42:
TextBox3.Text = "Language is bqLangSlovenian"
break;
case 43:
TextBox3.Text = "Language is bqLangSpanish"
break;
case 44:
TextBox3.Text = "Language is bqLangSpanish_Trad_Sort"
break;
case 45:
TextBox3.Text = "Language is bqLangSwedish"
break;
case 46:
TextBox3.Text = "Language is bqLangThai"
break;
case 47:
TextBox3.Text = "Language is bqLangTurkish"
break;
case 48:
```

```
TextBox3.Text = "Language is bqLangUkrainian"
break;
case 49:
TextBox3.Text = "Language is bqLangVietnamese"
break;
default:
TextBox3.Text = "Language Value is Not Available"
}

}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

TextBox1.Text ="End " + ActionName
```

UnicodeEnabled (Property)

Action:

Application object

Description:

This property is set to true in Unicode enabled applications.

Example:

This example shows how to read and show the value of the UnicodeEnabled (Property):

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name
var ActionName = "UnicodeEnabled"
TextBox1.Text ="Start " + ActionName
if (TextBox5.Text == "")
{
TextBox1.Text ="Step 1"
try
{
TextBox3.Text =  "UnicodeEnabled is: " + Application.UnicodeEnabled
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

else
```

```
{
TextBox1.Text ="Step 2"
try
{
Application.UnicodeEnabled = eval(TextBox5.Text)
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
TextBox1.Text ="Step 3"
try
{
TextBox3.Text =  "UnicodeEnabled is: " + Application.UnicodeEnabled
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
}
TextBox1.Text ="End " + ActionName
```

UnionController (Property)

Applies To:

AppendQueriesSection object

Description:

Returns or sets the value of the Append Query union operator. This operator determines how rows are retrieved when you use the Append Query Option feature.

UnionController uses the BqUnionController constant group, which consists of the bqUnion and bqUnionAll constants value. Use bqUnion to retrieve all distinct rows selected by either query without duplicates. Use bqUnionAll to programmatically retrieve all rows selected by either query, including duplicate rows.

Action:

Read-write

Constants:

The BqUnionController constant consists of these values:

- bqUnion
- bqUnionAll values
- bqUnionIntersection
- bqUnionMinus

This is the UnionController Constant Definition:

```
typedef enum BqUnionController
{
    bqUnion = 1,
    bqUnionAll,
} BqUnionController;
```

Example:

This example shows how to append a query using the Union operator.

```
ActiveDocument.Sections["Query"].AppendQueries.Add()
ActiveDocument.Sections["Query"].AppendQueries[1].UnionController=bqUnion
```

UniqueName (Property)

Applies to:

OLAPLevelorHierarchyNew collection

Description:

This property returns the fully qualified member name as a string containing the location of the member name, including all parent members for example, "Year.Qtr1.Jan".. Where shared members are concerned, the "real" location of the member is returned, for example, "Product. 100.100-20" is always returned, even for the "Product.Diet.100-20" version of the member. The value of this property may be passed directly to the QueryLabel.Add method as the MemberName argument.

Action:

Read only, String

Example:

This example shows how to display the member name and the fully qualified (unique name) in the Console window:

```
var product = Sections["Query"].Catalog.Dimensions["Product"];
for (var i = 1; i <= product.Count; i++) {
    Console.Writeln(product.Item(i).Name);
    Console.Writeln(product.Item(i).UniqueName);
}
```

UniqueRows (Property)

Applies To:

QuerySection

Description:

Returns or sets the value of a Query section unique row property. Setting this property to true causes the query to return only unique rows of data by eliminating duplicate rows from the data set retrieved by the query.

Action:

Read-write, Boolean

Example:

This example sets each query in a document to return unique rows.

```
var SecCount = ActiveDocument.Sections.Count
for (j = 1; j <= SecCount ; j++)
     {
      if (ActiveDocument.Sections[j].Type == bqQuery)
            ActiveDocument.Sections[j].UniqueRows = true
     }
```

Unit (Property)

Applies To:

Targets object

Description:

Returns or sets the types of deviation unit for Live Line Chart target range. This property is defined by the BqLiveChartTargetUnit, which consists of the following values:

- bqLiveChartTargetUnitAbs—Sets the deviation by an absolute unit.
- bqLiveChartTargetUnitPct—Sets the deviation by a percentage value

This property is used in conjunction with the Value (Property)

URL (Property)

Applies To:

EmbeddedBrowser object, HyperLink object, PluginDocument (Interactive Reporting Web Client only)

Description:

PluginDocument—Returns the value of the URL (Uniform Resource Locator) associated with the document. If the document is registered with the OnDemand Server, the URL contains the address to the server and the name of the Broker. If the document came from a Web server or local file system, the URL contains the fully qualified server name and directory.

In Avalanche, these mapping occurs for a Hyperlink (Object):

Hyperlink “Current Window” maps to “New Window”

When an explicit Smartcut URL is entered (in the OpenURL method or Hyperlink URL property), the Smartcut may contain a “/get” or “/withnav_get” command. This behavior applies to “/get” mapping:

- In Interactive Reporting documents (BQYs): The “legacy layout is used. The appearance is that from 8.3.2, but updated colors and icons display.
- In Analyzer and Reports: The Masthead, View pane, and Menu are hidden.
- In other doc types: Only content displays.
- “/withnav_get”: The full framework displays as it would after users log into Avalanche.

> **Note:** The URL (Property) of embedded browser and hyperlink objects requires a fully qualified URL address if theInteractive Reporting document is deployed in the EPM Workspace. For example, type the URL address as: http://www.hyperion.com instead of hyperion.com. If the document is to be deployed inInteractive Reporting Studio or Oracle's Hyperion® Interactive Reporting Web Client, you may specify a truncated address.

Action:

Read-only, String

Example:

This example illustrates how to use the URL property to direct users to help information stored on the same server.

```
if(Application.Name.indexOf("BrioQuery") != -1)
   {
        Alert("This property is not valid in Hyperion")
    }
else
    {
        var MyURL = ActiveDocument.URL
        Application.OpenURL(MyURL + "\/helpinfo.html,_new")
    }
```

UseAliasTable (Property)

Applies To:

QueryOptions object

Description:

Display aliases when performing database retrievals rather than database member names. Aliases are alternate names for database members. Using an alias table, you can retrieve data that uses

the database name, which is often a stock number or product code, or an alias. Aliases can be more descriptive than database member names. An exception is thrown if this property is modified when the query is disconnected.

Action:

Read-write, Boolean

Example:

This example shows how to enable the UseAliasTable (Property) when a check box is checked.

```
if (ActiveDocument.Sections["QueryOptions - AliasTables"].Shapes["CheckBox1"].Checked)
{
    ActiveDocument.Sections["Query"].QueryOptions.UseAliasTable = true
}
else
{ActiveDocument.Sections["Query"].QueryOptions.UseAliasTable = false}
```

UseLegacyColors (Property)

As of Release 11.1.2, the "ColorScheme (Property)" on page 57 should be used instead of "UseLegacyColors" property. The UseLegacyColors (Property) as of Release 11.1.2 is hidden and retained for compatibility with the scripts created in previous versions of application

.The value of UseLegacyColors property is mapped to ColorScheme property as follows

Table 3

ColorScheme	UseLegacyColors
bqChartColorSchemeStandard	false
bqChartColorSchemeLegacy	true
bqChartColorSchemeCustom	false

Applies To:

ChartSection

Description:

For new Charts, use the colors in the Legacy Color Palette (in Release 8.3 and earlier) by setting the UseLegacyColors to true. If this property is false, new Chart use the default colors in the Default Color Palette.

If you modify Charts (for example, you drill down or change the chart type), and select the "Use legacy colors for data points" option, old, version 8.4 colors are used. Otherwise, new colors are used.

Action:

Read-write, Boolean

Example:

This example sets the UseLegacyColors property to true.

```
ActiveDocument.Sections["Chart"].UseLegacyColors=true
```

Username (Property)

Applies To:

Connection object, MetaDataConnection object

Description:

Returns or sets the value of the user name property. The user name property of connection objects refers to the user name used by the Interactive Reporting connection file (.oce).

Action:

Read-write, String

Example:

This example shows how to create a connection and set its properties.

```
var myCon = Application.CreateConnection()
myCon.Api = bqApiSQLNet
myCon.Database = bqDatabaseOracle71
myCon.HostName ="PlutoORACLE"
myCon.TimeLimit = 20  //minutes
myCon.Username = "Hyperion
myCon.SaveAs("d:\\OCEs\\PlutoORACLE.oce")
```

Value (Property)

Applies To:

Targets object

Description:

Returns or sets the value of the level of deviation unit defined for a (Live) LineChart object target range. This property is used in conjunction with theUnit (Property) Unit (Property).

Example:

This example shows how to select a Live LineChart with an area chart format, and specify a 10 percent level of deviation:

```
LineChart.Subtype = bqLiveLineChartTypeArea
LineChart.Targets.Unit = bqLiveChartTargetUnitPct
LineChart.Targets.Value = 10
```

ValueSource (Property)

Applies To:

Limit object

Description:

Returns an enumerated value that specifies where limit values originated.

Action:

Read-only

Constants:

The BqLimitValueSource constant consists of the bqLimitSourceDatabase and bqLimitSourceFile values.

Example:

This example shows how to use the ValueSource (Property) to determine the location of the limits values.

```
ActiveDocument.Sections["Query"].Limits[1].LoadFromFile("d:\\LimitData.txt")
if (ActiveDocument.Sections["Query"].Limits[1].ValueSource != bqLimitSourceFile)
     Alert("An error has occurred,Error!")
```

Variable (Property)

Applies To:

OLAPFilter object

Description:

When this property is read, it returns if a filter is variable or not. Variable filters display the Member Selection dialog box when the query is processed. When set to true this filter is marked as variable. When set to false, this filter is marked as not variable.

Action:

Read-write, Boolean

Example:

This example shows how to read and set a variable:

```
if (Sections["Query"].Filters[1].IsVariable == true) {
    // this is a variable filter
}
```

```
// Set the filter to be a variable filter
Sections["Query"].Filters[1].IsVariable = true;
```

VariableLimit (Property)

Applies To:

Limit

Description:

Enables a limit to be considered a variable limit and prompts the user for a limit value when the query is processed.

Action:

Read-write, Boolean

Example:

This example checks to see if any query limits are set as variable limits and reverts them into normal limits.

```
for (j=1 ; j <= ActiveDocument.Sections["Query"].Limits.Count; j++)
     if (ActiveDocument.Sections["Query"].Limits[j].VariableLimit == true)
           ActiveDocument.Sections["Query"].Limits[j].VariableLimit = false
```

VariableSlicerMode (Property)

Applies To:

OLAPSlicer collection

Description:

Sets the variable slicer value to a members list or a tree control on the Slicer dialog box. The members list view is useful when you want to see all members at the same level as the previously selected member. The tree control enables you to see all available parent-child slicer value relationships.

Action:

Read-write, BqSlicerDisplayOptions constant group

Constants:

The VariableSlicerMode property uses the BqSlicerDispayOptions constant group, which consists of bqListView and bqTreeView.

Example:

This example shows how to set the variable slicer mode to tree view.

```
ActiveDocument.Sections["OLAPQuery"].Slicers.VariableSlicerMode = bqTreeView
```

Version (Property)

Applies To:

Application

Description:

Returns the value of the Interactive Reporting application version number.

Action:

Read-only, String

Example:

This example shows how to display the current version number.

```
Alert (Application.Version)
```

VerticalAlignment (Property)

Applies To:

CellFormat object, Shape object, EmbeddedBrowser object, HyperLink object,

Description:

Returns or sets the vertical alignment of the text in a shape object. This property corresponds to the features on the Alignment Properties dialog box.

Action:

Read-write, BqVerticalAlignment constant

Constants:

The BqVerticalAlignment constant group consists of these values:

- bqAlignBottom
- bqAlignMiddle
- bqAlignTop

Example:

This example changes a text label to eight points, bold, italic and vertically aligns text at the top.

```
var MyLabel = ActiveDocument.Sections["Dashboard"].Shapes["TextLabel"]
MyLabel.Font.Size = 8
MyLabel.Font.Style = bqFontStyleBoldItalic
MyLabel.VerticalAlignment=bqAlignTop
```

View (Property)

Applies To:

Topic

Description:

Returns or sets the display characteristics of topics in the Data Model.

Action:

Read-Write, BqTopicView constant

Constants:

The BqTopicView constant consists of these values:

- bqDetailView
- bqIconView
- bqStructureView
- bqTopicViewNone

Example:

This example resets all the Topics in a Data Model to the structure view.

```
var TopicCount = ActiveDocument.Sections["Query"].DataModel.Topics.Count
for (j =1 ; j <= TopicCount ; j++)
ActiveDocument.Sections["Query"].DataModel.Topics[j].View = bqStructureView
```

ViewCount (Property)

Applies To:

ListBox object

Description:

Enables you to specify the number of items to display in the list control view. This allows you to page the list control view using the Object Model.

The ViewCount (Property) value is not related to the actual number of items that are displayed in the listbox. The ViewCount (Property) represents the number of items that could be displayed in the view (for example, the "page size").

The ViewCount (Property) is an integer value. For example, if "2.5" items can be displayed, then "2" is returned (the floor). The minimum value of this property is one even if less than one item could fit in the listbox.

Action:

Read Only, View Count as Number

Example:

This example sets the view count property to four.

```
ActiveSection.Shapes["ListBox1"].ViewCount=4
```

ViewIndex (Property)

Applies To:

ListBox object

Description:

Enables you to automatically scroll to selected items not shown in listbox view areas. The default property value is the index number of the first item displayed in the current list view. If the index number is set to a new index value, the listbox scrolls to the item and displays at the top of the view window. If an invalid number is specified, a "Invalid index number specified in ControlName" is thrown.

Action:

Read-Write, ViewIndex as Number

Example:

This example sets the new index position to six.

```
ActiveDocument.Sections["EIS"].Shapes["ListBox1"].ViewIndex=6
```

Visible (Property)

Applies To:

Application object, (Live) BarChart object, (Live) BlockChart object, ChartSection object, Column object, ControlsCheckBox object, ControlsCommandButton object, ControlsDropDown object, ControlsListBox object, ControlsRadioButton object, ControlsTextBox object, (Live) FunnelChart object, Hyperlink object, Legend object, Live)

LineChart, (Live) PieChart object, PivotLabelValue object, PivotSection object, (CubeQuery)QuerySection object, (Live) RadarChart object, QuerySection object, Requests collection, ReportSection object. ResultsSection object, Section object, Shape object, Slider object, Tickmark object, Title object, Toolbar object, TopicItem object, TrafficLight object, ValueLabels object

Description:

Enables the display of a base object. To hide an object, set visible as false.

Note: Do not use Toolbars["Standard"].Visible, Toolbars["Formatting"].Visible, Toolbars["Sections"].Visible, and Toolbars["Navigation"].Visible in Interactive Reporting documents to be deployed in the EPM Workspace.

Note: Do not use Application.Visible in Interactive Reporting documents to be deployed in the EPM Workspace.

Action:

Read-write, Boolean

Example:

This example unhides all the sections in a document.

```
var SecCount = ActiveDocument.Sections.Count
for ( j =1 ; j <= SecCount ; j++)
     if (ActiveDocument.Sections[j].Visible == false)
                ActiveDocument.Sections[j].Visible = true
```

Width (Property)

Applies To:

Line object, Shapes collection (Placement node), Image object

Description:

The Width (Property) determines the width in pixels of the specific image in the Resource Manager. The property is read-only, and the value returned is a non-string number between zero and the maximum width of the image. The default value of the property is the maximum width of the image.

Action:

Read-write for Line and Shape collections, **Read only** for Image object. Number

Example 1:

This example changes all the rectangles to have a border width of five pixels.

```
var ShapeCount = ActiveDocument.Sections["Dashboard"].Shapes.Count
var ShapesCol = ActiveDocument.Sections["Dashboard"].Shapes
for ( j =1 ; j <= ShapeCount ; j++)
     if (ShapesCol[j].Type == bqShapeTypeRectangle)
         ShapesCol[j].Line.Width = 5
```

Example 2:

This example shows how to show the width of a Resource Manager image in a text box:

```
//Clear all TextBoxes except TextBox5
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ActiveSection.Name

var ActionName = "RM Width"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{

TextBox1.Text ="Step 1"

try
{
TextBox3.Text =  "Width is: " +
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Width
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}
else
{
TextBox1.Text ="Step 2"

try
{
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Width =
TextBox5.Text
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
```

```
{
TextBox3.Text =  "Width is: " +
ActiveDocument.ResourceManager.Images[ListBox1.SelectedList.ItemIndex(1)].Width
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

TextBox1.Text ="End " + ActionName
```

WindowState (Property)

Applies To:

Application (Oracle's Hyperion® Interactive Reporting Studio)

Description:

Returns or sets the display of the main application window. Using the BqWindowState constant, the window can be minimized, maximized or restored back to a default state.

Note: Do not use Application.WindowState in Interactive Reporting documents to be deployed in the Oracle Enterprise Performance Management Workspace, Fusion Edition.

Action:

Read-write, BqWindowState constant

Constants:

The BqWindowState constant consists of these values:

- bqWindowStateMaximized
- bqWindowStateMinimized
- bqWindowStateNormal

Example:

This example checks if Interactive Reporting is maximized, and changes its state based on the result.

```
if( Application.WindowState != bqWindowStateMaximized)
       Application.WindowState = bqWindowStateMaximized
else
     Application.WindowState = bqWindowStateNormal
```

XOffset (Property)

Applies To:

Shapes collection (Placement node)

Description:

Enables the parent shape or control to be moved or resized. Sets or returns the position of the left-hand edge of a shape in pixels

Example:

In this example the XOffset Placement property is set to -10.

```
var objRect = ActiveSection.Shapes.CreateShape(bqRectangle)
objRect.Name = "LeftGoal"
objRect.Placement.XOffset = Field.Placement.XOffset - 10
objRect.Placement.YOffset = Field.Placement.YOffset - 10
objRect.Placement.Width = Ball.Placement.Width - 60
objRect.Placement.Height = Field.Placement.Height + 20
objRect.Fill.Color = bqBlack
```

YOffset (Property)

Applies To:

Shapes collection (Placement node)

Description:

Enables the parent shape or control to be moved or resized. Sets or returns the position of the top of a shape in pixels.

Example:

In this example the YOffset Placement property is set to -10.

```
var objRect = ActiveSection.Shapes.CreateShape(bqRectangle)
objRect.Name = "LeftGoal"
objRect.Placement.XOffset = Field.Placement.XOffset - 10
objRect.Placement.YOffset = Field.Placement.YOffset - 10
objRect.Placement.Width = Ball.Placement.Width - 60
objRect.Placement.Height = Field.Placement.Height + 20
objRect.Fill.Color = bqBlack
```

3 Constants

Constants are values that do not change and that define other, predetermined, values. The Interactive Reporting Object Model contains constants such as the following that are hardcoded in the client application:

- BqChartType
- BqColorType
- BqDataType

In the Oracle's Hyperion® Interactive Reporting Object Model, constants are identified by group name, each of which can be expanded to display their values.

This table links the constants

group and the properties and methods in which they are used.

Table 4 Constant Groups and Associated Properties/Methods

Constant Group	Associated Properties and Methods
BqAdaptiveState	AdaptiveState (Property)
BqApi	API (Property)
BqAppType	Application (Object)
BqAuditEventType	AuditSQL (Method)
BqBarLineShift	ShiftPoints (Property)
BqBarLineType	ShiftPoints (Property)
BqBorderStyle	“BorderStyle (Property) ” on page 45
BqBrushStyle	BrushStyle (Property)
BqChartAxisPlotValue	AxisPlotValues (Property)
BqChartAxisType	AxisType (Property) Trendlines (Property) Axis (Property) Add (Method)
BqChartLabelOrientation	Not Used

Constant Group	Associated Properties and Methods
BqChartType	ChartType (Property)
BqClusterBarType	Clusterby (Property), StackClusterType (Property)
BqColorType	Color (Property), MarkerBorderColor (Property), MarkerFillColor (Property)
BqColumnType	ColumnType (Property)
BqDashStyle	DashStyle (Property)
BqDatabase	Database (Property)
BqDataFunction	DataFunction (Property) Function (Property)
BqDataModelJoinsOption	Type (Property)
BqDataType	DataType (Property)
BqDbLibCancelMode	DBLibDatabaseCancel (Property)
BqDefaultFormatsColor	"BackgroundColor (Property)" on page 40, "BorderColor (Property)" on page 43
BqDrawingOrder	DrawingOrder (Property)
BqDrillDownDisplay	DrillDownDisplay (Property)
BqDrillThroughPrompt	
BqEncodng	
BqEvents	Events (Collection)
BqExportFileFormat	Export (Method), ExporttoStream (Method)
BqFillPattern	Pattern (Property)
BqFontEffect	Display (Property)
BqFontStyle	Style (Property)
BaGraphicsFileType	GraphicsFileType (Property)
BqHorizonalAlignment	Alignment (Property)
BqHTMLPageBreakUnits	HTMLVerticalPageBreakUnits (Property), HTMLExportBreakColumnCount (Property)
BqImportDataFileFormat	ImportDataFile (Method)
BqJoinType	Type (Property)
BQLanguage	UILanguage (Property)
BqLayer	Layer (Method)

Constant Group	Associated Properties and Methods
BqLimitOperator	Operator (Property)
BqLimitValueSource	ValueSource (Property)
BqLimitValueType	LimitValueType (Property)
BqLiveBarChartType	Subtype (Property)
BqLiveBlockChartType	Subtype (Property)
BqLiveChartEffect	Effect (Property)
BqLiveChartTargetUnit	Unit (Property) Unit (Property)
BqLiveLineChartType	Subtype (Property)
BqLivePieChartType	Subtype (Property)
BqLogicalOperator	LogicalOperator (Property)
BqMarkerStyle	MarkerStyle (Property)
BqMenuItem	EnableMenuItem (Method) , MenuItemEnabled (Method)
BqObjectOrientation	Orientation (Property)
BqOlapCalculationType	OlapCalculationType (Property)
BQOLAPCatalogDisplayMembers	CatalogDisplayMembers (Property) CatalogDisplayMembers (Property)
BQOLAPDrill	DefaultDrillOperation (Property)
BqOpenURLTarget	DisplayMode (Property)
BqOperator	AddFilter (Method), AddFilterValue (Method)
BqOperatorType	AddFilter (Method)
BqOrientation	Orientation (Property)
BqPageBreak	PageBreak (Property)
BqPictureEffect	Effect (Property)
BqPivotLabelDisplay	Display (Property)
BqPivotLabelType	Not Used
BqProcessEventOriginType	ProcessEventOrigin (Property)
BqProcessType	ProcessToTable (Method)
BqRefreshData	RefreshData (Property)
BqRepositoryFiletype	RepositoryFileType (Property)

Constant Group	Associated Properties and Methods
BqRepositoryObjectType	Not Used
BqRepositoryToolbarType	RepositoryBQYToolbarType (Property)
BqScriptableEvents	Type (Property)
BqScrollbarType	ShowScrollbar (Property)
BqSectionType	Type (Property)
BqShapeType	Type (Property), CreateShape (Method)
BqSlicerDisplayOptions	VariableSlicerMode (Property)
BqSortFunction	SortFunction (Property)
BqSortOrder	SortOrder (Property)
BqScrollbar	ScrollbarsAlwaysShown (Property)
BqTimeAwareIntervalUnit	View (Property)
BqToolbars	Type (Property)
BqTopicType	Type (Property)
BqTopicView	View (Property)
BqUnionController	TimeAwareLabelIntervalType (Property)
BqVerticalAlignment	VerticalAlignment (Property)
BqWindowState	WindowState (Property)

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y