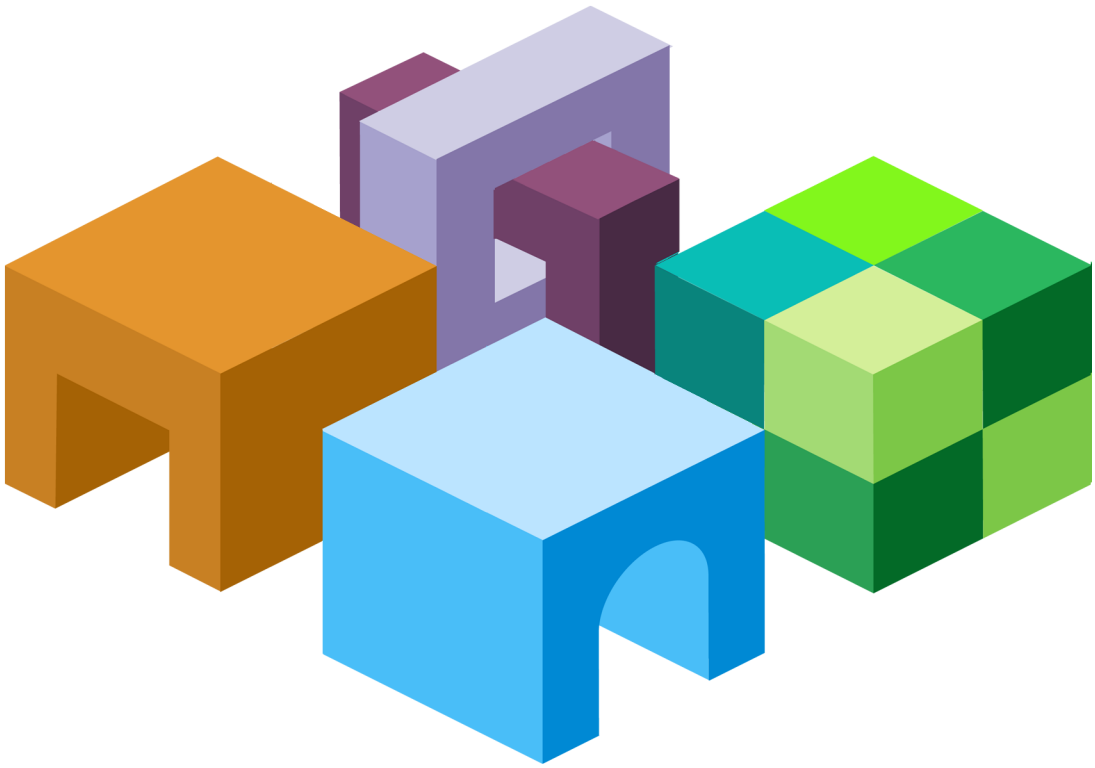

HYPERION® INTERACTIVE REPORTING

RELEASE 11.1.2

OBJECT MODEL AND DASHBOARD DEVELOPMENT SERVICES DEVELOPER'S GUIDE

VOLUME II: OBJECT MODEL GUIDE TO OBJECTS AND COLLECTIONS

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 Objects/Collections and the Object Model

Welcome to the *Volume II: Object Model Guide to Objects and Collections*. This section describes the objects associated with the Interactive Reporting Object Model (a hierarchical representation of Interactive Reporting).

Interactive Reporting exposes its functions in the Object Model using objects. An object is a programming structure that contains data and function. You manipulate objects using its properties (attributes) and methods (actions objects perform).

Objects are labeled by the component of an application to which they correspond. An object is categorized hierarchically. In the Interactive Reporting Object Model, the Application object resides at the top level. All subsequent levels are occupied by objects representing the components of the application.

Collections are multiple, similar, objects that enable you to perform interactive operations on the values that they contain.

Parameters contain all parameters passed to the Event handler.

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

2 Objects

In This Chapter

ActiveDocument (Object)

Member of:

Application object

Description:

Represents documents that users are currently viewing.

Tip: The ActiveDocument.Save() method saves a document to your local computer and not to the server.

Syntax:

Access the *active* document using this syntax:

```
Application.ActiveDocument
```

To access any document, specify this syntax:

```
Application.Documents["Document Name.bqy"]
```

Example:

This example indicates how to save and close documents under new names:

```
ActiveDocument.SaveAs("d:\\Hyperion Docs\\Updated File.bqy")
ActiveDocument.Close()
```

Methods:

Activate(), AddExportSection(SSectionName As String), Close([optional] SaveChanges as Boolean), Export([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] Prompt as Boolean), InterruptQueryProcess(), OnPostProcess(), OnPreProcess(), OnShutdown(), OnStartup(), ProcessAll() RemoveExportSections(), Save([optional] bCompressed as Boolean), SaveAs[(optional] Filename as String, [optional] bCompressed as Boolean, [optional] Prompt as Boolean], SaveToRepository, SaveToRepositoryAS (String Filename, [optional] String Filedesc, [optional] Boolean bOverride)

Properties:

Read only: Property Accessibility as Boolean, Property Path as String, Property ProcessEventOrigin as BqProcessEventOriginType, QueryInProcess as String

Read-write: Property Active as Boolean, Property Name as String, Property ShowCatalog as Boolean, Property ShowSectionTitleBar as Boolean

Collections:

Sections as Section, Events as Event, UserValues as UserValues

ActiveSection (Object)

Member of:

Application object

Description:

Because document objects record their sections, this object represents section objects in active sections. It also contains the name of current sections.

Syntax:

To expose the ActiveSection object, specify this syntax:

```
ActiveSection
```

Example:

The following example shows how to activate and duplicate the *SalesPivot* section. Oracle's Hyperion® Interactive Reporting duplicates the section and adds a section label to the Sections Catalog. The new label uses the original name, but displays the label number. For example, duplicating the chart three times displays Chart, Chart2, and Chart3 in the section pane.

```
ActiveDocument.Sections["SalesPivot"].Activate()
ActiveSection.Duplicate()
```

Methods:

Activate(), Copy(), Duplicate(), Export([optional], String Filename, [optional], BqExportFileFormat FileFormat, [optional] Boolean IncludeHeaders, [optional] Boolean Prompt), OnActivate(), OnDeactivate(), PrintOut[optional] Number FromPage, [optional] Number ToPage, [optional] Number Copies, [optional], String Filename, [optional] Boolean Prompt), Recalculate(), Remove{}

Properties:

Read-write: Property Active as Boolean, Property Name as String, Property ShowOutliner as Boolean, Property Type as BqSectionType, Property Visible as Boolean

Collections:

UserValues as UserValues, Shapes as ControlsCommandButton

AggregateLimits (Collection)

Member of:

QuerySection object

Description:

Represents items that enable you to limit a Request item that was computed using a data function in the Query section.

Setting an aggregate limit on Request items displays a divider on the Limit line. The limit icon displays to the right of the divider, indicating that the limit applied the SQL Having clause.

The AggregateLimits (Collection) is identical to the Limits (Collection) except it is used only for aggregate limits and the AvailableValues (Collection) is unavailable for use. For more information on regular limits and computed item limits, see the Limits (Collection).

Syntax:

In terms of structure and content, the AggregateLimits (Collection) is identical to the Limits (Collection). One difference is that the AggregateLimits (Collection) is used only for aggregate limits and not regular or computed item limits. Another difference is that the AvailableValues (Property) (also know as the ShowValue button in the user interface) is unavailable for computed or aggregate items.The syntax for aggregate limit items is Request.DisplayName (without reference to the Data Function). Refer to limit objects by number or name.

Computed Item Limit

Computed item limits and regular item limits share the identical syntax and collections. The one exception is the argument used with the CreateLimit(limit_item) method. The limit_item

argument type is a string for both regular limit items and computed items, and it refers to the limit item on the limit line. The regular limit item reference is a Topic.TopicItem.

For example, in a *Products table* and a *Product_Line* column, the reference is as follows:

```
CreateLimit("Products.Product_Line")
```

The syntax for the computed item limits is as follows:

```
Request.DisplayName.
```

For example, in a *DoubleSales* computed item on the Request line, the reference is:CreateLimit("Request.DoubleSales"). References other limit objects by number or name.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns item in collections at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example illustrates how to create aggregate limits, add values to the limits, display the limit names, and add limits to the limit line:

```
// Aggregate Limit
var MyLimit = ActiveDocument.Sections["Query"].AggregateLimits.CreateLimit
("Request.Dealer Price");
MyLimit.Operator =bqLimitOperatorLessThan;
MyLimit.CustomValues.Add("150");
MyLimit.SelectedValues.Add("150");
MyLimit.DisplayName = "Price";
ActiveDocument.Sections["Query"].AggregateLimits.Add(MyLimit);
```

Example 2:

This example illustrates how to create computed item limits, add values to the limits, display the limit names, and add limits to limit lines:

```
//Computed Item Limitvar MyLimit =
ActiveDocument.Sections["Query"].Limits.CreateLimit("Request.TotalSales");MyLimit.Operat
or=bqLimitOperatorGreaterThan;MyLimit.CustomValues.Add("5000");MyLimit.SelectedValues.Ad
d("5000");//MyLimit.DisplayName =
"Price";ActiveDocument.Sections["Query"].Limits.Add(MyLimit);
```

Example 3:

This example illustrates how to create a "Dealer Price" limit, add values to the limit, and add the regular item limit to the limit line:

```
//Regular Limitvar MyLimit =
ActiveDocument.Sections["Query"].Limits.CreateLimit("PCW_ITEMS.Dealer_Price");MyLimit.Op
erator
=bqLimitOperatorGreaterThan;MyLimit.CustomValues.Add("130");MyLimit.SelectedValues.Add("
130");//MyLimit.DisplayName =
"Price";ActiveDocument.Sections["Query"].Limits.Add(MyLimit)
```

Methods:

Add(Limit As Limit), CreateLimit(limitItem As String) As Limit, Item(Value as NameOrIndex) As Limit, RemoveAll()

Properties:

Read-only: Property Count As Number

Collections:

SelectedValues As LimitValues, CustomValues As LimitValues

AliasTable (Object)

Member of:

AliasTables collection

Description:

Represents an individual alias table name. The alias table names must be wrapped in an interface with a Name property so that they display properly in the script dialog object model tree.

Example:

This example shows how to return the alias table name:

```
var opts = Sections["Query"].QueryOptions;
var aliasTable = opts.AliasTables[1].Name;
```

Properties:

Read only: Name as String

AliasTable (Collection)

Member of:

QueryOptions object

Description:

A read-only collection that enables you to find out what alias tables are available in the current database.

Example:

This example shows how to read the count of alias tables:

```
var opts = Sections["Query"].QueryOptions;
if (opts.AliasTables.Count > 0) {
    // This database has some alias tables
}
```

Methods:

Item(Index as Number)—Returns a label at the given index, or with the given name if a dimension of that name is the data layout. If the index value is given as an integer, its value must be between 1 and the value returned by the Count property. If the index is not within this range, an ItemNotFound exception is thrown. If the index value is a string, it must match one of the names of the dimensions on the axis represent by this instance. If not, an ItemNotFound exception is thrown.

Properties:

Read only: Count as Number

Objects:

AliasTableName object

AppendQueries (Collection)

Member of:

QuerySection object

Description:

Represents a collection of query objects used in an appended query.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Syntax:

To expose the AppendQueries (Collection), use this syntax:

```
ActiveDocument.Sections["Query"].AppendQueries
```

The AppendQueries (Collection) includes these methods and properties:

```
ActiveDocument.Sections["Query"].AppendQueries.Add()
ActiveDocument.Sections["Query"].AppendQueries.Item()
ActiveDocument.Sections["Query"].AppendQueries.Count
```

The Requests (Collection) includes these methods and properties:

```
ActiveDocument.Sections["Query"].AppendQueries.Requests.Add(String, String)
ActiveDocument.Sections["Query"].AppendQueries.Requests.AddComputedItem(String, String,
BQDataType)
ActiveDocument.Sections["Query"].AppendQueries.Requests.Item(Value)
ActiveDocument.Sections["Query"].AppendQueries.Requests.RemoveAll()
ActiveDocument.Sections["Query"].AppendQueries.Requests.Count
```

An item of the Requests (Collection) includes these methods and properties:

```
ActiveDocument.Sections["Query"].AppendQueries.Requests[1].Remove()
ActiveDocument.Sections["Query"].AppendQueries.Requests[1].DataType
ActiveDocument.Sections["Query"].AppendQueries.Requests[1].DisplayName
ActiveDocument.Sections["Query"].AppendQueries.Requests[1].SQLName
ActiveDocument.Sections["Query"].AppendQueries.Requests[1].Visible
```

The Limits (Collection) under the AppendQuery item includes these methods and properties:

```
ActiveDocument.Sections["Query"].AppendQueries.Limits.Add(Limit)
ActiveDocument.Sections["Query"].AppendQueries.Limits.CreateLimit(String)
ActiveDocument.Sections["Query"].AppendQueries.Limits.Item(Value)
ActiveDocument.Sections["Query"].AppendQueries.Limits.RemoveAll()
ActiveDocument.Sections["Query"].AppendQueries.Limits.Count
```

Example 1:

This example illustrates how to append queries using the Union operator:

```
ActiveDocument.Sections["Query"].AppendQueries.Add()
ActiveDocument.Sections ["Query"].AppendQueries[1].UnionController=bqUnion
```

Example 2:

This example illustrates how to add *Periods* and *Quarters* to the Request line, and remove the second request line item:

```
ActiveDocument.Sections["Query"].AppendQueries.Requests.Add("Periods","Quarters")
ActiveDocument.Sections["Query"].AppendQueries.Requests[2].Remove()
```

Example 3:

This example illustrates how to place filters on the *Periods* request line item and add *Quarter 1* to the filter value:

```
MyLimit=ActiveDocument.Sections["Query"].AppendQueries.Limits.CreateLimit ("Periods")
ActiveDocument.Sections["Query"].AppendQueries.Limits[1].SelectedValues.Add("Q1")ActiveD
ocument.Sections["Query"].AppendQueries.Limits.Add(MyLimit)
```

Example 4:

This example illustrates how to count items in the AppendQueries (Collection):

```
QueryCount = ActiveDocument.Sections["Query"].AppendQueries.Count
```

Example 5:

This example illustrates how to name the appended query *MyQuery*:

```
ActiveDocument.Sections["Query"].AppendQueries[1].Name="MyQuery"
```

Example 6:

This example illustrates how to add "Periods" and "Quarters", and "Periods and "Year" to the Request line. The third line removes the second request line items.

```
ActiveDocument.Sections["Query"].AppendQueries.Requests.Add("Periods","Quarter")
ActiveDocument.Sections["Query"].AppendQueries.Requests.Add("Periods","Year")
ActiveDocument.Sections["Query"].AppendQueries.Requests[2].Remove()
```

Example 7:

This example illustrates how to return the data type, display name, SQL name, and make the base object visible.

```
RequestDT=ActiveDocument.Sections["Query"].AppendQueries.
Requests[1].DataTypeRequestDN=ActiveDocument.Sections["Query"].AppendQueries.
Requests[1].DisplayNameRequestSQLName=ActiveDocument.Sections["Query"].AppendQueries.
Requests[1].SQLNameRequestVisible=ActiveDocument.Sections["Query"].AppendQueries.Request
s[1].Visible
```

Example 8:

This example illustrates how to perform these tasks:

- Limit the *Periods* and *Quarter* request line items
- Use the equal operator
- Call the limit expression *MyLimit*
- Count the items in Collection limits

```
MyLimit=ActiveDocument.Sections["Query"].AppendQueries.Limits.CreateLimit("Periods.Quart
er")
ActiveDocument.Sections["Query"].AppendQueries.Limits[1].Operator=
bqLimitOperatorEqualActiveDocument.Sections["Query"].AppendQueries.Limits[1].SelectedVal
ues.Add("Q1")
ActiveDocument.Sections["Query"].AppendQueries.Limits.Add(MyLimit)
ActiveDocument.Sections["Query"].AppendQueries.Limits.Count
```

Example 9:

This example illustrates how to remove all items from the request line and return a count of items from the Request (Collection). The last line shows how to remove item number 1 from the request line.

```
ActiveDocument.Sections["Query"].AppendQueries.Requests.RemoveAll()
ActiveDocument.Sections["Query"].AppendQueries.Requests.Count
ActiveDocument.Sections["Query"].AppendQueries[1].Remove()
```

Methods:

Add (), Item(Value NameOrIndex) As AppendQueries

Properties:

Count As Number

Collections:

Parentheses as UnionPairs

AppendQuery (Object)

Description:

Represents an individual query object in an appended query.

Syntax:

```
ActiveDocument.Sections["Query"].AppendQueries[1].Remove()

ActiveDocument.Sections["Query"].AppendQueries[1].Name=String

ActiveDocument.Sections["Query"].AppendQueries[1].UnionController=BqUnionController
```

Example:

This example shows how to build an appended query using the Add() method to create a new query, add a limit to it, and associate the UnionController property with the Union All type:

```
ActiveDocument.Sections["Query"].AppendQueries.Add()

var newQuery = ActiveDocument.Sections["Query"].AppendQueries.Item(2)

//update new query here with request line items, filters and controller setting

ActiveDocument.Sections["Query"].AppendQueries["Query2"].Requests.Add("Table1", "Item1")

var newLimit =
ActiveDocument.Sections["Query"].AppendQueries["Query2"].Limits.CreateLimit("Stores.Stor
e_Name")
newLimit.Operator = bqLimitOperatorContains;
newLimit.LimitValueType= bqLimitValueTypeCustom;
newLimit.CustomValues.Add("B");
newQuery.Limits.Add(newLimit);

ActiveDocument.Sections["Query"].AppendQueries["Query"].UnionController = bqUnionAll;
```

Collections

Limits As Limit, :Requests As Requests

Constants:

The AppendQuery (Object) references the UnionController property which uses BqUnionController constant group. The BqUnionController consists of the following values:

- bqUnion = 1

- bqUnionAll = 4
- bqUnionIntersection = 2
- bqUnionMinus = 3

Application (Object)

Description:

Represents the entire Interactive Reporting application and contains these items:

- Application-wide settings and options
- Methods that return top-level objects, such as ActiveDocument
- Properties that return top-level objects, such as ActiveDocument

Do not use this object model syntax in Interactive Reporting to be deployed in Oracle Enterprise Performance Management Workspace, Fusion Edition:

- Application.Alert()
- Application.CreateConnection()
- Application.DoEvents()
- Application.LoadSharedLibrary()
- Application.Quite()
- ApplicationShell()
- Application.ShowMenuBar
- Application.ShowStatusBar
- Application.StatusText
- Application.Visible
- Application.WindowState

Example 1:

The *quit* method is called from the Application object.

```
Application.Quit()
```

Example 2:

This example illustrates how to display the Paging Toolbar if the application is EPM Workspace.

if (Application.Type = bqAppTypeThinClient)

```
{
Toolbars["Paging"].Visible = true
}
else
{
```

```
Toolbars["Standard"].Visible = true
}
```

If the application is EPM Workspace, then the Standard Toolbar is displayed.

Note: The Application.Quit() method applies only to Oracle's Hyperion® Interactive Reporting Studio and not to Oracle's Hyperion® Interactive Reporting Web Client.

Methods:

Number Alert(String Prompt, [optional] StringTitle, [optional] StringButton1Text, [optional] String Button2Text, [optional] String Button3Text) Function CreateConnection() As Connection, Function DoEvents(), ExecuteBScript(Script As String), Function LoadSharedLibrary(Name As String) As SharedLibrary, OpenURL(URL as as String,Target as String), Quit([optional] PromptBeforeQuitting as Boolean), SendSQL(Ocename As String, Username As String, Password As String, SQLString As String), Function Shell(Command as string, [optional] Arguments as String)

Properties:

Read-only: Property ActiveDocument As Document, Property Console As Console, Property CurrentDir as String, Property Documents As Documents, Property Name As String, Property PathSeparator As String, Property RecentFiles As RecentFiles, Property Toolbars As Toolbars, Property Type as BQAppType, Property Version As String

Read-write: Property CurrentDir As String, Property Name as String, Property ResetPrintProperties as Boolean, Property ShowCustomMenu As Boolean, Property ShowMenuBar as Boolean, Property ShowStatusBar As Boolean, Property StatusText As String, Property Visible As Boolean, Property WindowState As BqWindowState

Collections:

Documents as Documents, Toolbars as Toolbars, RecentFiles as RecentFiles

Objects:

ActiveDocument as Document, Console as Console, ActiveSection As Section, Session as Session

AreaChart (Object)

Member of:

ChartSection object

Description:

Represents all area chart properties

Example:

This example illustrates how to have an Area chart fill the area under the ribbon. Hyperion assumes that *Chart* is the Chart report in the active document:

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

Properties:

Read-write: Property FillUnderRibbon As Boolean

Association (Collection)

Member of:

Slider object

Description:

Represents all association objects, methods and properties. This collection is equivalent to setting options on the Association dialog box. An association object refers to the link between a Slider and its subordinate gauges and (Live) Chart objects. When using the Association object to add a link between a Slider dimension and a gauge and (Live) Chart, the Slider must use the same data set used by the gauge or (Live) Chart. A Slider can control multiple gauges or (Live) Charts in the same Dashboard section.

Example:

This example shows how to associate a (Live) BarChart object with a Slider:

```
Slider.Association.Add("BarChart")
```

Methods:

Add(Value as String), Item(Value as NameOrIndex), Remove(NameOrIndex as Value), RemoveAll()

Properties:

Read only: Count as Number

Objects:

Association object

Association (Object)

Member of:

Association collection

Description:

Represents the Association object that has been associated with the Slider. This object is the equivalent of the selected gauge or (Live) Chart name in the Association dialog box.

Example:

This example shows to return the name of Association object linked with the Slider in an Alert box:

```
Alert(Slider.Association["BarChart"].Name)
```

Methods:

None

Properties:

Read only: Namae as String

AvailableValues (Collection)

Member of:

LimitValues object

Description:

Represents the list of all valid limit criteria. See Limits (Collection).

Note: The Add (Method) is unavailable for AvailableValues (Collection) because values are in the database. For the "CustomValues (Collection)" on page 60, the Add (Method) adds a value to the list. For the "SelectedValues (Collection)" on page 198, the Add (Method) adds a value to the selected list. The AddAll (Method) of the SelectedValues (Collection) selects all values of AvailableValues or CustomValues (Collection).

Action:

Read-only

Example:

This example illustrates how to add all values in AvailableValues (Collection) to the SelectedValues (Collection). This is the same as performing a select all values and transferring the selection in the Limit User Interface:

```
LimitCount =  ActiveDocument.Sections["Results"].Limits[1].AvailableValues.Countfor
(i=1;i<=LimitCount;i++){MyVal =
ActiveDocument.Sections["Results"].Limits[1].AvailableValues[i]ActiveDocument.Sections["
Results"].Limits[1].SelectedValues.Add(MyVal)}
```

Methods:

Add(ValueItem as Value), AddAll(), Item(Number Index), RemoveAll()

Properties:

Read Only: Property Count as Number

AxisItems (Collection)

Now CategoryItems (Collection). See "LimitValues (Collection)" on page 123.

AxisLabels (Collection)

Member of:

ChartSection object

Description:

Represents labels for a chart axis, and maps to the Chart Outliner.

The AxisLabels (Collection) is instantiated three times for each Chart Section object in the form: XLabels, YLabels, and ZLabels.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent the call to the Item (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example illustrates how count labels on the X-axis:

```
ActiveDocument.Sections["AllChart"].XLabels.Count
```

Example 2:

In this example, the Axis labels are left-justified and rotated vertically:

```
ActiveDocument.Sections["Chart"].XLabels.Orientation = bqChartLabelOrientationVertical
ActiveDocument.Sections["Chart"].XLabels.Justification = bqLeftJustified
```

Methods:

DrillInto(ItemNameOrIndex, DrillName As String), FocusSelection(ItemArray), HideSelection(ItemArray), UnhideAll()

Properties:

Read-only: Property Count As Number

BarChart (Object)

Member of:

ChartSection object

Description:

Represents all bar chart properties:

Example:

This example illustrates how to enable bar values of bar charts:

```
ActiveDocument.Sections["Chart"].BarChart.ShowBarValues = true
```

Properties:

Read-write: Property ClusterBy As BqClusterBarType, Property ShowBarValues As Boolean

(Live) BarChart (Object)

Member of:

Dasbhoard objects

Description:

Represents a (Live) BarChart object and its properties.

The following (Live) BarChart object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Orientation—BqObjectOrientation
- Subtype—BqLiveBarChartType
- Type—BqShapeType

Example:

The example shows how to remove all fact items, and adds the fact item “Unit Sales”:

```
BarChart.Facts.RemoveAll()
BarChart.SourceSectionName = "Results"
BarChart.Facts.Add("Unit Sales")
```

Methods:

None

Properties:

Read-write:AccessibilityText (Property), AppendObjectText (Property), Comments as String, Effect as BqLiveChartEffect, Locked as Boolean, Name as String, Orientation as BqObjectOrientation, SourceSectionName as String, Subtype as BqLiveBarChartType, Visible as Boolean

Read only: Type as BqShapeType

Objects:

UserValues collection, Placement objects, Data object, Subcomponents object

BarLineChart (Object)

Member of:

ChartSection object

Description:

Represents a BarLineChart object and its properties.

Example:

This example shows how to change Barline Chart properties:

```
ActiveDocument.Sections["Chart"].BarLineChart.ShowBarValues = true
ActiveDocument.Sections["Chart"].BarLineChart.StackClusterType=bqBarLineCluster
ActiveDocument.Sections["Chart"].BarLineChart.ClusterBy = bqClusterByY
ActiveDocument.Sections["Chart"].BarLineChart.IgnoreNulls = false
ActiveDocument.Sections["Chart"].BarLineChart.ShiftPoints = bqShiftCenter
```

Properties:

Read-write: Property ClusterBy As BqClusterBarType, Property IgnoreNulls as Boolean, Property ShiftPoints as BqBarLine Shift, Property ShowBarValues As Boolean, Property StackClusterType as BqBarLineType

BeginQuery (Object)

Member of:

Parentheses object

Description:

Represents the beginning query of a Parentheses object expression. The Parentheses object expression nests parentheses () for the evaluation order of an appended query. The expression must include two queries or more. The default evaluation order is left to right.

Example:

This example shows how to set the union controller for the queries within the parenthetical expression to a "union all controller":

```
ActiveDocument.Sections["Query"].AppendQueries["Parentheses"]
["Query,Query2"].BeginQuery.UnionController = bqUnionAll
```

Methods:

Remove()

Properties:

Read-write: Name as String, UnionController as BqUnionController

Collections:

Requests collection, Limits collection

(Live) BlockChart (Object)

Member of:

Dashboard objects

Description

Represents a (Live) BlockChart object and its properties. BlockCharts objects can be shown in a cone or pyramid format

The following BarChart object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Orientation—BqObjectOrientation
- Subtype—BqLiveBlockChartType
- Type—BqShapeType

Example:

This example shows how to set several common (Live) BlockChart attributes, and

```
BlockChart.Comments = "Market Share"
BlockChart.Explode = true
BlockChart.Locked = false
BlockChart.Orientation = bqObjectOrientationVertical
BlockChart.Name = "European Market"
```

```
BlockChart.SourceSectionName = "Results"
BlockChart.Subtype = bqLiveBlockChartTypeCone
BlockChart.Visible = true
BlockChart.Facts.RemoveAll()
BlockChart.Facts.Add("Amount Sales")
```

Methods:

None

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property),: Comments as String, Effect as BqLiveChartEffect, Explode as Boolean, Locked as Boolean, Name as String, Orientation as BqObjectOrientation, SourceSectionName as String, Subtype as BqLiveBlockChartType, Visible as Boolean

Read only:: Type as BqShapeType

Objects:

UserValues collection, Placement object, Data object, Subcomponents object

Body (Object)

Member of:

ReportSection object

Description:

The Body (Object) represents the attributes that you typically display in report body sections.

Example:

This example illustrates how to spring and unspring a table and chart in the body section:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Spring("Chart")
ActiveDocument.Sections["Report"].Body.Charts["Chart"].UnSpring()
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects/Coll.:

LineFormat, FillFormat object, Tables, Fields, Shapes, Pivots, Chart CategoryItems (Collection)

Member of:

ChartSection object

Description:

Represents items for a Chart axis and maps directly to the Chart outliner.

The CategoryItems (Collection) is instantiated three times in a Chart Section as XCategories, Facts, and ZCategories.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

In this example, a chart is built using the request items specified in the query. All items in the outliner are removed; then specific items are added to the outliner:

```
ActiveDocument.Sections["Chart"].XCategories.RemoveAll()
ActiveDocument.Sections["Chart"].Facts.RemoveAll()
ActiveDocument.Sections["Chart"].XCategories.Add("Year")
ActiveDocument.Sections["Chart"].Facts.Add("Unit Sales")
```

or

```
for (I=1;I< ActiveDocument.Sections["Chart"].XCategories.Count; I++)
   ActiveDocument.Sections["Chart"].XCategories.Remove(I)
for (I=1;I< ActiveDocument.Sections["Chart"].Facts.Count; I++)
   ActiveDocument.Sections["Chart"].Facts.Remove(I)
   ActiveDocument.Sections["Chart"].XCategories.Add("Year")
   ActiveDocument.Sections["Chart"].Facts.Add("Unit Sales")
```

Methods:

Add(ItemName As String), AddComputedItem(Name As String, Expression As String, [optional] Index As Number), Item(Value as NameOrIndex) As AxisItem, Remove(Value NameOrIndex), RemoveAll()

Properties:

Read-only: Property AxisType As BqChartAxisType, Property Count As Number

bqoEvent (Object)

Member of:

Parameters

Description:

Passes event-related information as an optional parameter to the Event handler (OnClick handler). The bqoEvent has two properties: ClickX and ClickY. When the OnClient event occurs, the event handler has access to the event related information needed to process it. For example, the event might require the position of the mouse cursor for a picture, OnClick event, or information about which table column was clicked for a table embedded section object. If ClickX and ClickY values cannot be read, 0 is returned.

Note: In the Object Model hierarchy, Parameters contains all parameters passed to the handler. Even though JavaScript allows an arbitrary numbers of parameters to be passed to a function, the Object Model event handler is not a trure JavaScript function.It is an external function that ishandled by the current Object Model framework.. Consequently, it is subject to the restrictions imposed by the Object Model framework.

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"
```

BubbleChart (Object)

Member of:

ChartSection object

Description:

Enables you to set and retrieve bubble chart attributes. Bubble charts are typically used to display three data dimensions in two dimensional charts. Bubble charts are often used to display financial data because you can differentiate values by bubble size. This like using scatter chart to plot data as a collection of bubbles.

Example:

This example illustrates how to display negative values in bubble charts:

```
ActiveDocument.Sections["Chart2"].BubbleChart.ShowNegativeValues = true
```

Properties:

Read-write: MaxBubbleSize As Number, ShowNegativeValues as Boolean, ShowZeroBubbles as Boolean

Note: When the Chart type is set to Bubble, only items appropriate for Bubble chart display in the Script Editor Object Model browser.

Bullet (Object)

Dashboard objects, Shapes collection

Description:

Represents an individual bullet gauge in the Dashboard section.

Example:

This example shows how to display the title of the bullet gauge, hide the legend, make tickmarks visible, and enable autoscaling:

```
Bullet.Title.Text = "Sales Bullet"
Bullet.Legend.Visible = false
Bullet.TickMark.Visible = true
Bullet.NumericRange.AutoScale = true
```

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Locked as Boolean, SourceSectionName as String, Subtype as String, Visible as Boolean

Read only::Type as Number

Objects and Collections:

UserValues collection, Placement object, Subcomponents object, Themes collection, Data object

Categories (Collection)

Member of:

Data object

Descriptions:

Represents all category objects, methods and properties. Category objects represent label or qualitative objects, and correlate with the XCategories collection of legacy charts.

Example:

This example shows how to remove the slider category object "Amount Sales", and adds the category object "Country":

```
Slider.Categories.Remove("Amount Sales")
Slider.Categories.Add("Country")
```

Methods:

Add(ItemName as String), Item(Value as NameOrIndex), Remove(Value NameOrIndex), RemoveAll

Properties:

Read only: Count as Number

Category (Object)

Member of:

Categories collection

Description:

Represents an individual Category object. Category objects represent label or qualitative objects, and correlate with the XCategories collection of legacy charts.

Note: A (Live) BlockChart object references a "Wedge" category.

Example:

This example shows how to display theCategory object name of an Slider in an Alert box:

```
Alert(PieChart.Categories["Unit Sales"].Name)
```

Methods:

None

Properties:

Read only: Name as String

Objects:

None

CategoryAxis (Object)

Member of:

Subcomponents object

Description:

Represents attributes of the X and Y axes of a (Live) Chart object, including the associated X/Y axis titles. Setting properties for this object is the equivalent of setting properties for the Category Axis Label and Category Axis Title on the Font dialog box, and the Label Axis fields on the properties dialog box of the specific chart.

Example:

This example shows how to set the category axis title to “Market Share”, enables the visible property, and sets the title to red:

```
BarChart.CategoryAxis.Title.Text = "Market Share"
BarChart.CategoryAxis.Title.Visible = true
BarChart.CategoryAxis.Title.Font.Color = 16711680
```

Methods:

None

Properties:

None

Objects:

Labels object, Font object

CellFormat (Object)

Member of:

GridFormat object

Description:

Represents an individual CellFormat object (data value, label) in the Pivot and OLAPQuery (both CubeQuery and OLAPQuery) sections. Setting properties for an CellFomat object corresponds to setting the properties for a data value, font or label on the Default Formats dialog box. Default format are applied to the new sections of the current Interactive Reporting document (BQY) and new documents.

Example:

This example shows hot to set the background color for a Pivot section label to blue, and set the border style for the labels to horizontal.

```
DefaultFormats.Pivot.Labels.BackgroundColor = bqDefaultFormatsBlue
DefaultFormats.Pivot.Labels.BorderStyle = bqBorderStyle_Horizontal
```

Properties:

BackgroundColor as BqDefaultFormatsColor, BorderColor as BqDefaultFormatsColor, BorderStyle as BqBorderStyle, HorizontalAlignment as BqHorizonAlignment, VerticalAlignment as BqVerticalAlignment

Objects:

Font as Font

Note: Only the bqFontEffectUnderline from the Effect property of Font object is applicable for DefaultFormats. All other FontEffects are ignored.

Chart (Object)

Member of:

Shapes (Collection)

Description:

Represents an embedded Chart section object within the Shapes collection .

Example:

This example illustrates how to display the scrollbar in embedded Charts.

```
ActiveDocument.Sections["Dashboard"].Shapes["Chart1"].OnClick()
```

```
ActiveDocument.Sections["Dashboard"].Shapes["Chart1"].ShowScrollbar =
bqScrollbarTypeAlways
```

Methods:

OnClick()

Properties:

Read-only: Property Name as String, Property Type as BqShape

Read-write: Property Alignment as BqHorizontalAlignment, Property ShowScrollbar as BqScrollbarType, Property Text as String, Property VerticalAlignment as BqVerticalAlignment, Property Visible as Boolean

ChartSection (Object)

Member of:

Sections collection, Document object (ActiveSection)

Description:

Represents a chart section.

Example:

This example activates the *Sales Chart* section, activates the legend, changes the title to *International Sales Report*, changes the chart to horizontal bar, and exports the chart as *intlchrt.htm*:

```
myChart = ActiveDocument.Sections["Sales Chart"]
myChart.Activate()
myChart.ShowLegend = true
myChart.Title = "International Sales Report"
myChart.ChartType = bqChartTypeHorizontalBar
myChart.Export("c:\\html\\intlchrt.htm",bqExportFormatHTML,true)
```

Methods:

Activate(), Copy(), Duplicate(), Export([Filename As String], [FileFormat As BqExportFileFormat], [IncludeHeaders As Boolean], [Prompt as Boolean]), PivotThisChart() As PivotSection, PrintOut([FromPage As Number], [ToPage As Number], [Copies As Number], [Filename As String], [Prompt As Boolean]), Recalculate(), RefreshDataNow(), Remove()

Properties:

Read-only: Property Active As Boolean, Property LastPrinted as Date, Property Type As BqSectionType

Read-write: Property AutoResize as Boolean, Property AutoRotate as Boolean, Property ChartType As BqChartType, Property ColorScheme as qChartColorScheme, Property HTMLBoundaryHeight as Number, Property HTMLBoundaryMode as Boolean, Property

HTMLBoundaryWidth as Number, Property HTMLDisplayXViews as Boolean, HTMLDisplayZViews as Boolean, HTMLMaximumXBarsDisplayed as Number, HTMLMaximumZBarsDisplayed, Property HTMLSyncScrollingProps as Boolean, , Property MinimumFontSize as Number, Property Name As String, Property PrintAllViews as Boolean, Property RefreshData as BqRefreshData, Property RotationHorizontal as Number, RotationVertical as Number, Property Show3DObjects As Boolean, Property ShowBackPlane as Boolean, Property ShowBorder As Boolean, Property ShowHorizontalPlane As Boolean, Property ShowLegend As Boolean, Property ShowOutliner As Boolean, Property ShowPartialViewIndicator as Boolean, Property ShowSortLine as Boolean, Property ShowSubTitle As Boolean, Property ShowTitle As Boolean, Property ShowVerticalPlane As Boolean, Property SmartScaling as Boolean, Property SubTitle As String, Property Title As String, Property UseLegacyColors as Boolean, Property Visible As Boolean

Collections:

XCategories As CategoryItems, Facts As CategoryItems, ZCategories As CategoryItems, XLabels As AxisLabels, YLabels As AxisLabels, ZLabels As AxisLabels, UserValues as UserValues

Objects:

AreaChart As AreaChart, BarChart As BarChart, BarLineChart As BarLineChart, ColorSet, LabelsAxis As LabelsAxis, Legend as Legend, LineChart As LineChart, PieChart As PieChart , ValuesAxis As ValuesAxis, Legend As Legend

ColorRange (Object)

Member of:

ColorRanges (Collection)

Description:

Returns or sets the color for the numeric range used in the gauge. By default the new color range object has a transparent color in the numeric range to which it has been assigned.

Note: To use the minimum and maximum values used by the numeric range, do not define values for the ColorRange (Object) Max (Property) or Min (Property).

Example:

This example shows how to add the color range, specify a color range from 0 to 50 and add the tool tip: “Amount Sales.

```
Speedometer.ColorRanges[1].Color = "16711680"
Speedometer.ColorRanges[1].Max = "50"
Speedometer.ColorRanges[1].Min = "0"
Speedometer.ColorRanges[1].Tooltips = "Amount Sales"
```

Methods:

RemoveRange()

Properties:

Read-write: Property Color as Number (use BqColorType), Property Max as Number, Property Min as Number, Property Tooltip as String

ColorRanges (Collection)

Member of:

Subcomponents object

Description:

Represents a collection of color range objects associated with gauges in the Dashboard section. Color ranges are assigned to a scale of numeric values in the gauge.

Example:

This example show to add a newcolor range object. By default the new color range object has a transparent color in the numeric range to which it has been assigned.

```
ActiveDocument.Sections["Dashboard"].Shapes["Speedometer"].ColorRanges.Add()
```

Methods:

Add, Item(Value as name or index), RemoveAll()

Properties:

Read only: Property Count as Number

ColorSet (Object)

Member of:

ChartSection object

Description:

Represents the individual colors included in the color scheme for a chart section object based on either a Standard, Legacy or Custom color scheme. The ColorSet object allows you to change a color element when the Chart color scheme is set to bqChartColorSchemeCustom (for a custom chart color scheme). It does not allow you to add or remove color element, and as a result, the size of the collection is always equal to seventeen.

Example:

This example shows you how to set the chart color scheme to a custom color scheme, changes the first color in the scheme from number 1 to number 10 (from blue to black), and applies the color changes.

```
ActiveDocument.Sections["Chart3"].ColorScheme = bqChartColorSchemeCustom
ActiveDocument.Sections["Chart3"].Colors.Color1 = 10
ActiveDocument.Sections["Chart3"].Colors.ApplyColors()
```

Methods:

ApplyColors()

Properties:

Read-write: Color1–Color17

Colors (Collection)

Member of:

Targets object

Description:

Represents all colors objects associated with the target range of a (Live) LineChart object.

Example:

This example shows how to read the number of colors associated with a target range, and writes it to the Console window:

```
Console.Writeln(LineChart.Targets.Colors.Count)
```

Methods:

Item(Value as NameOrIndex)

Properties:

Read only: Count as Number

Column (Object)

Member of:

TableSection object, ResultsSection object

Description:

Represents a column within a Table or Results section.

Example 1:

This example illustrates how to populate a Dropdown list in a Dashboard section with Results column data. It assumes that you have *CommandButton* (button) and *Dropdown* (drop-down list) in the Dashboard section:

```
//Code behind the "CommandButton"
var NumRows = ActiveDocument.Sections["Results"].RowCount
for (I =1 ; I <= NumRows;I++)
DropDown.Add(ActiveDocument.Sections["Results"].Columns[1].GetCell(I))
```

Example 2:

This example shows how to change the number format of all numeric columns in a Results section:

```
var NumColumns=ActiveDocument.Sections["SalesResults"].Columns.Count
for (I=1; I<=NumColumns;I++)
{
var MyCol=ActiveDocument.Sections["SalesResults"].Columns.Item(I)
MyCol.ResizeToBestFit()
   if (MyCol.DataType = bqDataTypeNumber)
   MyCol.NumberFormat = "0.00"
}
```

Methods:

CreateDateGroup(), GetCell(nRow as Number), Remove(), ResizeTo BestFit()

Properties:

Read-only: Property ColumnType As BqColumnType, Property DataType As BqDataType, Property Index As Number, Property Name As String

Read-write: Property Alignment As BqHorizontalAlignment, Property NumberFormat As String, Property SupressDuplicates As Boolean, Property TextWrap As Boolean, Property Visible As Boolean

Columns (Collection)

Member of:

TableSection object, ResultSection object

Description:

Represents all columns in Table or Results sections.

Tip: All collections have a "Item(NameOrIndex)" method. This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to Item (Method). For example, these statements cause identical in behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to add a computed column, *MyComputed* to the Results section. This example includes strings and numeric calculations in the same computed columns:

```
var MyResults = ActiveDocument.Sections["Results"]
var NumColumns = MyResults.Columns.Count
var Expression = ("Number of Columns="+Number(NumColumns+1))
MyResults.Columns.AddComputed("MyComputed", Expression)
```

Methods:

Add(Name As String) As Column, AddComputed(Name As String, Expression As String) As Column, Item(NameOrIndex) As Column, ModifyComputed(NameOrIndex, Expression As String) As String, RemoveAll()

Properties:

Read-only: Property Count As Number

Connection (Object)

Member of:

Global object or Data Model object

Description:

The Connection object gives access to information about the connection to the database. It returns the same interface and object as the existing section's Connection object. It provides this information:

- Connection File (OCE) or database connection
- MetaData connection information for a data model
- Stand-alone object representing an OCE. Call the CreateConnection (Method) to create the object.

Example 1:

This example illustrates how to connect Data Models to their associated database and process queries. This example assumes that a connection file is associated with the Data Model:

```
//Check to make sure the connection has an associated OCE
```

```
if(ActiveDocument.Sections["Query"].DataModel.Connection.Filename != "")
{
with(ActiveDocument.Sections["Query"].DataModel.Connection)
{
Username = "hyperion"
SetPassword("HyperionHyperion")
Connect()
}
ActiveDocument.Sections["Query"].Process()
}
else
{
Alert("Your DataModel does not have an OCE","Information")
}
```

Example 2:

This example shows how to create an OCE and save 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")
```

Example 3:

This example shows how to connect to sample OCE "Query", is in the Foundation repository:

```
MyOCE = ActiveDocument.Sections["Query"].DataModel.Connection.Filename
```

Example 4:

This example shows how to determine if there is a connection to Oracle Essbase, and how to show the results.

```
Sections["Query"].Connection.Connect();
if (Sections["Query"].Connection.Database == bqDatabaseEssbase)
    Console.Writeln("Connected to Essbase");
Console.Writeln(Sections["Query"].Connection.CubeName);
```

Method

Connect([optional] GetCredentials as Boolean), Disconnect(), Open(Filename As String), Save(), SaveAs(Filename As String), SetPassword(Password As String), UseAlternateMetadataLocation(Value As Boolean, [optional] MetadataOce As String)

Properties:

Read-only: Property Connected As Boolean, Property Filename As String

Read-write: Property AllowNonJoinedQueries As Boolean, Property Api As BqApi, Property AutoCommit As Boolean, Property Database As BqDatabase, Property DataBaseList As String, Property DBLibAllowChangeDatabase As Boolean, Property DBLibApiSeverity As Number, Property DBLibDatabaseCancel As BqDbLibCancelMode, Property DBLibPacketSize As Number, Property DBLibServerSeverity As Number, Property DBLibUseQuotedIdentifiers As Boolean, Property DBLibUseSQLTable As Boolean, Property EnableAsyncProcess As Boolean, Property EnableTransactionMode As Boolean, Property HostName As String, Property MetadataPassword As String, Property MetadataUser As String, Property MetaFileChoice As String, Property ODBCDatabasePrompt As Boolean, Property ODBCEnableLargeBufferMode As Boolean, Property SaveWithoutUsername As Boolean, Property ShowAdvanced As Boolean, Property ShowBrioRepositoryTables As Boolean, Property ShowMetadata As Boolean, Property SpecificMetadataLogin As Boolean, Property SQLNetRetainDateFormats As Boolean, Property StringRetrieval As Boolean, Property TimeLimit As Number, Property Username As String

Console (Object)

Member of:

Application object

Description:

Represents the Console window.

Note: Do not use this object model syntax for Interactive Reporting documents to be deployed in EPM Workspace:

Console.Write() Console.WriteLn()

Example 1:

This example shows how to display the names of document sections in the Console. Print names on new lines using Carriage Return "\r" and New Line "\n". The example uses the Write (Method):

```
for(I=1 ; I <= ActiveDocument.Sections.Count; I ++)
        Console.Write (ActiveDocument.Sections[I].Name+"\r\n")
```

Example 2:

This example shows how to print document section names on individual lines using the Writeln method:

```
Console.Writeln(ActiveDocument.Name +"'s sections are: ")
for (j=1 ; j < ActiveDocument.Sections.Count ; j++)
Console.Writeln("Section #"+j +" = " +ActiveDocument.Sections[j].Name)
```

Method:

Write(OutputData), Writeln (OutputData and new line after the inserted text)

Control (Object)

Member of:

Controls collection

Description:

Represents a control. All controls are inherited from this basic object. Consequently, the Control (Object) is not called.

Controls (Collection)

Member of:

DashboardSection object

Description:

Contains all the control objects for a Dashboard section. Use Controls (Collection) to access a specific control object. All control object have common controls and methods, but properties that are specific to control returned.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to Item (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to enable disabled controls in Dashboard sections:

```
var ControlCount = ActiveDocument.Sections["Dashboard"].Controls.Count
for (I = 1 ; I <= ControlCount; I++)
{
// if the control is disabled then enable it
   if (ActiveDocument.Sections["Dashboard"].Controls[I].Enabled != true)
      ActiveDocument.Sections["Dashboard"].Controls[I].Enabled = true
}
```

Method:

Item(NameOrIndex) As Control

Properties:

Read-only: Property Count As Number

ControlsCheckBox (Object)

Member of:

Controls collection, DashboardSection object

Description:

The ControlsCheckBox (Object) represents a Dashboard check box that enables user to make choices.

Example:

This example shows how to change check box text and to determine if it is enabled. This script assumes a check box control,*CheckBox* in a Dashboard section:

```
CheckBox.Text = "Click here to change the value"
//if the CheckBox is not being shown, show it.
if (CheckBox.Visible != true)
      CheckBox.Visible = true
if(CheckBox.Checked == true)
      Alert("Checkbox is Checked","Info")
else
      Alert("Checkbox is Not Checked","Info")
```

Method:

OnClick(), OnClientClick()

Properties:

Read-only: Property Name As String, Property Type as BqShapeType

Read-write: AccessibilityText (Property), Property Alignment As BqHorizontalAlignment, AppendObjectText (Property), , Property Checked As Boolean, Property ClientScriptStatus as Boolean, Enabled As Boolean, Property Text As String, Property Type As BqShapeType, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Objects:

Fill As Fill, Font As Font

ControlsCommandButton (Object)

Member of:

Controls collection, DashboardSection object

Description:

Represents a Dashboard button.

Example:

This example shows how to change the text, font type, and font size of Command buttons:

```
CommandButton.Text = "Click Here"
CommandButton.Font.Name = "Courier"
CommandButton.Font.Size = 12
```

Method:

OnClick()

Properties:

Read-only: Property Name As String, Property Type As BqShapeType

Read-write: AccessibilityText (Property), AppendObjectText (Property), Enabled As Boolean, Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Object:

UserValues collection, Font As Font

ControlsDropDown (Object)

Member of:

Controls collection, DashboardSection object

Description:

Represents Dashboard drop-down list objects that enable users to select data.

Example:

This example shows how to populate a dropdown list from a query limit:

```
// Connect to the database to ensure showvalues will work
ActiveDocument.Sections["Query"].DataModel.Connection.Username = "hyperion"
ActiveDocument.Sections["Query"].DataModel.Connection.SetPassword("Hyperion
Hyperion")
ActiveDocument.Sections["Query"].DataModel.Connection.Connect()
//Load the list of Available Values
ActiveDocument.Sections["Query"].Limits[1].RefreshAvailableValues()
var ValueCount =
ActiveDocument.Sections["Query"].Limits[1].AvailableValues.Count
//Remove All Items from the DropDown
DropDown.RemoveAll()
 for (I = 1; I <= ValueCount; I ++)
```

```
{
DropDown.Add(ActiveDocument.Sections["Query"].Limits[1].AvailableValues[I])
}
```

Method:

Add(Value As String), Item(Index As Number), OnClick(), OnClientClick(), OnClientSelection(), OnSelection(), Remove(Index As Number), RemoveAll(), Select(Index As Number)

Properties:

Read-only: ,Property Name As String, Property Type As BqShapeType

Read-write: AccessibilityText (Property), Alignment (Property) As BqHorizontalAlignmentAppendObjectText (Property), AppendObjectText (Property), Property ClientScriptStatus as Boolean, Property Count As Number, Property Enabled As Boolean, Property SelectedIndex As Number, Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Object:

UserValues (Collection), Font As Font

ControlsListBox (Object)

Member of:

Controls collection, DashboardSection object

Description:

Represents a Dashboard list box that enables you to select items.

Example:

This example shows how to clear the values in a list box and use values from a results column. This example uses JavaScript sorting functions that sort data before populating the control:

Note: JavaScript arrays are 0 based; all collections are 1 based.

```
ListBox.RemoveAll()
MyArray = new Array()
RowCount = ActiveDocument.Sections["Results"].RowCount
//GetCell Returns the value of an individual cell in a Column
for (j = 1; j <= RowCount; j++)
   MyArray[j] = ActiveDocument.Sections["Results"].Columns[1].GetCell(j)
//Use JavaScripts built in Array sorting to sort the values
SortedArray = MyArray.sort()
// Add all the sorted items to the listbox control
for (j = 0; j< MyArray.length;j++)
```

```
ListBox.Add(SortedArray[j])
```

Methods:

Add(Value As String), Item(Index As Number) As String, OnClick(), OnClientClick (), OnDoubleClick(), Remove(Index As Number), RemoveAll(), Select(Index As Number), Unselect(Index As Number)

Properties:

Read-only: Property Count As Number, Property Name As String, Property Type As BqShapeType

Read-write: AccessibilityText (Property), Alignment (Property) As BqHorizontalAlignment,, AppendObjectText (Property), Property ClientScriptStatus as Boolean, Property Enabled As Boolean, Property MultiSelect As Boolean, Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Objects and Collections:

UserValues (Collection), Font As Font, SelectedList As SelectedList

ControlsRadioButton (Object)

Member of:

Controls collection, DashboardSection object

Description:

Represents a Dashboard radio button that enables users to select one value out of many. Radio buttons function like check boxes; however, when grouped, only one button can be selected.

Example:

This example shows how to determine which Radio button is selected from a group:

```
NumControls = ActiveDocument.Sections["Dashboard2"].Shapes.Count
for (I = 1; I <= NumControls;I++)
{
if (ActiveDocument.Sections["Dashboard2"].Shapes[I].Group == "ButtonGroup")
if(ActiveDocument.Sections["Dashboard2"].Controls[I].Checked==true)
          Alert("Radio Button"+ ActiveDocument.Sections["Dashboard2"].Controls[I].Name
+" Is checked")
}
```

Methods:

OnClick(), OnClientClick()

Properties:

Read-only: Property Group As String, Property Name As String, Property Type as BqShapeType

Read-write: AccessibilityText (Property), Alignment IProperty) As BqHorizontalAlignment, AppendObjectText (Property), Property Checked As Boolean, Property Enabled As Boolean, Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Objects:

UserValues collection, Fill As Fill, Font as Font

ControlsTextBox (Object)

Member of:

Controls collection, Dashboard Section object

Description:

Represents a Dashboard text box that must have unique names. Use these name to reference the object in scripts.

Example 1:

This example shows how to change and enable text box text. This script assumes it is run from the Dashboard section and uses the *TextBox* control:

```
TextBox.Text = "Hello World"
if (TextBox.Enabled == false)
          TextBox.Enabled = true
```

Example 2:

This example shows how to populate a Dashboard text box with query limit line values.

```
MyLimit = ActiveDocument.Sections["Query"].Limits[1].SelectedValues[1]
ActiveDocument.Sections["Dashboard"].Shapes["TextBox1"].Text = MyLimit
```

Methods:

OnChange(), OnClick(), OnClientClick(), OnClientEnter, OnClientExit, OnEnter(), OnExit()

Properties:

Read-only: Property Name As String, Property Type As BqShapeType

Read-write: AccessibilityText (Property), Alignment IProperty) As BqHorizontalAlignment, AppendObjectText (Property), Property ClientScriptStatus as Boolean, Property Enabled As Boolean, Property Password As Boolean, Property Scrollable As Boolean, Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Cookies (Object)

Member of:

Session object

Description:

Represents a list of key value pairs, stored as cookies. Cookies are units of read only text (less than 4K) that exist in the Interactive Reporting document file deployed for the Interactive Reporting Web Client or EPM Workspace session.

Note: Added key value pairs to the Cookies (Collection) do not write them back to the Web browser.

Example 1:

```
//Button 1 set action
var Username = ActiveDocument.ODSUsername
Session.Cookies.Add("User Cookie", Username)

//Button2 retrieve action
UserCookieValue = Session.Cookies["User Cookie"]
Alert("The username entered on the Server login is: "+UserCookieValue,"ODSUSername")
```

This example shows you how to retrieve the user name from the login:

Example 2:

This example shows how to test scripting in the client server version by creating temporary values in the:

```
//Add some test cookies
  Session.Cookies.Add("MyCookie","MyValue")
  Session.Cookies.Add("ApplicationName",Application.Name)
//Write out the values to the console window
  Console.Write (Session.Cookies["MyCookie"])
  Console.Write (Session.Cookies["ApplicationName"])
```

Methods:

Add(String Key, String Value), Item (Key as string)

Properties:

Read-only: Property Count As Number

CornerLabels (Object)

Member of:

PivotSection object, OLAPQuerySection object

Description:

Represents the Pivot report corner labels feature. Corner labels are the names of Pivot Outliner values in the actual pivot. You can include corner labels on pivot reports and specify their position (none, top, or side).

Example 1:

In this example, corner labels are displayed on the side of a pivot report:

```
LabelActiveDocument.Sections"Pivot"].CornerLabels.Display= BqPivotLabelDisplaySide
```

Example 2:

In this example, corner labels are displayed on the top of the pivot report:

```
ActiveDocument.Sections["Pivot"].CornerLabels.Display= BqPivotLabelDisplayTop
```

Example 3:

In this example, corner labels are displayed on both the top and side of the pivot report:

```
ActiveDocument.Sections["Pivot"].CornerLabels.Display= BqPivotLabelDisplayBoth
```

Example 4:

In this example, corner labels are hidden:

```
ActiveDocument.Sections["Pivot"].CornerLabels.Display= BqPivotLabelDisplayNone
```

Example 5:

This example shows how to use an if...else statement to retrieve and display corner label display setting in a text box:

```
var display = ActiveDocument.Sections["OLAPQuery"].CornerLabels.Display
if (display  == bqPivotLabelDisplayBoth)
{
TextBox9.Text = "Both";
}
else if (display  == bqPivotLabelDisplayNone)
{
TextBox9.Text = "None";
}
else if (display  == bqPivotLabelDisplaySide)
{
TextBox9.Text = "Side";}
else if (display  == bqPivotLabelDisplayTop){
TextBox9.Text = "Top";}
```

Properties:

Read-write: Property Display as BqPivotLabelDisplay

Constants:

BQPivotLabelDisplay, which includes these values:

- bqPivotLabelDisplayBoth
- bqPivotLabelDisplayNone
- bqPivotlabelDisplaySide
- bqPivotLabelDisplayTop

CreatedDate (Object)

Member of:

Document object

Description:

Represents document creation dates.

Action:

Read-only

Example:

This example shows how to display a document creation date.

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

CubeQuerySection (Object)

Member of:

Sections collection

Description:

Represents a CubeQuery section.

Example

This example shows to make the CubeQuery section active, process the query, download to results, and show the results as a chart:

```
ActiveDocument.Sections["Query"].Activate()
ActiveDocument.Sections["Query"].Process()
```

```
ActiveDocument.Sections["Query"].DownloadToResults()
ActiveDocument.Sections["Query"].ShowAsChart()
```

Methods:

Activate(), Copy(), DownloadToResults(), Duplicate(), Export([optional]Filename As String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders, [optional]Prompt as Boolean, [optional] Encoding as BqEncoding), ExportToStream ([optional] Filename As String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders As Boolean, [optional] DataStreaming as Boolean, [optional] Prompt As Boolean, [optional] Encoding as BqEncoding), PrintOut([optional] FromPage As Number, [optional] ToPage As Number, [optional] Copies As Number, [optional] Filename As String, [optional] Prompt as Boolean), Process(), Recalculate(), Remove(), ShowAsChart()

Properties:

Read only: Active as Boolean, Type as BqSectionType

Read-write:IncludeInProcessAll as boolean, LastPrinted, Name as string, ProcessSequenceNum as number, ShowFilters as boolean, ShowOutliner as boolean, Visible as boolean

Objects and Collections:

Catalog object as OLAPCatalogNew, Columns collection as QueryLabel, Filters collection as OLAPFilters, QueryOptions as EssbaseSpecific object, Rows collections as QueryLabel, UserValues collection as user values

CustomValues (Collection)

Member of:

LimitValues object

Description:

Represents a custom list associated with a limit. Users open limits to select values.

Retrieve a custom limit value using the CustomValue (Collection).

Note: The Add (Method) is unavailable for the AvailableValues (Collection) because the values are fetched from the database. For the "CustomValues (Collection)" on page 60, the Add (Method) adds a value to the list. For the "SelectedValues (Collection)" on page 198, the Add (Method) adds a value to the selected list. The AddAll (Method) of the SelectedValues (Collection) selects all values of AvailableValues or CustomValues (Collection).

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1)`

```
myItem = Documents["StartUp.bqy"] myItem =
Documents.Item("StartUp.bqy")
```

Example:

This example shows how to add all AvailableValues (Collection) values to the SelectedValues (Collection). This is the same as performing a select all values and transferring the selection in the Limit dialog:

```
LimitCount =
ActiveDocument.Sections["Results"].Limits[1].AvailableValues.Count
for (i=1;i<=LimitCount;I++)
{
MyVal =
Add(ActiveDocument.Sections["Results"].Limits[1].AvailableValues[i]
ActiveDocument.Sections["Results"].Limits[1].SelectedValues.Add(MyVal)
}
```

Methods:

Add(Value ValueItem), AddAll(), Item(Number Index), RemoveAll()

Properties:

Read-only: Property Count as Number

DashboardSection (Object)

Member of:

Sections collection

Description:

Represents a Dashboard section.

Note: All section objects are inherited from a base section object. For this reason some sections have methods and properties which do not necessarily apply to them. Methods and properties that do not apply, throw exceptions.

Example:

This example shows how to access a list of controls in a Dashboard section. It also shows how to rename, hide, or display a section:

```
MyDashboard = ActiveDocument.Sections["Dashboard"]
Console.Write("Number of Controls = "+MyDashboard.Controls.Count)
Console.Write("The First Control is Named: "+MyDashboard.Controls[1].Name)
MyDashboard.Name = "My Dashboard Section"
//If the section is hidden then show it
```

```
if (MyDashboard.Visible == false)
            MyDashboard.Visible = true
```

Methods:

Activate(), Copy(), Duplicate(), Export([optional] Filename As String, FileFormat As BqExportFileFormat, [optional], IncludeHeaders As Boolean, [optional] Prompt as Boolean), ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean, OnActivate(), OnDeactivate(), PrintOut([optional] FromPage As Number, [optional] ToPage as Number, [optional] Copies as Number, [optional] Filename as String, [optional] Prompt as Boolean), Recalculate(), Remove()

Properties:

Read-only: Property Active As Boolean, Property LastPrinted As Date, Property Type As BqSectionType

Read-write: Property Name As String, Property ShowOutliner as Boolean, Property ShowSortLine as Boolean, Property Visible As Boolean

Collections:

Controls as Controls, Shapes As Shapes

Data (Object)

Member of:

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

Description:

Represents items in the data layout (outliner), which includes actual and target fact value, and axis objects used in gauges, (Live) charts and sliders.

When referencing a Data (Object), you must specify the section name from which the column is drawn in the SourceSectionName (Property). Otherwise, it is not clear which section to use if there are multiple sections with the same column name

```
// Associate data with chart
// NOTICE: SourceSectionName must be specified first BarChart.SourceSectionName =
"Results"; BarChart.Facts.Add("Unit Sales"); BarChart.Categories.Add("Region");
```

Example:

This example shows how to remove the Unit Sales fact object, and add the Amount Sales fact object:

```
Speedometer.Facts.Remove("Unit Sales")
Speedometer..SourceSectionName = "Results”
Speedometer.Facts.Add("Amount Sales")
```

Methods:

None

Properties:

None

Objects and Collections:

Categories collection, Facts collection, TargetFact collection

DataLabels (Object)

Member of:

PivotSection object, OLAPQuery object

Description:

Represents the Pivot or OLAPQuery data labels feature. Data labels are the column and row heading on the top and sides of a pivot report and define the categories for numeric values. You can include data labels on pivot reports and specify their position (none, top, side, or both).

Example 1:

In this example, data labels are displayed on the side of the pivot report:

```
ActiveDocument.Sections["Pivot"].DataLabels.Display= BqPivotLabelDisplaySide
```

Example 2:

In this example, data labels are displayed on the top of a pivot report:

```
ActiveDocument.Sections["Pivot"].DataLabels.Display= BqPivotLabelDisplayTop
```

Example 3:

In this example, data labels are displayed on the top and side of a report:

```
ActiveDocument.Sections["Pivot"].DataLabels.Display= BqPivotLabelDisplayBoth
```

Example 4:

In this example, data labels are hidden:

```
ActiveDocument.Sections["Pivot"].DataLabels.Display= BqPivotLabelDisplayNone
```

Example 5:

In this example, data labels are hidden:

```
ActiveDocument.Sections["Pivot"].DataLabels.Display= BqPivotLabelDisplayNone
```

Properties:

Read-write: Property Display as BqPivotLabelDisplay

Constants:

BQPivotLabelDisplay, which includes these values:

- bqPivotLabelDisplayBoth
- bqPivotLabelDisplayNone
- bqPivotlabelDisplaySide
- bqPivotLabelDisplayTop

DataModelSection (Object)

Member of:

QuerySection object

Description:

Represents the underlying Data Model for a Query Section or DataModelSection object. The Data Model (Object) contains connection information, Table Catalog, etc. you access from the Data Model or Query section.

Example 1:

This example shows how to define basic properties of a Data Model. It disables AutoJoin and AutoAlias, limits queries to 20 minutes, and enables joins between iconized topics. Use the *with* statement to call object methods and properties without qualifying them:

```
with (ActiveDocument.Sections["Query"].DataModel)
{
   AutoAlias = false
   AutoJoin = false
   TimeLimit = 20
   ShowIconJoins = true
}
```

Example 2:

This example shows how to build a Data Model using the Table Catalog object. It assumes a database connection.

```
with (ActiveDocument.Sections["Query"].DataModel)
{
Topics.RemoveAll()
AutoJoin = false
//Create two new topics from tables in Table Catalog
Catalog.Refresh()
```

```
Table1 =Catalog.CatalogItems["WINE"]
Table2 =Catalog.CatalogItems["WINE_SALES"]
Topics.Add(Table1)
Topics.Add(Table2)
Field1 = Topics[1].TopicItems["Wine Id"]
Field2 = Topics[2].TopicItems["Wine Id"]
//Create a new join by joining two TopicItems together
Joins.Add(Field1,Field2,bqJoinSimpleEqual)
// Now add topic items to the request line
for (I = 1; I <= Topics[1].TopicItems.Count; I++)
ActiveDocument.Sections["Query"].Requests.Add(Topics[1].Name,Topics[1].
TopicItems[I].DisplayName)
}
```

Methods:

AuditSQL(EventType as BqAuditEventType, SQLString as String) SyncWithDatabase()

Properties:

Read-only: TimeLimitActive as Boolean

Read-write: Property AutoAlias As Boolean, Property AutoJoin As Boolean, Property RowLimit as Number, RowLimitActive as Boolean, Property ShowIconJoins As Boolean, Property TimeLimit As Number, Property TimeLimitActive

Objects:

Catalog As DMCatalog, Connection As Connection, MetaDataConnection As Connection, JoinOptions as JoinOptions

Collections:

Joins As Joins, Limits As Limits, Topics As Topics, Local Results as LocalResults, LocalJoins as LocalJoins

Date Field (Object)

Member of:

Fields collection

Description:

Sets the current date in MM/DD/YY format.

Example:

This example shows how to add 3–inch wide line border to the Date Field:

```
ActiveDocument.Sections["Sales Report"].ReportHeader.Fields["Date Field"].Line.Color
=10040166
ActiveDocument.Sections["Sales Report"].ReportHeader.Fields["Date Field"].Line.Width =4
```

Methods:

Layer(Value As BqLayer), Spring(Name as String), UnSpring

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType (Number)

Objects/Coll.:

UserValues, LineFormat, FillFormat, FontFormat

DateNow Field (Object)

Member of:

Fields collection

Description:

Sets the current date MM/DD/YY format.

This object represents the date when the Date Now field is added to a report.

Example:

This example shows how to concatenate the string *Created on:* and the date on which the DateNow Field was added to the report:

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

Methods:

Layer(Value as BqLayer), Spring(Name as String), UnSpring()

Properties:

Read-write: Property Formula as String, Property HorizontalAlignment as BqHorizontalAlignment, Property Text as String, Property TextWrap as Boolean, Property VerticalAlignment as BqVerticalAlignment

Read-only: Property Name as String, Property Type as BqShapeType

Objects/Coll.:

UserValues, LineFormat object, FillFormat, Font

DateTime Field (Object)

Member of:

Fields collection

Description:

Sets the current date in MM/DD/YY HH:MM: AM format.

The DateTimeNow (Object) represents the date and time when it is first added. It does not change.

Example:

This example shows how to change the size of the data in DateTime Field to twelve point:

```
ActiveDocument.Sections["Sales Report"].ReportHeader.Fields["DateTime Field"].Font.Size
= 12
```

Methods:

Layer(BqLayer value), Spring(String Name), UnSpring

Properties:

Read-write: Property Formula as String, Property HorizontalAlignment as BqHorizontalAlignment, Property Text as String, Property TextWrap as Boolean, Property VerticalAlignment as BqVerticalAlignment

Read-only: Property Name as String, Property Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

DateTimeNow Field (Object)

Member of:

Fields collection

Description:

Sets the current date and time in MM/DD/YY HH:MM: AM format.

The DateTimeNow Field (Object) represents the date and time when this field is added to reports. It does not change.

Example:

This example shows how to red color fill the DateTimeNow Field in the report header band:

```
ActiveDocument.Sections["SalesReport"].ReportHeader.Fields["DateTimeNow
Field"].Fill.Color = bqRed
```

Methods:

Layer(BqLayer value), Spring(String Name), UnSpring

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

DBSpecific (Object)

Member of:

OLAPQuerySection object

Description:

Represents database-specific OLAPQuery options. The properties and method in the DBSpecific object vary based on the connection type (Essbase or MS/OLE DB for OLAP).

Example:

This example shows how to have three decimal points returned from the server:

```
ActiveDocument.Sections["OLAPQuery2"].DBSpecific.DecimalPlaces = 3
```

Methods:

Essbase only: AliasTable(AliasTableName As String, [optional] PromptOption As Number, [optional] PromptDialog As Boolean)

Properties:

Essbase: Property Decimal Places as Number, Property EnableForHybridAnalysis as Boolean, Property SuppressEmptyRows as Boolean, Property SuppressMissingRows as Boolean, Property SuppressZeroRows as Boolean

DefaultFormats (Object)

Member of:

Application object

Description:

Represents the collection of GridFormats objects. These objects set the default formats for data values, fonts and labels in the Pivot and OLAPQuery (both CubeQuery and OLAPQuery) sections.

Example:

This example shows how to set the default format for data values in a Pivot section to red, sets the border style to 3–D raised, and sets the font size to 12:

```
DefaultFormats.Pivot.DataValues.BorderColor = bqDefaultFormatsRed
DefaultFormats.Pivot.DataValues.BorderStyle = bqBorderStyle_Raise1
DefaultFormats.Pivot.DataValues.Font.Size = 12
```

Objects:

GridFormats object

DefinedJoinPaths (Collection)

Member of:

DataModel object

Description:

Defined join paths are customized join preferences that enable you to include or exclude tables based on items referenced on Request and Limit lines. This limits the query to tables based on available groupings; generating the most efficient SQL for Data Model queries. The features in this collection correspond to Define Join Paths dialog box options.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example shows how to select a user defined join path option and delete the join path:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionDefJoin
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath
["MyJoinPath"].Remove()
```

Example 2:

This example shows how to select the user defined join path option, and change a defined join path by adding a join path topic:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionDefJoin
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath
["MyJoinPath"].AddTopic("Periods")
```

Example 3:

This example shows how to select the user-defined join path option, create a defined join path, and add all join path topics to the defined join path:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionDefJoin
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath.Add
("MyJoinPath")
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath
["MyJoinPath"].AddAllTopics()
```

Methods:

Add(DefinedJoinPath as String), Item (NameOrIndex as Value), Remove(NameOrIndex As String), RemoveAll()

Properties:

Read-write: Count As Number

DefinedJoinPath (Object)

Member of:

DefineJoinPaths collection

Description:

Contains the customized join preferences that enables Interactive Reporting to include or exclude appropriate tables based on the items referenced on the Request and Limit lines.

Example 1:

This example shows how to select the user-defined join path option, and change a defined join path by adding a join path topic:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionDefJoin
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath
["MyJoinPath"].AddTopic("Periods")
```

Example 2:

This example shows how to select the user defined join path option, create a defined join path, and add all join path topics to the defined join path:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionDefJoin
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath.Add("MyJoinPath"
)
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.DefinedJoinPath["MyJoinPath"].Ad
dAllTopics()
```

Methods:

AddAllTopics(), AddTopic(DefinedJoinPathsName As String), Remove(), RemoveAllTopics(), RemoveTopic(DefinedJoinPathName As String)

Properties:

Read-write: Name As String

DerivableQuery (Object)

Member of:

DerivableQueries collection

Description:

Represents a derivable query set in a Table Catalog.

Example:

This example shows how to display deliverable query names in Alert boxes:

```
AlertActiveDocument.Sections["Query"].DataModel.Catalog.DerivableQueries["Query2"].Name
```

Properties:

Read Only: Property Name as String

DerivableQueries (Collection)

Member of:

DMCatalog object

Description:

Represents all derivable queries in the Table Catalog.

Tip: All collections have a "Item(NameOrIndex)" method. This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to count Derivable Queries in the Table Catalog and display the data in an Alert box:

```
Alert(ActiveDocument.Sections["Query"].DataModel.Catalog.DerivableQueries.Count)
```

Methods:

Item(NameOrIndex) As DMResult

Properties:

Read-only: Property Count As Number

DerivedItem (Object)

Member of:

DerivedTable object

Description:

Represents a (topic) item in a derived table.

Example:

This example shows how to display a derived item name in an Alert box:

```
Alert(ActiveDocument.Sections["Query2"].DataModel.DerivedTables["Query"].DerivedItems["S
tore Id"].DisplayName)
```

Properties:

Read-only: Property DisplayName as String, Name as String

DerivedItems (Collection)

Member of:

DerivedTable object

Description:

Represents all derived items in the derived table.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and it returns an item in the collection at a particular index or with a specific name. Use, brackets ([]) to represent calls to the Item (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to display the count of derived item in an Alert box:

```
Alert(ActiveDocument.Sections["Query2"].DataModel.DerivedTables["Query"].DerivedItems.Co
unt, "Number of Derived Items")
```

Methods:

Item(value NameOrIndex)

Properties:

Read-only: Property Count As Number

DerivedTable (Object)

Member of:

DerivableQueries collection

Description:

Represents a derived table in the Table Catalog.

Example:

This example shows how to remove a derived table from a query:

```
ActiveDocument.Sections["Query2"].DataModel.DerivedTables["Query"].Remove()
```

Collections:

DerivedItems collection

Methods:

Remove()

Properties:

Read Only: Property Name

DerivedTables (Collection)

Member of:

Represents derived tables in a DMCatalog object

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to count derivable queries in the Table Catalog and display the number in an Alert box:

```
Alert(ActiveDocument.Sections["Query"].DataModel.Catalog.DerivableQueries.Count)
```

Methods:

Add(DerivableQuery DerivableQueryObject), Item(Value NameOrIndex), RemoveAll()

Properties:

Read-only: Property Count As Number

DesktopClient (Object)

Member of:

Events collections

Description:

Enables or disables execution of an Interactive Reporting Studio document event in the 8.0 client environment. The object properties take a Boolean value, which is true by default.

Example:

This example shows how to disable all document events in the Interactive Reporting Studio environment:

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

```
Documents["Sample1.bqy"].Events["DesktopClient"].ExecuteOnPreProcess=false
Documents["Sample1.bqy"].Events["DesktopClient"].ExecuteOnShutDown=false
Documents["Sample1.bqy"].Events["DesktopClient"].ExecuteOnStartUp=false
```

Properties:

Read-write: ExecuteOnPostProcess as Boolean, ExecuteOnPreProcess as Boolean, ExecuteOnShutDown as Boolean, ExecuteOnStartUp as Boolean

Read-only: Name as String

Dimension (Object)

Member of:

Dimension collection

Description:

Represents a table dimension in the Report section. A dimension is typically a qualifiable text value, such as a region, and includes date values. It defines the secondary headings or labels that comprise a report. Dimensions ares repeated in each group. Usually, you use items containing text values (for example, Year or item type) for table dimensions. For example, if you select item type as the table dimension, item type is a dimension within each group header. Under the dimension Item Type, the name of each item type (such as CD ROM) appears. A fact is an quantifiable value, such as amount of sales, budget or revenue.

Example 1:

This example shows how to put the *City* dimension before the *State Province* :

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions["City"].Move("State
Province")
```

Example 2:

This example shows how to suppress duplicate values in report table columns:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions["City"].SuppressDuplic
ates = true
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions["State
Province"].SuppressDuplicates = true
```

Example 3:

This example shows how to make the background color of the *City* dimension light blue and bold the font:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions["City"].BackgroundColo
r = bqLightBlue
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions["City"].Font.Style =
bqFontStyleBold
```

Methods:

Move(LabelNameBefore as String), Remove()

Properties:

BackgroundAlternateColor as BqColorType, BackgroundAlternateFrequency as Number, BackgroundColor as BqColorType, BackgroundShowAlternateColor as Boolean, HorizontalAlignment as BqHorizontalAlignment, Name as String, NumberFormat as String, SuppressDuplicates as Boolean, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Objects:

Font object

Dimensions (Collection)

Member of:

ReportTable collection

Description:

Represents all table dimension objects in the report section. A dimension is typically a qualifiable value, such as a region or product line. A fact is a quantifiable value, such as budget or revenue.

Note: When you use the Add (Method), Move (Method), and Remove (Method) with this collection, and the SuspendCalculation (Property) is true (which is the default), use the Recalculate (Method) to force the Report section to recalculate itself.

Example:

This example shows how to add the *City* label as a dimension:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Dimensions.Add
("City", "Results")Recalculate()
```

Methods:

OLAPDimension Item(value NameOrIndex), RetrieveDimension

Properties:

Read-only: Count as Number

DMCatalog (Object)

Member of:

DataModelSection object

Description:

Represents the Table Catalog. This object lists the database tables used when a Data Model is built.

Example 1:

This example shows how to create a Data Model by inserting tables from the Table Catalog. It also shows how to change the basic display properties of the Table Catalog:

```
with (ActiveDocument.Sections["Query"].DataModel)
{
            Catalog.ShowFullName= true
//Updates the Table Catalog with the most current view of the tables
            Catalog.Refresh()
            Table1 =Catalog.CatalogItems["WINE"]
            Table2 =Catalog.CatalogItems["WINE_SALES"]
//Create two new topics from tables in Table Catalog
            Topics.Add(Table1)
            Topics.Add(Table2)
}
```

Example 2:

This example shows how to list available database tables:

```
// display Table Catalog
ActiveDocument.Sections["DataModel"].DataModel.Catalog.Refresh()
```

Example 3:

This example shows how to add topics to a Data Model section:

```
CatItem =
ActiveDocument.Sections["DataModel"].DataModel.Catalog.CatalogItems["PCW_ITEMS"]
ActiveDocument.Sections["DataModel"].DataModel.Topics.Add(CatItem)
```

Example 4:

This example prints the names of databases for each table in the Table Catalog:

```
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)
```

Methods:

Refresh()

Properties:

Read-write: Property ShowDerivableQueries as Boolean, Property ShowFullNames As Boolean, Property ShowLocalResults As Boolean

Collections:

CatalogItems As DMCatalogItems, Results As Results

DMCatalogItem (Object)

Member of:

DMCatalogItems collection

Description:

Represents a table in the Table Catalog.

Example:

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

```
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.Write(OutputString+"\r\n")
         }
}
```

Methods:

None

Properties:

Read-only: DatabaseName As String, Property Name As String, Property Owner As String

DMCatalogItems (Collection)

Member of:

DMCatalog object

Description:

Represents a list of all items in the Table Catalog.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

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.Write(OutputString+"\r\n")
      }
}
```

Methods:

Item(NameOrIndex) As DMCatalogItem

Properties:

Read-only: Property Count As Number

Document (Object)

Member of:

Documents collection, Application object

Description:

Contains the content of the file (document) created byInteractive Reporting that you store on your computer. Each Interactive Reporting document contains multiple sections.

Note: The Document.Close() and Documents.Open() syntax are not supported in EPM Workspace.

Example 1:

This example shows how to references a document object that enumerates the document object or reference to the ActiveDocument object. *myDoc* refers to the same document object:

```
myDoc = Documents[1] or
myDoc = Documents["Testdoc.bqy"] or
if "Testdoc.bqy" is the current document then
myDoc = ActiveDocument
```

Example 2:

In this example, the Section Title bar is disabled in the ActiveDocument, and the document saved with a new name:

```
ActiveDocument.ShowSectionTitlebar = false
ActiveDocument.SaveAs("d:\\Hyperion Docs\\Updated File.bqy")
ActiveDocument.Close()
```

Methods:

Activate(), AddExportSection(SectionName As String), Close([optional] SaveChanges As Boolean]), Export([optional]Filename As String, [optional]FileFormat As BqExportFileFormat, [optional] Prompt As Boolean]), InterruptQueryProcess(), OnPostProcess(), OnPreProcess(), OnShutdown(), OnStartup(), ProcessAll(), RemoveExportSection(), Save([optional] bCompressed As Boolean), SaveAs([optional] Filename As String, bCompressed As Boolean, [optional] Prompt As Boolean],

Properties:

Read-only: Property Accessibility as Boolean, Property Active As Boolean, Property Name As String, Property Path As String, Property ProcessEventOrigin as BqProcessEventOriginType, Property QueryInProcess as String

Read-write: Property ShowCatalog As Boolean, Property ShowSectionTitleBar As Boolean

Collections:

Sections As Sections, Events as Events, UserValues as UserValues

Object:

LastSaved As LastSaved

Documents (Collection)

Member of:

Application object

Description:

Represents all document objects in applications.

Note: The Document.Close() Documents.Open(), and Documents.Name (Property) are not supported in EPM Workspace.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")\
```

Example:

This example shows how to print all open document names to the Console window. It compares these names with the ActiveDocument (the document which has Focus) and appends its name with “Active”:

```
For (I= 1;I <= Documents.Count;I++)
            {
            if (Documents[I].Name == ActiveDocument.Name)
                        Console.Writeln(ActiveDocument.Name + "- Active")
            else
                        Console.Writeln (Documents[I].Name)
            }
```

Methods:

Add([optional] Name As String), Item(Value as Name or Index) New([optional] Name As String), Open(Filename As String, DisplayName as String)

Properties:

Read-only: Property Count As Number

EmbeddedBrowser (Object)

Member of:

Controls collection, Dashboard object

Description:

Represents an embedded browser control in the Dashboard section. This embedded control is an instance of a web browser window, arbitrarily positioned on the Dashboard. The EmbeddedBrowser is like an embedded section object, except content can be referenced externally by URL (rendered in a web browser), rather than a section.

Example:

This example shows how to reference URLs using drop-down controls:

```
if (ActiveDocument.Sections["Dashboard"].Shapes["DropDown1"].SelectedIndex==1)
ActiveDocument.Sections["Dashboard"].Shapes["EmbeddedBrowser1"].URL="www.ACME.com";
if (ActiveDocument.Sections["Dashboard"].Shapes["DropDown1"].SelectedIndex==2)
ActiveDocument.Sections["Dashboard"].Shapes["EmbeddedBrowser1"].URL="www.BestComputerPro
ducts.com";
if (ActiveDocument.Sections["Dashboard"].Shapes["DropDown1"].SelectedIndex==3)
ActiveDocument.Sections["Dashboard"].Shapes["EmbeddedBrowser1"].URL="www.Comp.com";
```

Methods:

ModifyRepositoryFileAnalyzer, ModifyRepositoryFileBQY, ModifyRepositoryFileBQYJob, ModifyRepositoryFileOther, ModifyRepositoryFileReports, ModifyRepositoryFileSQRJob

Properties:

Read-write: AccessibilityText (Property) Alignment as BqHorizontalAlignment, AppendObjectText (Property), ClientScriptStatus as Boolean, Enabled as Boolean, Text as String, URL as String, VerticalAlignment as BqVerticalAlignment, Visible as Boolean

Read-only: Name as String, Repository as Boolean, RepositoryBQYSection as String, Repository BQYToolbarType as BQRepositoryBQYToolbarType, RepositoryDocument as String, RepositoryFiletype as BqRepositoryFiletype, RepositoryJobRun as Boolean, RepositorySmartcut as String, RepositorySmartcutParams, RepositoryToolbartType as BqRepositoryToolbarType, ShowScrollbar as BqScrollbarType, Text as String, Type as BqShapeType

Objects:

UserValues collection, LineFormat object, FillFormat object, Font object

EndQuery (Object)

Member of:

Parentheses object

Description:

Represents the ending query of a Parentheses object expression. The Parentheses object expression nests parentheses () for the evaluation order of an appended query. The expression must include two queries or more. The default evaluation order is left to right.

Example:

This example shows how to remove the ending query of a Parentheses object expression..

```
ActiveDocument.Sections["Query"].AppendQueries["Parentheses"]
["Query,Query2"].EndQuery.Remove()
```

Methods:

Remove()

Properties:

Read-write: Name as String, UnionController as BqUnionController

Collections:

Requests collection, Limits collection

Events (Collection)

Member of:

Documents collection

Description:

Represents these event collections objects within the document collection:

- DesktopClient
- PlugInClient
- ThinClient
- Scheduler
- SmartView

Use these objects to enable or disable document events in a 8.0 client environment.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and returns an item in the collection at a particular index or with a specific name. Use brackets "[]" to represent calls to theItem (Method). For example, these statements cause identical behavior: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how shows how to count and display the sections in an Interactive Reporting document (.bqy) that contain document events in a Console window:

```
for(I=1 ; I <= Documents["Sample1.bqy"].Events.Count; I ++)
         Console.Write (ActiveDocument.Sections[I].Name+"\r\n")
```

Methods:

Item(Value As NameOrIndex)

Properties:

Read-write: Count As Number

Objects:

DesktopClient, PlugInClient, ThinClient, Scheduler, SmartViewClient

EventScript (Object)

Member of:

EventScripts collection

Description:

Provides read and write access to all event handling scripts provided for all shapes, documents and Dashboard sections with event handlers.

The EventScript object has three properties: Name, Script, and Type, and no methods. The Name property is a read-only string containing the event name for example, “OnClick”.

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.

The Name property allows the object tree to display the EventScript objects with their event names and provides named access by way of the Item method.

The Type property allows for better performance when checking event types if this is required.

Constants:

The Type property of the EventsScripts (Collection) is a read-only value of the enumeration BqScriptableEvents, which consists of the following values:

- bqScriptableEventsOnActivate
- bqbqScriptableEventsOnCellDoubleClick
- bqScriptableEventsOnChange
- bqScriptableEventsOnClick
- bqScriptableEventsOnClientClick
- bqScriptableEventsOnClientEnter
- bqScriptableEventsOnClientExit
- bqScriptableEventsOnClientSelection
- bqScriptableEventsOnDeactivate
- bqScriptableEventsOnDoubleClick
- bqScriptableEventsOnEnter
- bqScriptableEventsOnExit

- bqScriptableEventsOnPostProcess
- bqScriptableEventsOnPreProcess
- bqScriptableEventsOnRowDoubleClick
- bqScriptableEventsOnSelection
- bqScriptableEventsOnShutdown
- bqScriptableEventsOnStartup

Example:

This example shows how to create a command button named “cbcreated” on an OnClick event. It 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")
```

Properties:

Read only: Name as String. Type as BqScriptableEvents

Read-write: Script as String

Methods:

None

EventScripts (Collection)

Member of:

Document object, Shape object

Description:

Represents a collection event handling scripts has been provided for all shapes, documents, and Dashboard section with event handlers.

The EventScripts collection has a read-only Count property (the collection contains 1.Count EventScript objects) and an Item method which can return a reference to an EventScript object given an event name or 1-based index number.

Example:

This example shows how to write the number of event handling scripts for commandbutton 1:

```
Console.Writeln(CommandButton1.EventScripts.Count)
```

Methods:

Item(Value as Name of Index)

Properties:

Read only: Count as Number

Objects:

EventScript object

FactAxis (Object)

Member of:

Subcomponent object

Description:

Represents attributes of the Facts axis of a (Live) Chart object, including the associated Fact axis labels and titles. Setting properties for this object is the equivalent of setting properties for the Fact label and title on the Font dialog box.

Example:

This example shows how to set common fact axis attributes:

```
BarChart.FactsAxis.NumberFormat = "$#,##0.00"
BarChart.FactsAxis.AutoScale = false
BarChart.FactsAxis.ScaleMax = 1000
BarChart.FactsAxis.ScaleMin= 10
BarChart.FactsAxis.Labels.Font.Color = 16711680
BarChart.FactsAxis.Title.Text = "Amount Sales"
```

Methods:

None

Properties:

AutoScale as Boolean, NumberFormat as String, ScaleMax as Number, ScaleMin as Number

Objects:

Labels object, Title object

Facts (Collection)

Member of:

CategoryItems collection, Data object

Description:

For charts, this collection represents all Chart fact objects within the "Body (Object)" on page 36on page 36.

For gauges, this collection represents all Bullet, Slider, Speedometer, Thermometer and TrafficLight actual fact objects. Additionally, it does not include the AddComputed method, or the Axis Type (Property). Data functions can be applied to fact values to aggregate values.

Example:

In this example, all items in the Chart Outliner are removed, and specific items added:

```
ActiveDocument.Sections["Chart"].Facts.RemoveAll()
ActiveDocument.Sections["Chart"].Facts.Add("Amount_Sales")
ActiveDocument.Sections["Chart"].Facts.Add("Unit_Sales")
```

Methods:

Add(ItemName As String), AddComputedItem(Name As String, Expression As String, [optional] Index As Number), Item (Value as NameOrIndex) Remove(Value as NameOrIndex), RemoveAll()

Properties:

Read-only: Property Axis Type as BqChartAxisType, Property Count As Number

Facts (Object)

Member of:

Facts collection

Description:

For a chart object, this object represents a chart Y axis. The Facts object properties affect Y axis position and the Y Facts categories in the Outliner.

For a gauge, this object represents this objects represents an individual Bullet, Slider, Speedometer, Thermometer and TrafficLight actual fact object. It does not have any methods, and only includes a read-only Name property.

Example:

In this example, the fact object uses grand total data percent:

```
ActiveDocument.Sections["Chart"].Facts["Amount Sales"].DataFunction =
bqDataFunctionPercentOfGrand
```

Methods:

Hide(), UnHide()

Properties:

Read-Write: Property Axis AxisPlotValue as BqChartAxisPlotValue, Property DataFunction as BqDataFunction, Property Name as String

Read-Only: Name as String

Field (Object)

Member of:

Fields collection

Description:

The Field (Object) sets a computable field.

Example:

This example shows how to display fields with text messages (text labels):

```
try
{
ActiveDocument.Sections["Report"].PageHeader.Fields["Field"].Formula = "'This is a text
label'"
ActiveDocument.Sections["Report"].Recalculate()
}
catch(e)
{
Console.Writeln(e.toString())
}
```

Methods:

Layer(Value As BqLayer value), Spring (Name As String), UnSpring ()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

Fields (Collection)

Member of:

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

Description:

Represents all field objects in the report section.

Example:

This example shows how to count the number of Fields inserted in the Body band of the report:

```
Alert(ActiveDocument.Sections["Report"].Body.Fields.Count + " Number of fields in this
band")
```

Methods:

Item(Value As NameOrIndex)

Properties:

Read-only: Count as Number

Objects:

ReportName Field, Path Field, FileName Field, Date TimeNow Field, TimeNowField, DateNow Field, Time Field, Last Printed Field, Date Field, LastSaved Field, Page XofY Field, PageCount Field, PageNm Field, Query SQL field, Result Limit, Query Limit, Field

FillFormat (Object)

Member of:

Body object, Control object, LegendItem object, PageHeader object, PageFooter object, ReportHeader object, ReportFooter object, Shape object

Description:

The FillFormat (Object) contains all properties associated with object background formatting.

Constants

- Fill.Color=BqColorType
- Fill.Pattern=BqFillPatternType

Example:

This example shows how to set the fill pattern of a Report Group Header to fifty (50)

```
Documents["Sample 1.bqy"].Sections["Report"].Groups["Report
Group1"].Header.Fill.Pattern = bqFillPattern50
```

Methods:

None

Properties:

Read-write: Property Color As BqColorType (Number), Property Pattern as BqFillPattern

Font (Object)

Member of:

Control object, Title object, ValueLabels object

Description:

Contains all font methods and properties.

Example:

This example shows how to change the size, color, and alignment of a text label:

```
MyLabel = ActiveDocument.Sections["Dashboard"].Controls["TextLabel"]
MyLabel.Font.Size = 10
MyLabel.Font.Color = bqBlue
MyLabel.Font.Align = bqAlignTop
```

Constants:

The Align property uses the BqVerticalAlignment constant group, which consists of these values:

- bqAlignBottom
- bqAlignMiddle
- bqAlignTop

The Color 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 Effect property uses the BqFontEffect constant group, which has these values:

- bqFontEffectNone
- bqFontEffectSubScript
- bqFontEffectSuperScript
- bqFontEffectStrikeThrough
- bqFontEffectStrikeThrough
- bqFontEffectUnderline

The Justify property uses the BqHorizontalJustification constant group, that has these values:

- bqAlignCenter
- bqAlignLeft

- bqAlignRight

Example:

This example shows how to make a text label blue and italic.

```
ActiveDocument.Sections["Dashboard"].Shapes["TextLabel1"].Font.Color=bqBlue
ActiveDocument.Sections["Dashboard"].Shapes["TextLabel1"].Font.Style=bqFontStyleItalic
```

Methods:

None

Properties:

Read-write: Property Color As BqColorType, Property Effect As BqFontEffect (not supported in the EPM Workspace), Property Name As String, Property Size As Number, Property Style As BqFontStyle

Footer (Object)

Member of:

ReportGroup object

Description:

Represents the attributes of the report group footer band.

In the user interface, when you drag an item from the Catalog pane to the Report Group Outliner, Interactive Reporting automatically supplies a report group header band and adds a label inside the band, which identifies the group. A group header categorizes data into repeating collections of records in a header band.

Example:

This example shows how to change the color page number field footer to red:

```
ActiveDocument.Sections["Report"].Groups["Report Group2"].Footer.Fields["PageNm
Field"].Font.Color = 16711680
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects:

LineFormat object, FillFormat object, Tables collection, Fields collection, Shapes collection, Shapes collection, Pivots collection, Pivot collection, Chart collection

Form (Object)

Member of:

Session object

Description:

Represents a list of key value pairs stored and generated from a POST method of an HTML form. Form elements are the controls that enable users to make selections on HTML pages. When a browser collects data from HTML forms, it directs it to a Hyper Text Transfer Protocol (HTTP) server indicated in the HTML Form, and starts the Data Access Servlet (DAServlet). The DAServlet collects the form values and packages them for the Interactive Reporting document (.bqy). The Form (Object) provides read-only access to the form elements values. Because HTML forms are browsers-based this collection applies only to the Interactive Reporting document (. bqy). However, the Form (Object) is exposed in client server products for plug-in scripts.

If you develop HTML Forms for Interactive Reporting, note the following:

- Method—Specifies how data passes to the server when forms are submitted. Send forms using GET or POST methods. GET sends from results in the URL submitted to the script. POST encodes data sent to the script. The script in the following example uses the POST method.
- Action—Defines how forms are processed and when they are submitted to the Data Access Servlet (part of Hyperion Foundation Suite 8). Actions contain the location of BQY documents.
- JScript=enable—Key-value pair that enables browsers to read and access JavaScript.
- (input)—

  Form fields ares referred to as an input items (the HTML tag is <input>). Input item are depicted in form output by a name and value. The name indicates the script handling the for uses. The value is the content. These form fields are available:

  - Hidden text—Invisible to users. Hold unique data passed between server and client that may include information to be accessed later, such as date values.
  - Text Field—Single or multiple lines. Text fields are handy for short and long answers to questions. Default size is one row by 20 rows.
  - Check box—Enables users to select one answer.
  - Radio buttons—Consist of a group of two or more buttons. Users can select only one button in a group. Other selected button are cleared.
  - Select values—Creates a drop-down that enables users to select from a list of items.

Note: The Session.Form.Add () and Session.Form.Item() object model syntax is *not* supported in EPM Workspace.

Note: If a form launches the Interactive Reporting Web Client with the "withnav" variant of the Smartcut, the Session.Form collection items is inaccessible. In this instance, use withnav_get or withnav_run. This behavior can be set as the default mode for a Smartcut by editing the ws.conf line to show: WebClient.UserInterface.Navigation.ShowNavigationBar=true

Tip: All collections have the "Item(NameOrIndex) method." This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example 1:

This example shows how to read values of a Form elements and use them in a plug-in script:

Basic HTML Form:

```
<HTML>
<BODY>
<!"Note: The Action Key have a value which opens a document from the web client. You
MUST include the "Jscript=enable" key-value pair to initialize the plug-in scripting ->
<FORM METHOD = "post" ACTION = "http://your.server.com/ods-cgi/odscgi.exe?
Method=getDocument&Docname=-1835-83481598112-58541278350-125-8-1-1387-9434&JScript=enabl
e">
<P>Text Box <INPUT id=text1 name=text1></P>
   <P>Password <INPUT id=password1 name=password1 type=password></P>
<P>Text Area <TEXTAREA id=TEXTAREA1 name=TEXTAREA1></TEXTAREA></P>
<P>Check Box<INPUT id=checkbox1 name=checkbox1 type=checkbox></P>
<P>Radio <INPUT id=radio1 name=radio1 value = "1st" type=radio><INPUT
id=radio1 name=radio1 type=radio value = "2nd" CHECKED></P>
<P>DropDown<SELECT id=select1 name=select1 >
        <OPTION value=Value1>Display1
        <OPTION value=Value4>Display4</SELECT></P>
<P>ListBox <SELECT id=select2 name=select2 size=4 multiple>
        <OPTION value=Value1>List1
        <OPTION value=Value4>List4</SELECT></P>
<P><INPUT id=submit1 name=submit1 type=submit value=Submit></P>
</FORM>
</BODY>
</HTML>
//Script running on plug-in
//Write all values to console window
Console.Writeln("Text1 Value = "+ Session.Form["text1"])
Console.Writeln("password1 Value = "+ Session.Form["password1"])
Console.Writeln("TEXTAREA1 Value = "+ Session.Form["TEXTAREA1"])
Console.Writeln("checkbox1 Value = "+ Session.Form["checkbox1"])
Console.Writeln("radio1 Value = "+ Session.Form["radio1"])
Console.Writeln("select1 Value = "+ Session.Form["select1"])
```

```
Console.Writeln("select2 Value = "+ Session.Form["select2"])
```

Example 2:

The following example shows how to pass store code entered in a form text field to Interactive Reporting documents. Values are passed and written to the Console window, passed as limit value, and then the queries processed.

The first part of this example shows an HTML file collects the form value field:

The StoreSales.htm and bqy show how to pass a form value and apply it as a limit:

```
<HTML>
<BODY>
<FORM METHOD = "post" ACTION = "http://elearn.hyperion.com/Hyperion/dataaccess/Browse?
REQUEST_TYPE=GET_DOCUMENT&DOC_UUID=000000f5f703e795-0000-0404-40a02b5e&DOC_VERSION=1&JSc
ript=enable">
<p align="center"><font face="Verdana, Arial, Helvetica, sans-serif"><b><font
size="5">Store Sales by Product Name - 2001</font></b></font></p>
<p>Please enter your store code:
   <input type="text" name="storecode"></P>
<P><INPUT id=submit1 name=submit1 type=submit value=Submit></P>
</FORM>
</BODY>
</HTML>
```

The second part of the example shows how a document script was scripted to collect and write the values on an OnStartup event:

```
//Write all values to console window
var storecode = Session.Form["storecode"];
Console.Writeln(storecode);
var StoreCodeLimit = ActiveDocument.Sections["Query"].Limits["Store Code"];
StoreCodeLimit.SelectedValues.RemoveAll();
StoreCodeLimit.SelectedValues.Add(storecode);
ActiveDocument.Sections["Query"].Process();
```

Example 3:

This example shows how to read values of Form elements within an HTML file and pass them to the Interactive Reporting Web Client. Passed values are displayed in the Console Window.

The URL cited reflects the URL name of Interactive Reporting Web Client which was published using Hyperion Foundation Suite 8.0. To determine the URL of Interactive Reporting Web Client, perform these tasks:

- Log on to Hyperion Foundation Suite 8.
- Access the selected Interactive Reporting document.
- Position the cursor on the Interactive Reporting Web Client icon for the document.
- Right-click and select properties.
- Copy and paste the URL into the HTML file.

The first part of the example shows how the HTML file was coded to collect form value fields. From the *text* input type, the state location value is passed. From the *radio* input type, the college-

level value is passed. From the *check box* input type, the book title is passed. From the *select name* input type field, the State/Province value is passed:

```
<HTML>
<BODY>
<!"Note: The Action Key have a value which opens a document from the web client. You
MUST include the "Jscript=enable" key-value pair to initialize the plug-in scripting ->
<FORM METHOD = "post" ACTION = "http://elearn.hyperion.com/Hyperion/dataaccess/Browse?
REQUEST_TYPE=GET_DOCUMENT&DOC_UUID=000000f56c19b5c5-0000-0404-40a02b5e&DOC_VERSION=1&JSc
ript=enable>
<p align="center"><font face="Verdana, Arial, Helvetica, sans-serif"><b><font
size="5">Book
    Purchase Survery</font></b></font></p>Metro Bookseller Store Location
    <input type="text" name="storelocation"></P>
  </p>
    <input type="text" name="state"></P>
  </p>
  <p>Choose your college year:
    <input type="radio" name="radiobutton1" value="freshman">
    freshman
    <input type="radio" name="radiobutton2" value="sophmore">
    sophmore
    <input type="radio" name="radiobutton3" value="junior">
    junior
    <input type="radio" name="radiobutton4" value="senior">
    senior</p>
  <p>Choose your favorite books (select as many as apply):</p>
  <p>
    <input type="checkbox" name="checkbox" value="The DataWarehouse Toolki">
    The DataWarehouse Toolkit</p>
  <p>
    <input type="checkbox" name="checkbox2" value="A Short History of Byzantium">
    A Short History of Byzantium</p>
  <p>
    <input type="checkbox" name="checkbox3" value="A Course in Game Theory">
    A Course in Game Theory</p>
  <p>
    <input type="checkbox" name="checkbox4" value="Brilliant Deductions">
    Brilliant Deductions</p>
  <p>Select your home state:
    <select name="state_province">
      <option>California</option>
      <option>New York</option>
      <option>Texas</option>
    </select>
    </p>
<P><INPUT id=submit1 name=submit1 type=submit value=Submit></P>
</FORM>
</BODY>
</HTML>
```

In the second part of the example, the Document OnStartup script writes the values passed from the HTML form; the variable names correspond to the control (input) names set in the HTML document:

```
//Write all values to console window
Console.Writeln("Text Value = "+ Session.Form["storelocation"])
```

```
Console.Writeln("radiobutton Value1 = "+ Session.Form["radiobutton1"])
Console.Writeln("radiobutton Value2 = "+ Session.Form["radiobutton2"])
Console.Writeln("radiobutton Value3 = "+ Session.Form["radiobutton3"])
Console.Writeln("radiobutton Value4 = "+ Session.Form["radiobutton4"])
Console.Writeln("checkbox Value = "+ Session.Form["checkbox"])
Console.Writeln("checkbox2 Value = "+ Session.Form["checkbox2"])
Console.Writeln("checkbox3 Value = "+ Session.Form["checkbox3"])
Console.Writeln("checkbox4 Value = "+ Session.Form["checkbox4"])
Console.Writeln("select Value = "+ Session.Form["state_province"])
```

Methods:

Add(Key As String, Value As String), Item(NameorIndex) As Form

Formatting (Object)

Member of:

Toolbars collection

Description:

Represents the Formatting toolbar in the application. The Formatting toolbar is used to provide text formatting, styling and editing commands.

If the Formatting (Object) has any of its associated properties accessed fromEPM Workspace, the script commands are ignored, no exceptions are thrown, and any scripts continue.

Constants:

The Type property of the Toolbar object uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows you enable the Formatting toolbar in the application:

```
//Syntax for turning on the Formatting toolbar
Toolbars["Formatting"].Visible=true;
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

(Live) FunnelChart (Object)

Member of:

Dashboard objects

Description:

Represents a (Live) FunnelChart object and its properties.

The following (Live) object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Orientation—BqObjectOrientation
- Type—BqShapeType

Example 1:

This example shows how to set common Funnel Chart attributes:

```
FunnelChart.Comments = "Sales Amounts"
FunnelChart.Effect = bqLiveChartEffect3D
FunnelChart.Locked = false
FunnelChart.Name = "Sales Cycles"
FunnelChart.SourceSectionName = "Results"
FunnelChart.Visible = true
```

Example 2:

This example shows how to remove all bar width facts, and add “Unit Sales” as the bar width fact item:

```
FunnelChart.WidthFacts.RemoveAll()
FunnelChart.WidthFacts.Add("Unit Sales")
```

Methods:

None

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property), :Comments as String, Effect as BqLiveChartEffect, Locked as Boolean, Name as String, Orientation as BqObjectOrientation, SourceSectionName as String, Visible as Boolean

Read only:: Type as BqShapeType

Objects and Collections:

UserValues collection, Placement object, Data object, Subcomponents object

GridFormats (Object)

Member of:

DefaultFormats object

Description:

Represents a collection of the individual CellFormat objects for the Pivot and OLAPQuery (both CubeQuery and OLAPQuery) sections. Setting the properties for this individual CellFormat objects corresponds to setting the same properties in the Default Formats dialog box. Default formats are applied to the new sections of the current Interactive Reporting document (BQY) and new documents.

Example:

This example shows how to set the default format for data values in a Pivot section to red, sets the border style to 3–D raised, and sets the font size to 12:

```
DefaultFormats.Pivot.DataValues.BorderColor = bqDefaultFormatsRed
DefaultFormats.Pivot.DataValues.BorderStyle = bqBorderStyle_Raise1
DefaultFormats.Pivot.DataValues.Font.Size = 12
```

Objects:

CellFormat object

Group (Object)

Member of:

ReportSection object

Description:

Categorizes data into repeating collections of records in a header band.

Example:

This example shows how to remove all items in Report Group 1:

```
ActiveDocument.Sections["Report"].Groups["Report Group1"].Remove()
```

Methods:

Move(LabelNameBefore String), Remove()

Properties:

Read-only: Name as String

Objects:

Header object, Footer object, GroupItems collection, SortItems collection

Groups (Collection)

Member of:

ReportSection object

Description:

Represents the Report Groups; that is, the user selects the Groups portion of the Outliner in the Reporter. It is treated like a header band, but there are separate objects for the actual Report page headers/footers and Report header/footer.

Note: If you use the Add (Method), Move (Method), and Remove (Method) with this collection, and the SuspendCalculation (Property) is true (default), use the Recalculate (Method) to recalculate the Report section.

Example:

This example shows how to add the *Year* column member:

```
ActiveDocument.Sections["Report"].Groups.Add(Year)
Recalculate()
```

Methods:

Add(Member as String, [optional] Section Dependency as String), Item (Value As NameOrIndex), RemoveAll()

Properties:

Read-only: Count as Number

GroupItem (Object)

Member of:

GroupItems collection

Description:

Represents a column that was dragged to the Report Group Outliner, such as the *Store Id* column from the Results section.

Example:

This example shows you how to write the name of the *Amount Sales* group item to the Console window:

```
Console.Writeln(ActiveDocument.Sections["Report"].Groups["Report
Group1"].GroupItems["Amount Sales"].Name)
```

Methods:

Move(LabeLNameBefore as String), Remove()

Properties:

Read-only: Name As String

GroupItems (Collection)

Member of:

ReportGroup object

Description:

Represents items for a specific report group.

Example 1:

This example shows how to remove all group items in the report group band in Report Group 1:

```
ActiveDocument.Sections["Report"].Groups["Report Group1"].GroupItems.RemoveAll()
```

Example 2:

This example shows how to add the *Year* and *Results* group items to Report Group 1:

```
ActiveDocument.Sections["Report"].Groups["Report Group1"]. GroupItems.Add("Year",
"Results")
```

Methods:

Add(String Member, [optional] StringSectionDependency), Item(NameOrIndex as Value). RemoveAll()

Properties:

Read-only: Count as Number

Header (Object)

Member of:

ReportGroup object

Description:

The Header (Object) represents the attributes of the report group header band. When you drag an item from the Catalog pane to the Report Group data layout, Interactive Reporting automatically supplies a group header band and adds a label inside the band, which identifies the group. A group header categorizes data into repeating collections of records in a header band.

For example, if you create a report to show purchases by state, each state serves as a group header for the report. Other items can be added as sub-categories, such as buyers.

Example:

This example shows how to count and display the number of tables in the Report Group 1 header:

```
ActiveDocument.Sections["Report"].Groups["Report Group1"].Header.Tables.Count
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects and Coll.:

LineFormat, FillFormat, Tables, Fields, Shapes, Shapes collection, Pivots collection, Pivot collection, Chart collection

HyperLink (Object)

Member of:

Controls collection, DashboardSection object

Description:

Represents an hyperlink control in the Dashboard section. The content may be displayed in a pop-up window or the current window. This functionally is equivalent to a text graphic with the Object Model command OpenURL associated with its OnClick event.

In Avalanche, this mapping occurs for a Hyperlink (Object):

Hyperlink “Current Window” is mapped 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 the "/get" mapping

- For Interactive Reporting documents (.bqy), the "legacy layout" (i.e. how it looked in 8.3.2, but colors and icons are updated) is displayed.
- For Web Analysis and Financial Reporting—The Masthead, View pane, and Menu are hidden.
- For other doc types—Everything but content area is hidden.
- "/withnav_get"— Full framework is displayed (just as if the user logged into Avalanche normally)

Example:

This example shows how to make HyperLink2 visible in the Dashboards section, and make the content replaces the top HTML window:

```
HyperLink2.Visible=true
HyperLink2.DisplayMode=bqOpenURLTargetTop
```

Methods:

ModifyRepositoryFileAnalyzer, ModifyRepositoryFileBQY, ModifyRepositoryFileBQYJob, ModifyRepositoryFileOther, ModifyRepositoryRepositoryFileReports, ModifyRepositoryFileSQRJob

Properties:

Read-write: AccessibilityText (Property), Alignment as BqHorizontalAlignment,, AppendObjectText (Property), DisplayMode as BqOpenURLTarget, Enabled as Boolean, Text as String, URL as String, VerticalAlignment as BqVerticalAlignment, Visible as Boolean

Read-only: Name as String, Repository as Boolean, RepositoryBQYSection as String, Repository BQYToolbarType as BQRepositoryBQYToolbarType, RepositoryDocument as String, RepositoryFiletype as BqRepositoryFiletype, RepositoryJobRun as Boolean, RepositoryReportsDisplayFormat as BqExportFileFormat, RepositorySmartcut as String, RepositorySmartcutParams, RepositoryToolbarType as BqRepositoryToolbarType,

Objects:

UserValues collection, LineFormat object, FillFormat object, Font object

Image (Object)

Member:

Images collection

Description:

Represents an individual image object available in the Resource Manager (an user interface utility used to load, manage and share picture in the Interactive Reporting document file.

- BMP—Bitmap (default picture object)
- GIF—Graphic Interchange Format
- JPG, JPEG—JPEG File Interchange Format
- PNG—Portable Network Graphics Format

The Height (Property) for this object 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.

For an Image object, this 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.

The Name (Property) for this object is equivalent to reading the name in the Resource Manager user interface. The property is read-write, and returns a string representing the name of the image. The Default value of the property is the name of the image. The property persists with the document/application.

The UID (Property) for this object 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; there is no default value.

Example:

This example shows how to show the height 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 Height"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{

TextBox1.Text ="Step 1"

try
{
TextBox3.Text =  "Height is: " + ActiveSection.Shapes["Picture"].RMImage.Height
}
catch(e)
{
```

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

}

else
{
TextBox1.Text ="Step 2"

try
{
ActiveSection.Shapes["Picture"].RMImage.Height = TextBox5.Text
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
{
TextBox3.Text =  "Height is: " + ActiveSection.Shapes["Picture"].RMImage.Height
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

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

Properties:

Read only: Height as Number, UID as String, Width as Number

Read write: Name as String

Images (Collection)

Member of:

ResourceManager object

Description:

Represents a collection of all image objects available in the Resource Manager (an user interface utility used to load, manage and share pictures in the Interactive Reporting document file).

The Images collection supports the following picture graphics objects:

- BMP—Bitmap (default picture object)

- GIF—Graphic Interchange Format
- JPG, JPEG—JPEG File Interchange Format
- PNG—Portable Network Graphics Format

The Add (Method) for this collection takes two argument. The first argument is a string. This is equivalent to specifying the path to the image. The value provided must be string representing the windows path, mapped windows network drive, UNC path, or UNIX path. If no value is specified, the method does not operate correctly and generates an exception. The second argument is also a string. This is equivalent to selecting a given image from the resource manager in the user interface. The value provided must be string representing the case-sensitive name for an image that appears for an image in the resource manager. The default value for the second parameter is the file name of the picture. If the display name is not specified, Interactive Reporting assigns a file name is to the image.

The FindItems (Method) is equivalent to determining a group of images with the same name in the Resource Manager user interface. The return value is a pseudo-array containing all images in the Resource Manager with the display name equal to selected. The first item in the array is at ordinal 1.

The Item (Method) takes one argument. The argument is a number or string. This is equivalent to selecting the n object in a list, or an object byname. The value provided must be string representing an integer between 1 and the number of like objects present, or a string matching an item name. The default value is 1.

Example 1:

This example shows how to add the image “splash.jpg” to the Resource Manager:

```
ActiveDocument.ResourceManager.Images.Add("c:\\splashy.jpg")
```

Example 2:

This example shows how to find a group of images with the same name in the Resource Manager. The return value is captured by the Count property and displayed in a text box.

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

var ActionName = "RM FindItems()"

TextBox1.Text ="Start " + ActionName

strDisplayName = TextBox5.Text

if (TextBox5.Text == "")
{

try
{

arrFindRM= ActiveDocument.ResourceManager.Images.FindItems("Waves")
```

```
//arrFindRM= ActiveDocument.ResourceManager.Images.FindItems("")

TextBox3.Text = "Count for this image is: " + arrFindRM.Count

ActiveSection.Shapes["ListBox2"].RemoveAll()

for (i = 1; i <= arrFindRM.Count; i++)
{
Console.Writeln(arrFindRM[i].Name)
ActiveSection.Shapes["ListBox2"].Add(arrFindRM[i].Name)
} //end for

} //end try
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}//end catch

} //end if
else
{

try
{
arrFindRM= ActiveDocument.ResourceManager.Images.FindItems(TextBox5.Text)

TextBox3.Text = "Count for this image is: " + arrFindRM.Count

ActiveSection.Shapes["ListBox2"].RemoveAll()

for (i = 1; i <= arrFindRM.Count; i++)
{
Console.Writeln(arrFindRM[i].Name)
ActiveSection.Shapes["ListBox2"].Add(arrFindRM[i].Name)
} //end for

} //end try
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}//end catch

}

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

Methods:

Add (String Path, [optional] string displayName). FindItems(String DisplayName), Item (Value IndexOrUID), Remove(ResourceManagerImage image), RemoveAll()

Properties:

Read only: Count as Number

Objects:

Item (Object)

Member of:

Items collection

Description:

Represents a specific legend object.

This object attributes directly affect the display of an object in the legend of a Chart section.

Example:

This example shows how to add a diagonal brush style and the color red to the *Books* legend item:

```
ActiveDocument.Sections["Chart"].Legend.Items["Books"].Fill.BrushStyle =
bqBrushStyleDiagCross
ActiveDocument.Sections["Chart"].Legend.Items["Books"].Fill.Color =  16711680
```

Properties:

Read-only: Name as String

Objects:

FillFormat, LineFormat

Items (Collection)

Member of:

Legend object

Description:

Represents all legend items.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to count legend items in an Alert box.

```
Alert(ActiveDocument.Sections["Chart"].Legend.Items.Count)
```

Methods:

Item(Value NameOrIndex)

Properties:

Read-only: Count as Number

Join (Object)

Member of:

DataModel object

Description:

Represents a join between topics in a Data Model.

Example:

This example shows how to change the type of join to a left join and print the names of the joined topic items to the Console window:

```
ActiveDocument.Sections["Query"].DataModel.Joins[1].Type = bqJoinLeft
Console.Writeln(ActiveDocument.Sections["Query"].DataModel.Joins[1].TopicItem1.DisplayNa
me="Sales")
Console.Writeln(ActiveDocument.Sections["Query"].DataModel.Joins[1].TopicItem2.PhysicalN
ame)
```

Console Output:

WineId

WineId

Methods:

Remove()

Properties:

Read Only: Topic1Name, Topic2Name

Read-write: Type as BqJoinType

Objects:

TopicItem1, TopicItem2

JoinPath (Object)

Member of:

JoinOptions object

Description:

Represents a topic referenced in the join path. This feature is used when you use Defined Join Paths to customize join preferences on the Joins tab of Data Model options.

Example:

This example shows how to add all topics to the join path:

```
Documents["Sample
1.bqy"].Sections["SalesQuery"].DataModel.JoinOptions.DefinedJoinPaths["JoinPath1"].AddAl
lTopics()
```

Methods:

AddAllTopics, AddTopic(DefinedJoinPathsName as String), Remove(), RemoveAllTopics(), RemoveTopic(DefinedJoinPathName as String)

Properties:

Read Only: Name as String

Joins (Collection)

Member of:

DataModel object

Description:

Represents joins between topics in a Data Model.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and it returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example shows how to remove all the joins in the current Data Model and create a simple join between the Wine.Wine_Id and Wine_Sales.Wine_Id topic items:

```
with(ActiveDocument.Sections["Query"].DataModel)
{
Joins.RemoveAll()
Field1 = Topics["WINE"].TopicItems["Wine Id"]
Field2 = Topics["WINE_SALES"].TopicItems["Wine Id"]
//Create a new join by joining two TopicItems together
Joins.Add(Field1,Field2,bqJoinSimpleEqual)
}
```

Example 2:

This example shows how to add a join between two tables:

```
// add join between PCW_PERIODS (Day) and PCW_SALES (Order_Date)
PCWPeriods_Day = PCWPeriods.TopicItems["Day"]
PCWSales_OrderDate = PCWSales.TopicItems["Order_Date"]
Day_OrderDate_Join =
ActiveDocument.Sections["DataModel"].DataModel.Joins.Add(PCWPeriods_Day,PCWSales_OrderDa
te,bqJoinSimpleEqual)
```

Methods:

Function Add(TopicItem1 As TopicItem, TopicItem2 As TopicItem, Type As BqJoinType) As Join, Function Item(NameOrIndex) As Join, RemoveAll()

Properties:

Read-only: Property Count As Number

JoinOptions (Collection)

Member of:

DataModel object

Description:

Represents the available join usage preferences.

Constants:

The Type property of the JoinsOptions (Collection) uses the BqDataModelJoinsOptions constant group, which consists of these values:·

- bqDataModelJoinsOptionAllTopics (= 0, Read-only)
- bqDataModelJoinsOptionAutoJoin (= 4, Read-only)
- bqDataModelJoinsOptionDefJoin (=3, Read-only)
- bqDataModelJoinsOptionMinTopics (=1, Read-only)
- bqDataModelJoinsOptionRefTopics (=2, Read-only)

Note: If you choose to define your own joins paths by selecting the bqDataModel JoinsOptionDefJoin constant, specify your join preferences using the "DefinedJoinPaths (Collection)" on page 69.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to specify to use only topics represented by items on the Request line for joins:

```
ActiveDocument.Sections["Query"].DataModel.JoinsOptions.Type=
bqDataModelJoinsOptionMinTopics
```

Methods:

None

Properties:

Read-write: Property Type As BqDataModelJoinsOptions

Collections:

DefinedJoinPaths As DefinedJoinedPaths

Labels (Object)

Member of:

CategoryAxis object

Description:

Represents a (Live) chart label for the X and Y axes on a (Live) chart.

Example:

This example shows how to enable the category labels and color them red:

```
BarChart.CategoryAxis.Labels.Visible = true
BarChart.CategoryAxis.Labels.Font.Color = 16711680
```

Methods:

None

Properties:

Visible as Boolean

Objects:

None

LabelsAxis (Object)

Member of:

ChartSection object

Description:

Acts as a logical container for both labels axes in a chart.

Example:

This example shows how to set basic properties of the XAxis label and the ZAxis label:

```
with(ActiveDocument.Sections["Chart"])
{
      LabelsAxis.XAxis.ShowValues = true
      LabelsAxis.XAxis.ShowTickmarks = true
      LabelsAxis.ZAxis.ShowValues = false
       LabelsAxis.ZAxis.ShowTickmarks = false
}
```

Methods:

None

Properties:

XAxis: Property AutoFrequency as Boolean, Property LabelFrequency as Number, Property LabelText as String, Property MaximumBarsDisplayed as Number, Property MaximumBarsEnable as Boolean, Property ShowLabel as Boolean, Property ShowTickmarks as Boolean, Property ShowValues as Boolean, Property TickmarkFrequency as Number

ZAxis: Property LabelText as String, Property MaximumBarsDisplayed as Number, Property MaximumBarsEnable as Boolean, Property ShowLabel as Boolean, Property ShowTickmarks as Boolean, Property ShowValues as Boolean

Objects:

XAxis As XaxisLabel, ZAxis As ZAxisLabel

LabelValues (Object)

Member of:

ChartSection object, XLabels object, YLabels object, and Zlabels object

Description:

Represents the values on the YLabel, XLabel, or ZLabel.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections, and the method returns collection items using a particular index or name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to set up LabelValues items 1 and 2 in a new array, which can be used with the FocusSelection and HideSelection methods:

```
var Xarray = new Array();
Xarray[0] = ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(1)
Xarray[1] = ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(2)
```

Methods:

Item (Index As Number) As AxisLabelValueItem

Properties:

None

LastPrinted Field (Object)

Member of:

Fields collection

Description:

Sets the date on which the report section was last printed, in MM/DD/YY format. If the section has never been printed a value of zero is returned. If the section has been printed then the property shows up as an "object" in the tree rather than a property.

Example:

This example shows how to reposition the LastPrinted Field (Object) behind another object (such as a shape object):

```
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["LastPrinted
Field"].Layer(bqLayerBack)
```

Methods:

Layer(Value As BqLayer), Spring(Name As String), UnSpring()

Properties:

Read-write: Property Formula as String, Property HorizontalAlignment as BqHorizontalAlignment (Number), Property Text as String, Property TextWrap as Boolean, Property VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Property Name as String, Property Type as BqShapeType (Number)

Objects/Coll.:

User Values, LineFormat, FillFormat, Font

LastSaved (Object)

Applies To:

Document object, PluginDocument

Description:

Returns a value corresponding to the date on which a document was last saved. To get the date value, you must use the methods and properties of the Date object.

Action:

Read-only: Date object

Example:

This example shows how to print the date the document was last saved to the Console window:

```
Console.Writeln(ActiveDocument.LastSaved.toString())
Thu Jun 03 13:56:13 GMT-0700 (Pacific Daylight Time) 2001
```

LastSaved Field (Object)

Member of:

Fields collection

Description:

Sets the date on which the document was last printed, in MM/DD/YY format.

Example:

This example shows how to change the font color to red in the Last Saved field:

```
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["LastSaved Field"].Font.Color
= 16711680
```

Methods:

Layer(Value As BqLayer), Spring(Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignmentRead-only: Name as String, Type as BqShapeType

Objects:

UserValues, LineFormat, FillFormat, FontFormat object

LeftAxis (Object)

Member of:

ValuesAxis object

Description:

Represents all the left values axis properties in a chart.

Example:

This example shows how to set some basic properties of the left axis:

```
with(ActiveDocument.Sections["Chart"].ValuesAxis)
{
            LeftAxis.AutoScale = true
            LeftAxis.AutoInterval = true
            LeftAxis.ShowLabel = false
}
```

Methods:

None

Properties:

Read-write: Property AutoInterval As Boolean, Property AutoScale As Boolean, Property IntervalFrequency As Number, Property LabelText As String, Property ScaleMax As Number, Property ScaleMin As Number, Property ShowLabel As Boolean

Legend (Object)

Member of:

ChartSection object, Subcomponents object

Description:

For a ChartSection object, the Legend (Property) represents all of the methods and properties applicable to a chart legend.

For a gauge or (Live) chart, the Legend (Property) represents the legend associated with a selected gauge or (Live) chart.

Example 1:

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

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

Example 2:

This example shows how to show the trafficlight object legend.

```
TrafficLight.Legend.Visible = true
```

Constants:

The Legend (Object) uses the BqChartAxisType constant group, which consists of these values:

- bqChartXAxis
- bqChartYAxis
- bqChartZAxis

Methods:

None

Properties:

Property Focus as BqChartAxisType (ChartSection object only), Visible as Boolean (gauge and (Live) chart objects only)

Objects and Collections:

Items as Legend Items (ChartSection object only)

Font (gauge and (Live) chart objects only)

Legend (Collection)

Member of:

ChartSection object

Description:

Enables you to set and get legend item attributes of a chart. You use this collection to set and retrieve the line width of a line chart; or to modify the foreground color of a Bar chart.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example 1:

This example shows how to change the color, fill pattern, line color, and the line width of a legend item:

```
ActiveDocument.Sections["Chart"].Legend.Items[1].Fill.Color = bqBlue
ActiveDocument.Sections["Chart"].Legend.Items[1].Fill.BrushStyle = bqBrushStyleCross
ActiveDocument.Sections["Chart"].Legend.Items[1].Line.Color= bqBlue
ActiveDocument.Sections["Chart"].Legend.Items["Q1"].Line.Width = 6
```

Example 2:

The Legend style property sets the appearance of the line, such as a solid, dotted or a dashed line. this example shows how to set a "dot, dash" style:

```
ActiveDocument.Sections["Chart"].Legend.Items["1"].Line.Style= bqDashStyleDotDash
```

Example 3:

This example shows how to set the marker style, marker size, marker border color, and marker fill color of a legend item:

```
ActiveDocument.Sections["Chart"].Legend.Items["Q1"].Line.MarkerStyle = bqSquare
ActiveDocument.Sections["Chart"].Legend.Items[1].Line.MarkerSize = bq6pt
ActiveDocument.Sections["Chart"].Legend.Items[1].Line.MarkerBorderColor= bqRed
ActiveDocument.Sections["Chart"].Legend.Items["Q1"].Line.MarkerFillColor= bqGreen
```

Methods:

LegendItems.Item(NameOrIndex)

Properties:

Read-only: Property Count as Number

Limits (Collection)

Member of:

QuerySection object, DataModel object, TableSection object

Description:

Represents limits within a Results, Query, or Data Model section. It is analogous with the Limit line in Interactive Reporting. The identical syntax and collection used to manipulate regular query limit items are used for computed items. The one difference is the argument used in the Create.Limit(limit_item) method. The limit_item argument type is a string for both regular query items and computed items. The limit_item is a reference to the limit item on the limit line.

The reference to a regular query limit item is Topic.TopicItem. For example, given a *Products* table and a *Product_Line* column, the reference is CreateLimit(*Products* table and a *Product_Line*).

The reference to a computed item limit is Request.DisplayName. For example, given a "DoubleSales"computed item on the request line, the reference is CreateLimit("Request.Double Sales").

Tip: All collections have a "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example shows how to remove all existing limits and create a new query limit from an existing topic:

```
ActiveDocument.Sections["SalesQuery"].Limits.RemoveAll()
MyLimit = ActiveDocument.Sections["SalesQuery"].Limits.CreateLimit
("Sales_Fact.Unit_Sales")
MyLimit.Operator = bqLimitOperatorGreaterThan
MyLimit.CustomValues.Add(50)
MyLimit.SelectedValues.Add(50)
//Adds the limit to the Limit Line -
ActiveDocument.Sections["SalesQuery"].Limits.Add(MyLimit)
```

Example 2:

This example shows how to remove all existing limits and create a new query computed item limit:

```
ActiveDocument.Sections["SalesQuery"].Limits.RemoveAll()
MyLimit = ActiveDocument.Sections["SalesQuery"].Limits.CreateLimit
("Requests.Sales_Per_Unit")
```

```
MyLimit.Operator = bqLimitOperatorLessThan
MyLimit.CustomValues.Add(20)
MyLimit.SelectedValues.Add(20)
//Adds the limit to the Limit Line -
ActiveDocument.Sections["SalesQuery"].Limits.Add(MyLimit)
```

Example 3:

This example shows how to create a query computed item limit using the name of the computed item:

```
//Example creates a query computed item limit--use name of computed
itemmylimit=ActiveDocument.Sections["Query"].Limits.CreateLimit("Request.Sales_Differenc
e")ActiveDocument.Sections["Query"].Limits.Add(mylimit)
```

Example 4:

This example adds a Custom Value to an existing computed item limit:

```
//Example adds a CustomValue to an existing computed item
limitActiveDocument.Sections["Query"].Limits[2].CustomValues.Add('2')Or
ActiveDocument.Sections["Query"].Limits["Sales Difference"].CustomValues.Add('2')
```

Example 5:

This example adds a CustomSQL to an existing computed item limit:

```
//Example Adds CustomSQL to an existing computed item
limitActiveDocument.Sections["Query"].Limits["Sales
Difference"].LimitValueType=bqLimitValueTypeSQLActiveDocument.Sections["Query"].Limits["
Sales
Difference"].CustomSQL="SQLString"orActiveDocument.Sections["Query"].Limits[2].LimitValu
eType=bqLimitValueTypeSQLActiveDocument.Sections["Query"].Limits[2].CustomSQL="SQLString
"
```

Example 6:

This example shows how to create and set variable limits:

```
// create and set variable limit - Store_Id
mylimit = ActiveDocument.Sections["Query"].Limits.CreateLimit("PCW_SALES.Store_Id")
mylimit.Operator = bqLimitOperatorLessThanOrEqual
mylimit.CustomValues.Add(10)
mylimit.SelectedValues.Add(10)
ActiveDocument.Sections["Query"].Limits.Add(mylimit)
mylimit.VariableLimit = true
```

Methods:

Add(Limit As Limit), Function CreateLimit(limitItem As String), Function Item(NameOrIndex), RemoveAll()

Note: The argument for the CreateLimit method is different for regular limits, computed item limits, and aggregate limits. For regular limits the argument is a reference to the table topic and the topic item, for example, CreateLimit("Sales_Facts.Amount_Sales"). For

both computed item limits and aggregate limits the argument is a reference to the item's Display Name on the request line, for example, CreateLimit("Request.Amount Sales").

Properties:

Read-only: Property Count As Number

Collections:

Parentheses object

Limit (Object)

Member of:

Limit collection

Description:

Represents a limit. The Limit (Object) applies to Results, Data Model and Query Limits.

Note: If you intend to populate a list box with limit values from the database, add the RefreshAvailableValues (Method) before populating the list box. This enables the script to update the Results set when the query is opened.

Example 1:

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
```

Example 2:

This example shows how to create a new query limit from an existing topic:

```
//Create an empty Limit object from the "Wine.Cost" Topic Item.
MyLimit = ActiveDocument.Sections["Query"].Limits.CreateLimit("Wine.Cost")
MyLimit.Operator = bqLimitOperatorGreaterThan
MyLimit.SelectedValues.Add(10)
MyLimit.Name = "Costly Wine"
//Adds the limit to the Limit Line -
ActiveDocument.Sections["Query"].Limits.Add(MyLimit)
```

Example 3:

This example shows how to populate a list box control with the list of available values for an existing results limit. The same logic may be applied to Query and Data Model limits:

```
LimitCount = ActiveDocument.Sections["Results"].Limits[1].AvailableValues.Count
for (i=1;I<=LimitCount;i++)
ListBox.Add(ActiveDocument.Sections["Results"].Limits[1].AvailableValues[i])
```

Methods

LoadFromFile(Filename As String) As Boolean, RefreshAvailableValues(), Remove()

Properties:

Read-only: Property ValueSource As BqLimitValueSource

Read-write: Property CustomSQL As String, Property DataType as BqDataType, Property DisplayName As String, Property FullName As String, Property Ignore As Boolean, Property IncludeNulls As Boolean, Property LimitValueType as BqLimitValueType, Property LogicalOperator as BqLogicalOperator as BqLogical Operator, Property Name as String, Property Negate As Boolean, Property Operator As BqLimitOperator, Property Prompt As String, Property SuspendRecalculation as Boolean, Property VariableLimit As Boolean

Collections:

AvailableValues As LimitValues, CustomValues As LimitValues, SelectedValues As LimitValues

LimitValues (Collection)

Member of:

Limit object

Description:

Represents all the values associated with the different types of limits—regular, computed, and aggregate. Each limit object has three LimitValues collections: "AvailableValues (Collection)" on page 31, "SelectedValues (Collection)" on page 198, and "CustomValues (Collection)" on page 60. The AvailableValues (Collection) is used for regular limits only.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example 1:

This example shows how to add all values in AvailableValues (Collection) to the SelectedValues (Collection). This is the same as performing a select all values and transferring the selection in the Limit User Interface:

```
LimitCount =
ActiveDocument.Sections["Results"].Limits[1]. AvailableValues.Count
for (i=1;i<=LimitCount;i++)
{
MyVal =
Add(ActiveDocument.Sections["Results"].Limits[1]. AvailableValues[i]
ActiveDocument.Sections["Results"].Limits[1]. SelectedValues.Add(MyVal)
}
```

Example 2:

This example adds a CustomValue to the computed item limit:

```
ActiveDocument.Sections["Query"].Limits[2]. CustomValues.Add('2')
```

Example 3:

This example shows how to pass selected values from a pulldown list into a limit in the results set:

```
ActiveDocument.Sections["Results"].Limits[1].SelectedValues.RemoveAll();
NumSelected=Listbox.SelectedList.Count; for (var i=1; i<=NumSelected; i++)
{
ActiveDocument.Sections["Results"].Limits[1].CustomValues.Add(Listbox.SelectedList.Item(
i));
ActiveDocument.Sections["Results"].Limits[1].SelectedValues.Add(Listbox.SelectedList.Ite
m (i));
};
```

Methods:

Add(ValueItem), AddAll(), Item(Index As Number), RemoveAll()

Note: For the "AvailableValues (Collection)" on page 31, the Add (Method) does nothing, because the values are obtained from the database.

Properties:

Read-only: Property Count As Number

LineChart (Object)

Member of:

ChartSection object

Description:

Represents all the methods and properties specific to Line Charts.

Methods:

None

Properties:

Read-write: Property IgnoreNulls As Boolean

(Live) LineChart (Object)

Member of:

Dashboard object

Description:

Represents a (Live) LineChart object and its properties. (Live) LineChart objects can be shown in a line or area format

The following LineChart object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Subtype—BqLiveLineChartType
- Type—BqShapeType
- Unit—BqLiveChartTargetUnit

Example:

This example shows how to set common (Live) LineChart properties and methods:

```
LineChart.Comments = "Monthly Growth"
LineChart.Effect  =  bqLiveChartEffect2D
LineChart.Locked = false
LineChart.Name = "Market Share"
LineChart.SourceSectionName = "Results"
LineChart.Subtype = bqLiveLineChartTypeLine
LineChart.Visible = true
LineChart.TargetFacts.RemoveAll()
LineChart.TargetFacts.Add("Target")
```

Methods:

None

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property),: Comments as String, Effect as BqLiveChartEffect, Locked as Boolean, Name as String, Orientation as

BqObjectOrientation, SourceSectionName as String, Subtype as BqLiveLineChartType, Visible as Boolean

Read only:: Type as BqShapeType

Objects:

UserValues collection, Placement object, Data object, Subcomponents object, Target object

LineFormat (Object)

Member of:

Body object, Shape object, Control object, LegendItem object, TrendLine obect

Description:

Contains all of the properties associated with border formatting.

Constants:

The LineFormat (Object) uses these constants:

- Line.Color=BqColorType
- Line.DashStyle=BqDashStyle
- Line.MarkerBorderColor=BqColorType
- Line.MarkerFillColor=BqColorType
- LineMarkerSize=BqMarkerSize
- LineMarkerStyle=BqMarkerStyle
- LineStyle=BqDashStyle
- Line.Width=BqLineWidth

Example 1:

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

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

Example 2:

This example shows how to change the marker color and style for a line chart:

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

Methods:

None

Properties:

Read-write: Property Color As BqColorType (Number), Property DashStyle as BqDashStyle, Property MarkerBorderColor as BqColorType, Property MarkerFillColor as BqColorType, Property MarkerSize as Number, Property MarkerStyle as BqMarkerStyle, Property Width as Number

LocalJoins (Collection)

Member of:

DataModel object

Description:

Enables you to derive the Topic Name of a topic item in a join or local join. You can also retrieve the Topic Item Name for a joins (not a local join).

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example shows how to use a simple equal join:

```
Topic1=ActiveDocument.Sections["Query"].DataModel.Topics["Sales Fact"].
TopicItems.Item(2)
Topic2=ActiveDocument.Sections["Query"].DataModel.Topics["Products"]. TopicItems.Item(1)
ActiveDocument.Sections["Query"].DataModel.Joins.Add
(Topic1,Topic2,bqJoinSimpleEqual)
```

Example 2:

This example shows how to use a simple equal join to join topics 1 and 2 in a local results set:

```
LRTopic1=ActiveDocument.Sections["Query2"].DataModel.LocalResults["1"].
LocalResultTopicItems.Item(7)
LRTopic2=ActiveDocument.Sections["Query2"].DataModel.LocalResults["2"].
LocalResultTopicItems.Item(7)
ActiveDocument.Sections["Query2"].DataModel.LocalJoins.Add(LRTopic1,LRTopic2,
bqJoinSimpleEqual)
```

Methods:

Add([TopicItem1 As BaseTopicItem], [TopicItem2 As BaseTopicItem], [Type As BqJoinType] As LocalJoin), Item(NameOrIndex) As LocalJoin, RemoveAll()

Properties:

Read-only: Property Count As Number

LocalResults (Collection)

Member of:

DataModel object

Description:

Enables you to use local results in joins, and the Request line for processing results sets.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example adds a local results topic to a query section:

```
vDM=ActiveDocument.Sections["Query"].DataModel.Catalog.Results.Item(1)ActiveDocument.Sec
tions["Query"].DataModel.LocalResults.Add(vDM)
or
vDM= ActiveDocument.Sections["Query"].DataModel.Catalog.Results.Item("sales_fact")
ActiveDocument.Sections["Query"].DataModel.LocalResults.Add(vDM)
```

Example 2:

This example shows how to remove all local results topics and count the local results topics in a query section:

```
ActiveDocument.Sections["Query"].DataModel.LocalResults.RemoveAll()
ActiveDocument.Sections["Query"].DataModel.LocalResults.Count
```

Example 3:

This example removes a single local results topic and gets the topic item count of the "Results2" local topic:

```
ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].Remove()
ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].
LocalResultsTopicItems.Count
```

Example 4:

This example shows how to add a join between a topic and a local results topic:

```
Topic1=ActiveDocument.Sections["Query"].DataModel.LocalResults["Sales Fact"].
LocalResultTopicItems.Item("Store Id")
Topic2=ActiveDocument.Sections["Query"].DataModel.Results["Results2"].TopicItems.Item("S
tore Id")
ActiveDocument.Sections["Query"].DataModel.LocalJoins.Add(Topic1,Topic2, bqJoinLeft)
```

Example 5:

This example shows how to remove a single local results topic:

```
//Remove a single Local Results Topic
ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].Remove()
```

Example 6:

This example shows how to get a topic item count of local results:

```
//Get the Topic Item count of Local Results
TextBox1.Text=ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].LocalR
esultsTopicItems.Count
```

Example 7:

This example shows you to get a count of local joins in a query section:

```
//Count Local Joins
TextBox1.Text=ActiveDocument.Sections["Query"].DataModel.LocalJoins.Count
```

Example 8:

This example shows how to remove all local joins:

```
//Remove All Local Joins
ActiveDocument.Sections["Query"].DataModel.LocalJoins.RemoveAll()
```

Example 9:

This example shows how to add a join between a topic and a local results topic:

```
//Add Join between Topic and Local Results Topic
Topic1=ActiveDocument.Sections["Query"].DataModel.LocalResults["Sales
Fact"].LocalResultTopicItems.Item("Store Id")
Topic2=ActiveDocument.Sections["Query"].DataModel.Results["Results2"].TopicItems.Item("S
tore Id")
ActiveDocument.Sections["Query"].DataModel.LocalJoins.Add(Topic1, Topic2,bqJoinLeft)
```

Example 10:

This example shows how to add a topic from local results to the Request Line:

```
//Add Topic from Local Results to Request Line
ActiveDocument.Sections["Query"].Requests.Add("Results2","Quarter")
```

Methods:

Add(LocalResultObject As DMResult), Item(NameOrIndex) As LocalResult, RemoveAll()

Properties:

Read-only: Count as Number

LocalResultsTopicItems (Collection)

Member of:

LocalResults object

Description:

Enables you to use local results topic items in joins and in the Request line for processing results sets.

Example 1:

This example removes a single local results topic and gets the topic item count of the *Results2* local topic:

```
ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].Remove()
ActiveDocument.Sections["Query"].DataModel.LocalResults["Results2"].
LocalResultsTopicItems.Count
```

Example 2:

This example adds a join between a topic and a local results topic:

```
Topic1=ActiveDocument.Sections["Query"].DataModel.LocalResults["Sales Fact"].
LocalResultTopicItems.Item("Store Id")
Topic2=ActiveDocument.Sections["Query"].DataModel.Results["Results2"].TopicItems.Item("S
tore Id")
ActiveDocument.Sections["Query"].DataModel.LocalJoins.Add(Topic1,Topic2, bqJoinLeft)
```

Methods:

Item()

Properties:

Read-only: Count as Number

Marker (Object)

Member of:

Subcomponents object

Description:

Represents the market object and its properties for a (Live) LineChart object.

The Style (Property) of the Market (Object) returns the read only BqMarkerStyle constant group, which consists of the following values:

- bqMarkerStyleCircle— 3
- bqMarkerStyleDiamond—1
- bqMarkerStyleNone—0
- bqMarkerStyleRectangle—2
- bqMarkerStyleTriangle— 4

Example:

This example shows how to return the marker style for a (Live) Line Chart object and write it to the Console window::

```
Console.Writeln(LineChart.Marker.Style)
```

Method:

None

Properties:

Read only: Style as BqMarkerStyle

Read-write: Visible as Boolean

Objects:

None

MetaDataConnection (Object)

Member of:

Global object or Data Model object

Description:

Represents a Data Model metadata connection definition. Metadata connection definitions are powerful tools that you can use to link the Interactive Reporting application to metadata, or information about your database. By modifying the SQL Interactive Reporting sends to your database server, you can dictate where Interactive Reporting finds the information it uses to create a Data Model from database tables.

Metadata connection definitions read metadata from tables on the database and apply it to Data Models through a live database connection. The specifications for reading these tables are stored

in the connection file. After they are configured, metadata definitions are available for anyone using the connection file.

Example:

This example creates and applies a metadata connection file (.oce) to the current document. The data source is "PlutoSQLSVR," 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.MetaFileChoice = "Broadbase"
myCon.UseAlternateMetadataLocation(true,c:\\OCEs\\PlutoMeta.OCE)
myCon.EnableAsyncProcess = true
myCon.SaveAs("d:\\OCEs\\PlutoSQL.oce")
//Now use this connection in  a datamodel
ActiveDocument.Sections["Query"].DataModel.MetaDataConnection.Open()
("d:\\OCEs\\PlutoSQL.oce")
```

Method

Connect([optional] GetCredentials as Boolean), Disconnect(), Open(Filename As String), Save(), SaveAs(Filename As String), SetPassword(Password As String), UseAlternateMetadataLocation(Value As Boolean, [optional] MetadataOce As String)

Properties:

Read-only: Property Connected As Boolean, Property Filename As String

Read-write: Property AllowNonJoinedQueries As Boolean, Property Api As BqApi, Property AutoCommit As Boolean, Property Database As BqDatabase, Property DataBaseList As String, Property DBLibAllowChangeDatabase As Boolean, Property DBLibApiSeverity As Number, Property DBLibDatabaseCancel As BqDbLibCancelMode, Property DBLibPacketSize As Number, Property DBLibServerSeverity As Number, Property DBLibUseQuotedIdentifiers As Boolean, Property DBLibUseSQLTable As Boolean, Property EnableAsyncProcess As Boolean, Property EnableTransactionMode As Boolean, Property HostName As String, Property MetadataPassword As String, Property MetadataUser As String, Property MetaFileChoice As String, Property ODBCDatabasePrompt As Boolean, Property ODBCEnableLargeBufferMode As Boolean, Property SaveWithoutUsername As Boolean, Property ShowAdvanced As Boolean, Property ShowBrioRepositoryTables As Boolean, Property ShowMetadata As Boolean, Property SpecificMetadataLogin As Boolean, Property SQLNetRetainDateFormats As Boolean, Property StringRetrieval As Boolean, Property TimeLimit As Number, Property Username As String

ModifiedDate (Object)

Applies To:

Document object

Description:

Represents the date on which the document was modified.

Action:

Read-only

Example:

This example shows how to display the date on which the document was modified.

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

Navigation (Object)

Member of:

Toolbars collection

Description:

Represents the Navigation toolbar in the application. The Navigation toolbar is used to return to the Dashboard section from another section when the Section catalog, Section titlebar, toolbars and menus have been turned off. The Navigation toolbar is hidden by default, but can be enabled by scripts. When activated, it is available in all sections and includes the Back, Forward and Dashboard Home buttons.

Constants:

The Type property of the Toolbar (Object) uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows how to enable the Navigation toolbar:

```
//Syntax for turning on the Navigation toolbar
```

```
Toolbars["Navigation"].Visible=true;
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

NumericRange (Object)

Member of:

Bullet object, Speedometer object, Thermometer object, TrafficLight object

Description:

Represents the numeric scale assigned to a gauge object and its properties.

Example:

This example shows how to disable the auto-scale feature for a speedometer, and sets the major and minor scales, and major and minor intervals:

```
Speedometer.NumericRange.AutoScale = false
Speedometer.NumericRange.MajorInterval = "20"
Speedometer.NumericRange.MaxScale = "100"
Speedometer.NumericRange.MinorInterval = "10"
Speedometer.NumericRange.MinScale = "0"
```

Methods:

None

Properties:

Read-write: AutoScale as Boolean, MajorInterval as Number, MaxScale as Number, MinorInterval as Number, MinScale as Number

Objects:

None

OLAPCatalog (Object)

Member of:

OLAPQuery object

Description:

Represents the OLAPTable Catalog. This object provides access to a multi-dimensional databases (MDD) such as the dimensions, level, member, measures, and properties.

Example 1:

The following example shows how to use a try-catch block to retrieve the dimensions from two OLAPCatalogs. If OLAPCatalog dimensions cannot be retrieved (no connection is made, the query does not exist etc.), an exception occurs and "No OLAPQueryx Dimensions retrieved" displayed in the Console window. If dimensions are retrieved, "OLAPQueryx Dimensions are retrieved" is displayed in the Console window.

```
try
{
QRetrieve =ActiveDocument.Sections["OLAPQuery1"]
QRetrieve.Catalog.Dimensions.RetrieveDimensions()
Console.Writeln("OLAPQuery1 Dimensions retrieved")
}
catch(e)
{
Console.Writeln("No OLAPQuery1 Dimensions retrieved")
}
try
{
QRetrieve =ActiveDocument.Sections["OLAPQuery"]
QRetrieve.Catalog.Dimensions.RetrieveDimensions()
Console.Writeln("OLAPQuery Dimensions retrieved")
}
catch(e)
{
Console.Writeln("No OLAPQuery Dimensions retrieved")
```

Example 2:

This example shows how to count the number of dimensions in the OLAPCatalog and display the value in an Alert box.

```
Application.Alert("The number of dimensions is:" +
 ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions.Count)
```

Example 3:

This example shows how to programmatically add topics to a Data Model section:

```
CatItem =
ActiveDocument.Sections["DataModel"].DataModel.Catalog.CatalogItems["PCW_ITEMS"]
ActiveDocument.Sections["DataModel"].DataModel.Topics.Add(CatItem)
```

Collections:

OLAPDimensions, OLAPValue Measures

OLAPCatalogNew (Object)

Member of:

(CubeQuery) Query object

Description:

Provides access to the dimensions of the connected CubeQuery database.

Example:

This example shows how to display the parent and dimension name of the member in the Console window:

```
var ccut = Sections["Query"].Catalog.Dimensions["Market"]["East"]["Connecticut"];
Console.Writeln(ccut.ParentName + ":" + ccut.Name);
```

Objects:

OLAPDimensionNew

OLAPConnection (Object)

Member of:

(OLAP) Query object

Description:

Represents an OLAP Query connection file (.oce) or the database connection. The OLAPQuery connection file is used to capture and store connection information such as the connection software, the database software, and the address of your database server and your database user name for a multi-dimensional database.

Example 1:

This example shows how to connect an OLAP database using an OCE saved locally:

```
//Connecting to OLAP
MyConnection=ActiveDocument.Sections["OLAPQuery"].Connection
MyConnection.Open("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\Essbase.oce")
MyConnection.Username="Essbase"
MyConnection.SetPassword("Essbase")
MyConnection.Connect()
```

Example 2:

This example shows how to connect to the OLAP Query.oce which was included in the sample files installed with Interactive Reporting 6.x. It also sets the user name and password to "hyperion":

```
MyConnection=ActiveDocument.Sections["OLAPQuery"].Connection
MyConnection.Open("c:\\Program Files\\Hyperion\\BIPlus\\data\\Open Catalog Extensions\
\Sample OLAP Query.oce")
MyConnection.Username="hyperion"MyConnection.SetPassword("hyperion")MyConnection.Connect
()
```

Method:

Connect(), Disconnect(), Open(Filename As String), Save(), SaveAs(Filename As String), SetPassword (Password As String)

Properties:

Read-only: Property Connected As Boolean, Property CubeName as String, Property Database as BqDatabase, Property ShowLevelProperties as Boolean

Read-write: Property Username As String

OLAPDimensionNew (Collection)

Member of:

OLAPCatalogNew (CubeQuery) object

Description:

Represents all dimension values associated with a CubeQuery. Dimensions are composed of values called members, which are arranged in a hierarchical structure. That is, it is a perspective or view of a specific dataset.

The Count property reads the number of dimensions in the Essbase database. If the query is disconnected this property return zero and the collection is empty. Otherwise, the Count property for this object returns the dimension at the given index, or with the given name, if a dimension of that name is in the data layout. If Index is given as an integer, its value must be between 1 and the value returned by the Count property. If Index is not within this range an ItemNotFound exception is thrown. If Index is given as a string, it must match one of the names of the dimensions in the database. If not, an ItemNotFound exception is thrown.

Example 1:

This example shows how to count the number of members in level 1 of a dimension. If the number of members is greater than the value used in the CatalogDisplayMembers (Property), then only the values in the CatalogDisplayMembers (Property) are put into the collection and the value returned by the Count (Property) is the same as the value of CatalogDisplayMembers:

```
var opts = Sections["Query"].QueryOptions;
var products = Sections["Query"].Catalog.Dimensions["Product"];
if (opts.CatalogDisplayMembers == products.Count) {
    // There may be more members at level 1 of the dimesion
    // than will be returned by the Item method. To get access
    // to all of them set CatalogDisplayMembers to 0.
}
```

Example 2:

This example shows how to display the name of the member represented by a selected label:

```
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);
}
```

Example 3:

This example show how to show the unique name of the member represented by the label:

```
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);
}
```

Methods:

Item value as name or index, RetrieveDimensions()

Properties:

Read only: Count as number

OLAPDimensions (Object)

Member of:

Catalog object

Description:

Represents all dimension values associated with an OLAPQuery. Dimensions are categories of information.

Example:

This example shows how to count the number of dimensions in an OLAPQuery and display the results in an Alert box:.

```
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions.Count)
```

Methods:

Item(Value As NameOrIndex), RetrieveDimensions ()

Properties:

Read-only: Attribute Dimension as Boolean, Count as Number, Name as String, UniqueName as String

OLAPFilter (Object)

Member of:

OLAPFilters collection

Description:

Represents an individual filter in the data layout. Only the variable property of a filter may be modified by way of the Object Model. The Dimension and Member properties are read-only. Dimension and Member property values are derived from the current alias table.

Example 1:

This example shows how to read if a filter is a variable or not:

```
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;
```

Example 2:

This example shows how to read if a filter has been to set on a member:

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

Example 3:

This example shows to add a filter:

```
var filters = Sections["Query"].Filters;
if (filters.Count > 0) {
    var filter = filters[1];
    var filterDim = filter.Dimension; // a string containing the dimension name
    var filterMem = filter.Member; // a string containing the member name
} else {
    filters.Add("Product", "100-20");
}
```

Methods:

Remove

Properties:

Read only: Dimension as String, Member as String, Name as String

Read-write: Ignore as Boolean, Variable as Boolean

OLAPFilters (Collection)

Member of:

(CubeQuery) QuerySection object

Description:

Represents a collection of filters defined in the data layout for the query. This is a standard collection and may be indexed numerically or by dimension name. A Member name cannot be used because two dimensions may have members with the same name.

The Add (Method) adds a new filter to the end of the collection. If there is a filter defined for the given dimension, or the dimension or member do not exist, an exception is thrown. The MemberLocation parameter must be a fully qualified path to a member, including its dimension and all intermediate members, delimited by periods ".". If the dimension name or any member name contains a period, it must be escaped using a backslash. The optional Variable arguments can be used to set the initial values of the OLAPFilter Variable property If the given member location cannot be mapped to a valid dimension and member, an exception is thrown.

Example:

This example shows how to show the count of filters in the query (note that if the value of Count is zero, there are no filters conditions):

```
Application.Alert(ActiveDocument.Sections["Query"].Filters.Count)
```

Methods:

Add(Dimension as String, MemberLocation as String, [optional] Variable as Boolean), Item(Value as Name or Index), RemoveAll

Properties:

Read only: Count as Number

OLAPLabel (Object)

Member of:

OLAPLabels collection

Description:

Represents a top or side label within an OLAP report.

Methods:

AddFilter (OperatorType As BqOperatorType, DataOperator as BqOperator, Value1 As String, Value2 As String, [optional] Variable as Boolean) AddFilterValue(MemberName As String, Operator As BqOperator), AddTotal, AutoSizeHeight(), AutoSizeWidth, DrillDown(),

DrillThrough([optional] PromptOption as BqDrillThroughPrompt, [optional] PivotName As String, [optional] PromptDialog as Boolean, DrillUp(), Remove()

Properties:

Read-only: Property Name As String, Property SortFactName as String, Property SortFunction as BqSortFunction, SortOrder as BqSortOrder

Read-write: Property IgnoreFilter as Boolean

Collections:

LabelValues As OLAP Query TopLabels and SideLabel collections.

OLAPLabels (Collection)

Member of:

OLAPQuerySection object

Description:

Consists of the OLAP Query TopLabels and SideLabels collections. These collections correspond to the labels within a OLAP Query section. These are columns added to the side and top labels groups in the outliner.

Interactive Reporting supports different OLAP datasource, including OLEDB for DB, Essbase, and MetaCube. Depending on the data source, different filter operators are supported. If you use an operator that is not applicable for the data source, an exception is thrown.

OLEDB for OLAP only enables users to add filter values by selecting them from the Show Values pane in the Filter dialog box. Essbaseand MetaCube enable users to type in a string as the default way of adding filter values and provides a Filter dialog. When not using Show Values, use the optional parameter value in the Add () method for filtering.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Constants:

The OLAPLabels (Collection) uses the BqOperator constant group, which consists of these values:·

- bqOperatorEqual
- bqOperatorGreaterThan
- bqGreaterThanOrEqual

- bqOperatorLessThan
- bqOperatorLessThanOrEqual
- bqOperatorNotEqual

Examples:

The following scripts show an OLAPQuery with these values:

- "State (All)" as a side label
- "Year (All)" as a top label
- "Amount" as a measure in the user interface

The scripts invoke the Console window to monitor each line that is executed.

Observe the hierarchy of dimensions and labels when you create scripts that include OLAP objects, methods, and properties. Levels from the same dimension must be grouped together in both side and top labels. Additionally, do not break dimensional hierarchy. For example, Year must proceed Quarter, which proceeds Month.

Example 1:

This script shows how to remove all labels. If label don't exist, use a try-catch statement:

```
OQPath = ActiveDocument.Sections["OLAPQuery"]
/*try-catch used here in case labels don't exist*/
Console.Writeln("Step1")
try{OQPath.SideLabels[1].Remove()}
catch(e)
{Console.Writeln(e.toString())}
Console.Writeln("Step2")
OQPath.Process()
Console.Writeln("End")
```

Example 2:

This script shows how to add the "State" value as the side label. Note the hierarchy in which the dimension "Location" proceeds the level "State".

```
Console.Writeln("Start")
OQPath = ActiveDocument.Sections["OLAPQuery"]
Console.Writeln("Step1")
OQPath.SideLabels.Add('Location.State')
Console.Writeln("Step2")
OQPath.Process()
Console.Writeln("End")
```

Example 3:

This script adds the Arizona state abbreviation code as a filter value. The *State* side label must already exist.

```
Console.Writeln("Start")
OQPath = ActiveDocument.Sections["OLAPQuery"]
```

```
Console.Writeln("Step1")
OQPath.SideLabels[1].AddFilterValue('AZ',bqOperatorEqual)
Console.Writeln("Step2")
OQPath.Process()
Console.Writeln("Step3")
OQPath.Activate()
Console.Writeln("End")
```

Example 4:

In this example, the state added in Example 3 is removed and a filter can be added from the drop-down list:

```
Console.Writeln("Start")
OQPath = ActiveDocument.Sections["OLAPQuery"]
Console.Writeln("Step1")
//try-catch used here in case labels don't exist*/try{OQPath.SideLabels[1].Remove()}
catch(e){}
Console.Writeln("Step2")
OQPath.SideLabels.Add('Location.State')
Console.Writeln("Step3")fv = DropDown1[DropDown1.SelectedIndex]
Console.Writeln("Step4, fv is " + fv)
OQPath.SideLabels[1].AddFilterValue(fv,bqOperatorEqual)
Console.Writeln("Step5")
OQPath.Process()
Console.Writeln("Step6")
OQPath.Activate()
Console.Writeln("End")
```

Example 5:

When you do not want to use the ShowValues property of filtering, use the Add () method as shown below.

```
//When NOT using Show Value method of filteringActiveDocument.Sections["OLAPQuery"].
TopLabels.Add('Time.Year','1999')
```

Example 6:

When you want to use the ShowValues property of filtering, use the AddFilterValue method.

```
ActiveDocument.Sections["OLAPQuery"].TopLabels.Add('Gender.Gender')
ActiveDocument.Sections["OLAPQuery"].TopLabels["Gender"].AddFilterValue('M',bqOperatorEq
ual)
```

Methods:

Add(String LevelName), Function Item(Value As NameOrIndex) As OLAPLabel, RemoveAll()

Properties:

Read-only: Property Count As Number

TopLabels Methods and Properties:

```
ActiveDocument.Sections["OLAPQuery"].TopLabels.Add('LevelName'[,optionalMeasure])
```

```
ActiveDocument.Sections["OLAPQuery"].TopLabels.Item()
ActiveDocument.Sections["OLAPQuery"].TopLabels.RemoveAll()
ActiveDocument.Sections["OLAPQuery"].TopLabels.Count
ActiveDocument.Sections["OLAPQuery"].TopLabels['LevelName'].Remove()
ActiveDocument.Sections["OLAPQuery"].TopLabels['LevelName'].FilterOperator= bqOperator
ActiveDocument.Sections["OLAPQuery"].TopLabels['LevelName'].AddFilterValue('LevelName','
MeasureName')
```

SideLabels Methods and Properties:

```
ActiveDocument.Sections["OLAPQuery"].SideLabels.Add('LevelName'[,optionalMeasure])
ActiveDocument.Sections["OLAPQuery"].SideLabels.Item()ActiveDocument.Sections["OLAPQuery
"].SideLabels.RemoveAll()
ActiveDocument.Sections["OLAPQuery"].SideLabels.Count
ActiveDocument.Sections["OLAPQuery"].SideLabels['LevelName'].Remove()
ActiveDocument.Sections["OLAPQuery"].SideLabels['LevelName'].FilterOperator= bqOperator
ActiveDocument.Sections["OLAPQuery"].SideLabels['LevelName'].AddFilterValue('LevelName',
'MeasureName')
```

OLAPLevelOrHierarchy (Collection)

Member of:

OLAPDimension collection

Description:

Represents a collection of OLAP dimension levels.

Example:

This example shows how to display the name of the level and the attribute dimension status in Alert boxes:

```
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Location"].Name)
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Location"].AttributeDimen
sion)
```

Methods:

Item(Value NameOrIndex)

Properties:

Read-only: Property AttributeDimension as Boolean, Property ContainsHybridAnalysisData as Boolean, Property Count as Number, Property Name As String

OLAPLevelOrHierarchy (Object)

Member of:

OLAPDimension collection

Description:

Represents a level within a dimension. For example, using the members listed in a Location dimension, Japan, USA, and France belong to the Country level. San Francisco, Paris, Tokyo, and Rome belong to the City level. and 35 Main Street belongs to the Address level.

Example:

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)
```

Methods:

Item(Value NameOrIndex)

Properties:

Read-only: Property AttributeDimension as Boolean, Property ContainsHybridAnalysisData as Boolean, Property Count as Number, Property Name As String, Property ParentName as String

OLAPLevelOrHierarchyNew (Collection)

Member of:

OLAPDimensionNew collection

Description:

Represents a collection of members at the selected dimension level in a CubeQuery.

The Count property of the collection reads the number of members in this level of this dimension. If the number of members is greater than the value of the CatalogDisplayMembers (Property), then only CatalogDisplayMembers are put into the collection and the value returned by the Count property is the same as the value of CatalogDisplayMembers (Property).

Example 1:

This example shows how to display the number of members in the "Year" dimension:

```
Alert(ActiveDocument.Sections["Query"].Catalog.Dimensions["Year"].Count)
```

Example 2:

This example shows how to count the number of members and return them to the Item (Method):

```
duct"]["Colas"];
if (opts.CatalogDisplayMembers == products.Count) {
    // There may be more members at level 2 of the dimesion
    // than will be returned by the Item method. To get access
    // to all of them set CatalogDisplayMembers to 0.
```

```
}
y the Count property is the same as the value of EssbaseSpecific.CatalogDisplayMembers.

Examples
```

Example 3:

This example shows how to return the name of the dimension:

```
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);
}
```

Methods:

OLAPLevelOrHierarchyNew value as Name or Index.

Properties:

Read only: AttributeDimension as Boolean, Count as Number, Name as String, UniqueName as String

OLAPMeasure (Object)

Member of:

OLAPMeasures collection

Description:

Represents a measure within an OLAP Query report.

Example:

This example shows how to add a filter value to a "Profit" measure. In this example, the operator used equals 13,438:

```
Console.Writeln("Start")ActiveDocument.Sections["OLAPQuery"].Measures["Profit"].AddFilte
rValue('1',bqOperatorEqual,'13438')Console.Writeln("End")
```

Methods:

AddFilter(dataOperator as BqOperator, columnIndex As String , Value As String, [optional]IsVariable As Boolean), AddFilterValues (ColumnIndex As String, Operator As BqOperator, MeasureValue A String), AutoSizeHeight(), AutoSizeWidth(), DrillDown(), DrillUp(), Hide(), Remove(), RemoveFilterValue([optional] Value As NameOrIndex) UnHide()

Properties:

Read-only: Property Count as Number, Property Name As String, Property ParentName as String

Read-write: Property DataFunction as BqDataFunction, Property HorizontalAlignment as BqHorizontal, IgnoreFile as Boolean, Property NumberFormat as String, Property VerticalAlignment as String

OLAPMeasures (Collection)

Member of:

OLAPQuerySection object, OLAPCatalog object

Description:

Consists of the OLAP Query Measures collections. These collections correspond to measures in an OLAP Query section. These are columns added to the side and top labels groups in the outliner.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example shows how to add a measure limit called "Units Plan":

```
ActiveDocument.Sections["OLAPQuery"].Measures.Add("Units Plan")
```

Methods:

Add(Measure As String), AddComputed(ColumnName as String, Expression as String), Item(Value As NameOrIndex) As OLAPLabel, ModifyComputed (ColumnName as String), Expression As String), RemoveAll(), ShowAll()

Properties:

Read-only: Property Count As Number

OLAPMemberSelectors (Collection)

Member of:

QueryLabel object

Description:

Represents a collection of members defined for a label in CubeQuery.

The Count property of this collection reads the number of member selectors for the parent label. If the value of Count is zero, there are no member selectors.

The Item method of this collection returns the member select at the given index. If Index is given as a numeric value it must be between 1 and the value returned by the Count property. If Index is not within this range an ItemNotFound exception is thrown. If Index is given as a string it must match the member name of one of the member selectors. If it does not an ItemNotFound exception is thrown.

The RemoveAll method removes all member selectors from the label. This method should not be used as an the argument to MemberSelectors.RemoveAll() in anInteractive Reporting document file (bqy) It is only meant to be used for Interactive Reporting document file run as jobs. By default the KeepLabel argument defaults to false, meaning the label is removed. If set to true, the label is kept but has no member selectors.

Example 1:

This example shows how to display the number of members for the "Market" dimension in an Alert box:

```
Alert(ActiveDocument.Sections["Query"].Rows["Market"].MemberSelectors.Count)
```

Example 2:

This example shows how to return the member selector at a given index:

```
var selector = Sections["Query"].Rows[1].MemberSelectors[1];
var selector = Sections["Query"].Rows[1].MemberSelectors["100-20"];
var selector = Sections["Query"].Rows[1].MemberSelectors.Item(1);
var selector = Sections["Query"].Rows[1].MemberSelectors.Item("100-20");
```

Example 3:

This example shows how to remove all member selectors, but retain the label:

```
Sections["Query"].Rows[1].MemberSelectors.RemoveAll();
Sections["Query"].Rows[1].MemberSelectors.RemoveAll(true);
```

Methods:

Item value as name or index, RemoveAll([optional] KeepLabel as Boolean)

Properties:

Read only: Count as Number:

OLAPMemberSelector (Object)

Member of:

OLAPMemberSelectors (Collection)

Description:

Represents an individual member selector object for a label in CubeQuery. A member selector object is read only. It cannot be modified or removed. If you do not want to include a selected member, use the Ignore (Property) to deselect an item from being used in the query without removing it.

Example:

This example shows how to ignore the member selector 100–20:

```
var selectors = Sections["Query"].Rows[1].MemberSelectors;
for (var i = 1; i <= selectors.Count; i++) {
    var s = selectors[i];
    if (s.Name == "100-20")
        s.Ignore = true;
}
```

Properties:

Read-write: DisableSelection as Booleans, NumberFormat as String

Read only: Name as String

OLAPQuerySection (Object)

Member of:

Sections collection

Description:

Represents an OLAP Query Section.

Note: You can use the OLAPQuerySection (Object) to process OLAP queries but not build them.

Example:

This example shows how to activate and process an OLAP Query:

```
ActiveDocument.Sections["OLAPQuery"].Activate()
ActiveDocument.Sections["OLAPQuery"].Process()
```

Methods:

Activate(), Copy(), DownloadToResults(), Duplicate(), Export([optional]Filename As String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders, [optional]Prompt as Boolean, ExportToStream([optional] Filename As String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders As Boolean, [optional] DataStreaming as Boolean, [optional] Prompt As Boolean), PrintOut([optional] FromPage As Number, [optional] ToPage As Number, [optional] Copies As Number, [optional] Filename As String, [optional] Prompt as Boolean, Recalculate(), Remove(), SetDrillThrough(MapOrUnMap as Boolean, RelationalQueryName as String, DimensionOrRelationalTopicName as String, FactName as String) ShowAsChart()

Properties:

Read-only: Property Active As Boolean, Property Type As BqSectionType

Read-write: Property CSSExport as Boolean, Property Database Totals as Boolean, Property DrillDownDisplay as BqDrillDownDisplay, Property ExportWithoutQuotes As Boolean, Property HardwireMode as Boolean, Property HTMLExportBreakColumnCount As Number, Property HTMLExportBreakRowCount As Number, Property HTMLHorizontalPageBreakEnabled as Boolean, Property HTMLHorizontalPageBreakUnits as BqHTMLPageBreakUnits, Property VerticalPageBreakEnabled as Boolean, Property HTMLVerticalPageBreakUnits as BqHTMLPageBreakUnits, Property IncludeInProcessAll as Boolean, Property Name As String, Property ProcessSequenceNum as Number, Property ShowOutliner As Boolean, Property ShowSlicer as Boolean, Property ShowSortLine as Boolean, Property Visible As Boolean

Objects:

Connection as OLAPConnection, DataLabels As PivotDisplay, CornerLabels as PivotDisplay, DBSpecific as DBSpecific, Catalog as OLAPCatalog

Collections:

UserValues as UserValues, TopLabels as OLAPLabel, SideLabels as OLAPLabel, Measures as OLAPMeasures, Slicers as OLAPSlicer, Events as Events

OLAPResults (Object)

Member of:

Sections collection

Description:

Represents an OLAP results set in a Table Catalog.

Example:

This example shows how to remove OLAPResults sets. This also removes the parent OLAPQuery and other sections dependent on the OLAPResults set.

```
ActiveDocument.Sections["OLAPResults"].Remove()
```

Methods:

Activate(), Copy(), Duplicate, Export([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] Prompt as Boolean, ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean), GetCell(nRow as Number, nCol as Number), PrintOut([optional] FromPage as Number, [optional] ToPage as Number, [optional] Copies as Number, [optional] Filename as String, [optional] Prompt as Boolean), Recalculate{}, Remove{}

Properties:

Read-only: Property Active as Boolean, Property RowCount as Number, Property Type as BqSectionType

Read-write: Property BackgroundAlternateColor as BQColorType, Property BackgroundAlternateFrequency as Number, Property BackgroundColor as BqColorType, BackgroundShowAlternateColor as Boolean, Property BorderColor as BqColorType, Property BorderWidth as Number, Property CSSExport as Boolean, Property ExportWithoutQuotes as Boolean, HTMLExportBreakRowCount as Number, HTMLVerticalPageBreakEnabled as Boolean, Property HTMLVerticalPageBreakUnits as BqHTMLPageBreakUnits, Property Name as String, Property ShowOutliner as Boolean, Property ShowRowNumbers as Boolean, Property Visible as Boolean

Collections:

UserValues as UserValues, Columns As Columns, Limits as Limits, SortItems as SortItems

OLAPSlicer (Object)

Member of:

OLAPSlicer object

Description:

Represents a slicer within an OLAP Query report.

Example:

This example shows how to remove the “Q3” slicer value from the query:

```
ActiveDocument.Sections["OLAPQuery"].Slicers["Q3"].Remove()
```

Methods:

Remove()

Properties:

Read-only: Property Name As String

OLAPSlicers (Collection)

Member of:

OLAPQuerySection object

Description:

Consists of the OLAP Query Slicers (Collection). These collections correspond to the slicer within a OLAP Query section. This is the column added to the slicer in the outliner.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example: 1:

This example shows how to add a slicer that limits the scope to Oakland, California:

```
ActiveDocument.Sections["OLAPQuery"].Slicers.Add('ProductLocation.{hierachy}. Store Type','USA.California.Oakland')
```

Example 2:

This example shows how to add a slicer that limits the scope to a company called Eastern Winds:

```
ActiveDocument.Sections["OLAPQuery"].Slicers.Add('Store Region.Region.Territory.Country.State.City.Name', 'Americas.North America.USA.New York.Manhattan.Eastern Winds New York')
```

Methods:

Add(LevelName As String, MemberName As String, Variable As Boolean) As OLAPSlicer, Item(Value As NameOrIndex) As OLAPSlicer, RemoveAll()

Properties:

Read-only: Property Count As Number

Read-Write: Property VariableSlicerMode as BqSlicerDisplayOptions

Methods and Properties Syntax:

```
ActiveDocument.Sections["OLAPQuery"].Slicers.Add('LevelName','UniqueName')
ActiveDocument.Sections["OLAPQuery"].Slicers.Item()
ActiveDocument.Sections["OLAPQuery"].Slicers.RemoveAll()
ActiveDocument.Sections["OLAPQuery"].Slicers.Count
ActiveDocument.Sections["OLAPQuery"].Slicers['SlicerName'].Remove()
```

OLAPValues (Object)

Member of:

OLAPLevelOrHierarchy object

Description:

Represents the content values (members) in a level.

Example:

This example shows how to count the number of members in a level:

```
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Location"]
["Values"].Count)
```

Methods:

Item(NameOrIndex))

Properties:

Read-only: Property Count As Number, Property Name As String, Property ParentName As String

OLAPValues (Collection)

Member of:

OLAPLevelOrHierarchy object

Description:

Represents the content values in a dimension.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example shows how to count values in dimensions:

```
Alert(ActiveDocument.Sections["OLAPQuery"].Catalog.Dimensions["Location"]
["Values"].Count)
```

Methods:

Item(Value NameOrIndex)

Properties:

Read-only: Property Count As Number,

PageCount Field (Object)

Member of:

Fields collection

Description:

Sets the current page of the total number of pages.

Example:

This example shows how to change the font color of the PageCount Field to red:

```
ActiveDocument.Sections["Report"].ReportHeader.Fields["PageCount Field"].Font.Color =
bqRed
```

Methods:

Layer(Value As BqLayer), Spring (Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects:

UserValues, LineFormat, FillFormat, FontFormat object

PageFooter (Object)

Member of:

ReportSection object

Description:

Represents the attributes of the page footer group.

Example:

This example shows how to suppress the display of the page footer:

```
ActiveDocument.Sections["Report"].PageFooter.Visible = false
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects:

LineFormat object, FillFormat object, Tables collection, Fields collection, Shapes collection, Shapes collection, Pivots collection, Pivot collection, Chart collection

PageHeader (Object)

Member of:

ReportSection object

Description:

Represents the attributes of the page header group.

Example:

This example shows how to set the line color of the page header to red:

```
Documents["Salesreport.bqy"].Sections["Report"].PageHeader.Line.Color = bqRed
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Object and Coll.:

LineFormat object, FillFormat object, Tables collection, Fields collection, Shapes collection, Shapes collection, Pivots collection, Pivot collection, Chart collection

PageNm Field (Object)

Member of:

Fields collection

Description:

Sets the current page number.

Example:

This example shows how to align vertically the text in PageNum Field at the top of the field:

```
ActiveDocument.Sections["Report"].PageHeader.Fields["PageNm Field"].VerticalAlignment =
bqAlignTop
```

Methods:

Layer(Value As BqLayer value), Spring(Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType (Number)

Objects:

UserValues, LineFormat, FillFormat, Font

PageXofY Field (Object)

Member of:

Fields collection

Description:

Sets the current page of the total number of pages.

Example:

This example shows how to add a green, two point, dash style to the PageXofY field (Object):

```
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["PageXofY
Field"].Line.DashStyle = 4
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["PageXofY Field"].Line.Color
= bqGreen
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["PageXofY Field"].Line.Width
= 2
```

Methods:

Layer(BqLayer value), Spring(String Name), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType (Number)

Objects:

UserValues, LineFormat, FillFormat, Font

Paging (Object)

Member of:

Toolbars collection

Description:

Represents the abbreviated version of the EPM Workspace 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 script command is ignored, no exception is thrown, and the script continues.

Constants:

The Type property of the Toolbar (Object) uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows you enable the Paging toolbar in the application:

```
//Syntax for turning on the Formatting toolbar
Toolbars["Formatting"].Visible=true;
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

Parentheses (Collections)

Member of:

Limits collection, AppendQueries collection

Description:

A collection of parentheses attributes for a limit value or appended query object.

To nest parentheses between limit items, add parentheses around the largest range of limit objects before nesting additional parentheses. For example, if *State*, *Amount Sales*, and *City* are on the limit line, type:

```
ActiveDocument.Sections["Query"].Limits.Parentheses.Add("State",
"City")
```

before typing:

```
ActiveDocument.Sections["Query"].Limits.Parentheses.Add("State",
"Amount Sales")
```

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns a collection item at a particular index or that has a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example 1:

This example shows how to remove all parentheses attributes set on the values of the limit line:

```
ActiveDocument.Sections["Query"].Limits.Parentheses.RemoveAll()
```

Example 2:

This example shows how to add parentheses:

```
ActiveDocument.Sections["Query"].Limits.Parentheses.Add("State"), "City")
```

Example 3:

This example shows how to add parentheses between the first query and the second query:

```
var firstQuery = ActiveDocument.Sections["Query"].AppendQueries.Item(1);

var secondQuery = ActiveDocument.Sections["Query"].AppendQueries.Item(2);
```

```
ActiveDocument.Sections["Query"].AppendQueries ["UnionParentheses"].Add(firstQuery,
secondQuery);
```

Methods:

Add(BeginLimit as String, EndLimit as String) or Add(AppendQuery BeginQuery, AppendQuery EndQuery), Item(Value as NameOrIndex), RemoveAll()

Properties:

Read-only: Count as Number

Objects:

Parenthesis object

Parenthesis (Object)

Member of:

Limits collection, Parentheses collection

Description:

For a Limits collection, returns or sets parentheses around values on limit lines. In Interactive Reporting, parentheses represent enclosed sub-operations. Sub-operations enable you to override the default evaluation order in operations using AND or OR operators.

For an AppendedQueries collection, this object nests parentheses () for the evaluation order of an appended query. The expression must include two queries or more. The default evaluation order is left to right.

Example 1:

This example shows how to remove all parentheses on the Limit line:

```
ActiveDocument.Sections["Query"].Limits.Parentheses.RemoveAll()
```

Example 2:

This example shows how to remove the first parenthetical expression:

```
ActiveDocument.Sections["Query"].Limits.Parentheses[1].Remove();
```

Example 3:

This example shows how to count the number of parenthetical expressions on the Limit line:

```
(ActiveDocument.Sections["Query"].Limits.Parentheses.Count);
```

Example 4:

This example shows how to remove the beginning query in an appended query expression:

```
ActiveDocument.Sections["Query"].AppendQueries["Parentheses"]
["Query,Query2"].BeginQuery.Remove()
```

Methods:

Remove()

Properties:

Read-only: BeginLimitName as String, EndLimitName as String, Name as String, BeginQuery as String, EndQuery as String

Path Field (Object)

Member of:

Fields collection

Description:

Returns the full path name of documents.

Example:

This example shows how to wrap the file path name within a path field:

```
ActiveDocument.Sections["Sales Report"].Body.Fields["Path Field"].TextWrap = true
```

Methods:

Layer(Value As BqLayer), Spring(Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects/Coll.:

UserValues, LineFormat, FillFormat, Font

PieChart (Object)

Member of:

ChartSection object

Description:

The PieChart (Object) represents Pie Chart settings.

Example:

This example shows how to define settings for a Pie Chart:

```
with(ActiveDocument.Sections["Chart"])
{
            PieChart.ShowLabels = true
            PieChart.ShowValues = true
            PieChart.ShowPercentages = false
            PieChart.ShowAllPositive = False
}
```

Methods:

None

Properties:

Read-write: Property Height As Number, Property Rotation As Number, Property ShowAllPositive As Boolean, Property ShowLabels As Boolean, Property ShowPercentages As Boolean, Property ShowValues As Boolean

(Live) PieChart (Object)

Member of:

Dashboard objects

Description:

Represents a (Live) PieChart object and its properties.

The following PieChart object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Subtype—BqLivePieChartType
- Type—BqShapeType

Example:

The example shows how to set common Live Pie Chart properties:

```
PieChart.Comments = "Market Share"
PieChart.Effect = bqLiveChartEffect3D
PieChart.Explode = true
PieChart.Locked = false
PieChart.Name = "Pie Chart Market Share"
PieChart.SourceSectionName = "Results"
PieChart.Subtype = bqLivePieChartTypePie
```

```
PieChart.Type = bqLivePieChart
PieChart.Visible = true
```

Methods:

None

Properties:

Read-write:: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Effect as BqLiveChartEffect, Explode as Boolean, Locked as Boolean, Name as String, SourceSelectionName as String, Subtype as BqLivePieChartType, Type as BqShapeType, Visible as Boolean

Objects:

UserValues collection, Placement object, Data object, Subcomponents object

Pivot (Object)

Member of:

Shapes (Collection)

Description:

Represents an embedded Pivot section object within the Shapes collection.

Example:

This example shows how to always display scrollbars in embedded Pivots.

```
ActiveDocument.Sections["Dashboard"].Shapes["Pivot1"].OnClick()
ActiveDocument.Sections["Dashboard"].Shapes["Pivot1"].ScrollbarsAlwaysShown = true
```

Methods:

OnClick()

Properties:

Read-only: Property Name as String, Property Type as BqShape

Read-write: Property Alignment as BqHorizontalAlignment, Property Property ShowAlwayShown as Boolean, Property Text as String, Property VerticalAlignment as BqVerticalAlignment, Property Visible as Boolean

Collections:

UserValues

PivotFact (Object)

Member of:

PivotFacts collection

Description:

Represents a fact item in a pivot.

Example:

This example shows how to show and autosize the height and width of the Unit Sales fact item:

```
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].AutoSizeHeight()
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].AutoSizeWidth()
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].UnHide()
```

Methods:

AutoSizeHeight(), AutoSizeWidth(), Hide(), Remove(), UnHide()

Properties:

Read-write: Property DataFunction As BqDataFunction, Property Index As Number, Property NumberFormat As String

Read-only: Property Name as String

PivotFacts (Collection)

Member of:

PivotSection object

Description:

Facts set within a pivot section. These are columns added to the facts groups in the outliner.

Tip: All collections have a method named "Item(NameOrIndex)." This is the default method for all collections and it returns an item in the collection at a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to add request items to facts collections:

```
ActiveDocument.Sections["Pivot"].Facts.RemoveAll()
```

```
ActiveDocument.Sections["Pivot"].Facts.Add("Year")
ActiveDocument.Sections["Pivot"].Facts.Add("Region")
```

Methods:

Add(RequestItemName As String, [optional] Index as Number) As PivotFact, AddComputedItem(Name As String Expression as String, [optional] Number As Index) As PivotFact, Item(Value as NameOrIndex) As PivotFact, RemoveAll()

Properties:

Read-only: Property Count As Number

PivotLabel (Object)

Member of:

Pivot Labels collection

Description:

Represents a top or side label in a Pivot report.

Example:

This example shows how to size the height and width of pivot labels automatically, and how to hide pivot facts:

```
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].AutoSizeHeight()
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].AutoSizeWidth()
Documents["Sample 1.bqy"].Sections["SalesPivot"].Facts["Unit Sales"].Hide()
```

Methods:

AddTotals(), AutoSizeHeight(), AutoSizeWeight(), PivotTo([Index As Number]), Remove(), SortByFact(FactName As String, SortFunction As BqSortFunction, [optional] SortOrder As BqSortOrder), SortByLabel([SortOrder As BqSortOrder])

Properties:

Read-only: Property Name as String, Property SortFactName As String, Property SortFunction As BqSortFunction, Property SortOrder As BqSortOrder

Read-write: Property Index as Number,

Collections:

LabelValues As Pivot, TopLabels and SideLabel collections

PivotLabels (Collection)

Member of:

PivotSection object

Description:

Consists of the Pivot TopLabels and SideLabels collections. These collections correspond to the labels within a pivot section. These are columns added to the side and top labels groups in the outliner.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example shows how to add a number of request items to the side and top labels collections:

```
With(ActiveDocument.Sections["Pivot"])
{
          TopLabels.RemoveAll()
          SideLabels.RemoveAll()
          TopLabels.Add("Year")
          SideLabels.Add("Grape")
          SideLabels.Add("Winery")
}
```

Methods:

Add(RequestItemName As String, [optional] Index as Number As PivotLabel), Item(Value as NameOrIndex) As PivotLabel, RemoveAll()

Properties:

Read-only: Property Count As Number

PivotLabelTotal (Object)

Member of:

SideLabel object, TopLabel object

Description:

Returns an additional row or column containing the total for a top label or side label object. This is like selecting a pivot side label column or top label row and selecting Add Totals from the Pivot menu.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example 1:

This example shows how to add a Totals column (using the default data function of "sum") for the "Product Line" from the Totals collection or from the TopLabels collection:

```
ActiveDocument.Sections["SalesPivot"].TopLabels["Product Line"].Totals.Add()
or
ActiveDocument.Sections["SalesPivot"].TopLabels["Product Line"].AddTotals()
```

Example 2:

This example shows how to determine the average using the data function property and the totals collection:

```
ActiveDocument.Sections["SalesPivot"].TopLabels["Product Line"].Totals[1].
DataFunction=bqDataFunctionAverage
ActiveDocument.Sections["SalesPivot"].TopLabels["Product Line"].Totals.Add()
```

Methods:

Remove

Properties:

Read-write: Property DataFunction as BqDataFunction

PivotLabelTotals (Collection)

Member of:

PivotLabelTotal object

Description:

Consists of pivot label total objects.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([])

to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example 1:

This example shows how to create a Totals (Collection) under SideLabel and TopLabel instances:

```
ActiveDocument.Sections["Pivot"].TopLabels["Product Name"].Totals
ActiveDocument.Sections["Pivot"].SideLabels["Region"].Totals
```

Example 2:

This example shows how to create an Add() method under each Totals (Collection). The Add() method always creates the SideLabel Totals row or TopLabel Totals column using the default totals data function. The default totals data function is dependent on the Pivot data function setting. To change the totals data function, add a SideLabel Totals row or a TopLabel Totals column and apply the DataFunction property as described in the DataFunction (Property).

```
ActiveDocument.Sections["Pivot"].SideLabels["Quarter"].Totals.Add()
ActiveDocument.Sections["Pivot"].TopLabels["Region"].Totals.Add()
```

Example 3:

This example shows how to create an Item() method under each Totals (Collection):

```
ActiveDocument.Sections["Pivot"].SideLabels["Quarter"].Totals.Item()
ActiveDocument.Sections["Pivot"].TopLabels["Region"].Totals.Item()
```

Example 4:

This example shows you to remove all totals under each Totals (Collection) using the RemoveAll() method:

```
ActiveDocument.Sections["Pivot"].SideLabels["Quarter"].Totals.RemoveAll()
ActiveDocument.Sections["Pivot"].TopLabels["Region"].Totals.RemoveAll()
```

Example 5:

This example shows how to create a Count property under each Totals (Collection):

```
ActiveDocument.Sections["Pivot"].SideLabels["Quarter"].Totals.Count
ActiveDocument.Sections["Pivot"].TopLabels["Region"].Totals.Count
```

Example 6:

This example shows how to create a DataFunction property under each instance of the Totals (Collection) that can be assigned the value from the constant group BqDataFunction. The DataFunctions property has the same effect as clicking the ALT key, clicking the total side or top label cell, and selecting a data function from the right-click or Pivot menu. The pivot data functions are as follows: sum, average, count, maximum, minimum, %column, %row, %grand, increase, %increase, non-null average, null count, and non-null count:

```
ActiveDocument.Sections["Pivot"].SideLabels["Quarter"].Totals[1].DataFunction =
BqDataFunction
ActiveDocument.Sections["Pivot"].TopLabels["Region"].Totals[1].DataFunction =
BqDataFunction
```

Methods:

Add(), Item(Value As NameOdIndex) RemoveAll()

Properties:

Read-only: Property Count As Number

Constants:

The PivotLabelTotals collection uses the BqDataFunction constant group. which consists of these values:

- bqDataFunctionAverage
- bqDataFunctionCount·
- bqDataFunctionCountDistinct
- bqDataFunctionIncrease
- bqDataFunctionMaximum
- bqDataFunctionMinimum
- bqDataFunctionNone
- bqDataFunctionNonNullAverage
- bqDataFunctionNonNullCount
- bqDataFunctionNullCount
- bqDataFunctionPercentOfColumn
- bqDataFunctionPercentOfIncrease
- bqDataFunctionPercentOfRow
- bqDataFunctionSum

PivotSection (Object)

Member of:

Sections collection

Description:

Represents a Pivot section.

Example:

Example:

This example shows how to modify the top and side labels on a pivot and create a chart based on the new pivot:

```
with(ActiveDocument.Sections["Pivot"])
{
          TopLabels.Add("Year")
          SideLabels.Add("Winery")
          ChartThisPivot()
}
```

Methods:

Activate(), ChartThisPivot() As ChartSection, Copy(), Duplicate(), Export(Filename As String, [optional] FileFormat As BqExportFileFormat, [optional] IncludeHeaders As Boolean, [optional] Prompt As Boolean), ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean, PrintOut([optional] FromPage As Number, [optional] ToPage As Number, [optional] Copies As Number, [optional] Filename As String, [optional] Prompt As Boolean), Recalculate(), RefreshDataNow(), Remove()

Properties:

Read-only: Property Active As Boolean, Property LastPrinted as Date, Property Type As BqSectionType

Read-write: Property CSSExport as Boolean, Property EnableNullFactsInComputedItems as Boolean, Property ExportWithoutQuotes as Boolean, Property HTMLExportBreakColumnCount as Number, Property HTMLExportBreakRowCount as Number, Property HTMLHorizontalPageBreakEnabled as Boolean, Property HTMLHorizontalPageBreakUnits as BqHTMLPageBreakUnits, Property HTMLVerticalPageBreakEnabled as Boolean, HTMLVerticalPageBreakUnits as BqHTMLPageBreakUnits, Property Name As String, Property RefreshData as BqRefreshData, Property ShowOutliner As Boolean, Property ShowSortLine as BooleanProperty SurfaceValues As Boolean, Property TrueComputedItemTotals as Boolean, Property Visible As Boolean

Collections:

Events as Events, Facts As PivotFacts, SideLabels As PivotLabels, TopLabels As PivotLabels, UserValues as UserValues

Objects:

DataLabels As DataLabels, CornerLabels As CornerLabels,

Placement (Object)

Member of:

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

Description:

All shapes and controls on a dashboard have a node called Placement, which contains properties and a method to enable the shape or control to be moved or resized. This node does not include moving shapes from one dashboard section to another. All offsets and sizes are given in pixels.

Example:

This example creates a rectangular shape as a goal, called LeftGoal, sets the placement properties, and colors the rectangle black.

```
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
```

Methods:

Modify(XOffset as number, YOffset as number, Width as number, Height as number)

Properties:

Read-write: Height as number, Width as number, XOffset as number, YOffset as number

PlugInClient (Object)

Member of:

Events collections

Description:

Enables the execution of an Interactive Reporting document event in the 8.0 Interactive Reporting Web Client environment. This object's properties take a Boolean value, which by default is true.

Example:

This example shows how to disable all Interactive Reporting document events when running in the Interactive Reporting Web Client environment:

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

Properties:

Read-write: ExecuteOnPostProcess as Boolean, ExecuteOnPreProcess as Boolean, ExecuteOnShutDown as Boolean, ExecuteOnStartUp as Boolean.

Read-only: Name as String

Example:

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

QueryLabel (Collection)

Member of:

(CubeQuery) QuerySection object

Description:

Represents either the dimensions on the top (columns), or on the side (rows) of the data layout in a CubeQuery. Each dimension is a single "label". The members of a dimension that are visible (that have been drilled into) are not "labels". The text shown in the Description area refers to the axis — column or row of the data layout. If the Add (Method) is used with this object, the selected dimension is added to the axis, as the innermost label. If the Item (Method) is used with this object, a label is returned at the given index, or with the given name, if a dimension of that name is in the data layout. If the index is given as an integer, its value must be between 1 and the value returned by the Count property. If the index is not within this range, an ItemNotFound exception is thrown. If the index is given as a string, it must match one of the names of the dimensions on the axis represented by this instance of the QueryLabels interface.

Example:

This example shows how to add dimensions to the a column and row panes of the data layout:

```
/ Add the "Connecticut" member to the Market label (which is a top, or column
// label)
Sections["Query"].Columns["Market"].FindAndAdd("Connecticut");

// Add the "Year" dimension as a side (row) label
Sections["Query"].Rows.Add("Year");
```

Methods:

Add(Dimension name as string), Item value as Name or String, RemoveAll()

Property:

Read only: Count as Number

QueryLabel (Object)

Member of:

QueryLabel collection

Description:

Represents a dimension in an instance of a QueryLabel collection. Each dimension label represents a dimension, and not a member of the dimension. If a CubeQuery has the Year dimension on the side and the Region dimension across the topic, it has only one entry in Rows (Year) and one entry in Columns (Region).

Select Method and Property Descriptions:

Where fully qualified member names are required, the format must be “Lev1.Lev2…”, eg “Qtr1.Jan”, or “East.New York”. Notice the dimension name is not required (it is already known from the fact the label represents a dimension). Fully qualified member names must include all levels of the member hierarchy.

When the Variable (Property) is read, the QueryLabel object returns if this label is variable or not. Variable labels display the Member Selection dialog when the query is processed. When set to “true”, this label is marked as variable.

The Add (Method) adds a specified member to the label. The MemberLocation argument must be the fully qualified member name, with each level separated by a period (for example, Year.Qtr1.Jan).

If an alias table is in use, the member must be specified using the aliases, for example, "Year.Quarter1.January", not "Year.Qtr1.Jan".

Wildcards may not be used. The FindAndAdd (Method) can be used to add members using wildcards.

The SelectorType argument specifies what type of MemberSelector is created and equates to the right-click menu in the Member Selection dialog. The bqOlapSelector enumeration contains values for Member, Children, Descendants, Bottom, Siblings, Same Level, and Same Generation. The IncludeSelf argument means the member given in the Member argument is included as well as any members generated by the SelectorType. For example, if "Qtr1" is added on the Year label with a SelectorType of bqOlapSelectorChildren, and if if IncludeSelf is true four members will be added (Qtr1, Jan, Feb, Mar) but only three members are added if IncludeSelf is false (Jan, Feb, Mar).

The IncludeSelf argument only affects the result when SelectorType is bqOlapSelectorChildren, bqOlapSelectorDescendants, or bqOlapSelectorBottom. An ItemNotFound exception will be thrown if the member does not exist at the location specified.

Note: The Add (Method) does not support the Select Next/Previous, Subset, and Dynamic Time Series selectors.

Example 1:

This example shows how to add different types of members to the "Year" dimension:

```
// Add the Qtr1.Jan member, left off IncludeSelf argument because it
// is ignored for this selector type
Sections["Query"].Rows["Year"].Add("Qtr1.Jan", bqOlapSelectorMember);

// Add Jan, Feb, Mar members
Sections["Query"].Rows["Year"].Add("Qtr1", bqOlapSelectorChildren, false);

// Add Qtr1, Jan, Feb, Mar members
Sections["Query"].Rows["Year"].Add("Qtr1", bqOlapSelectorChildren, true);

Sections["Query"].Rows["Year"].Add("Year", bqOlapSelectorBottom, true);
```

Example 2:

This example shows how to find and add member 100 – 200 to the member selector:

```
// Add the first instance of member 100-20 to the Product label.
Sections["Query"].Columns["Product"].FindAndAdd("100-20");

// Add all instances of member 100-20 to the Product label.
Sections["Query"].Columns["Product"].FindAndAdd("100-20", true);

// Add all instances of members starting with 100- to the Product label.
Sections["Query"].Columns["Product"].FindAndAdd("100-*", true);

// Assuming an alias table is in use, add all instances of members
// starting with "Cola" to the Product label.
Sections["Query"].Columns["Product"].FindAndAdd("Cola*", true);
```

Methods:

Add(MemberLocation as String, SelectorType as BqOLAPSelector, IncludeSelf as Boolean), AutoSizeHeight(), AutoSizeWidth(), Find(MemberName as String, IncludeSharedMembers as Boolean), FindAndAdd(MemberName as String, AddAll as Boolean)

Properties:

Read only: Name as String,

Read-write Variable as Boolean

QueryOptions (Object)

Member of:

QuerySection object

Description:

Returns an object enabling global, display and drill options in the Query Options dialog specific to Oracle Essbase. This object is a subtype to the Query Options dialog bo.

Example:

This example shows how to enable the use of an alias table.

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

Properties:

Read-Write: AliasTable as String, AutoRefreshQuery as Boolean, CatalogDisplayMembers as Number, DefaultDrillOperation as BqOLAPDrill, IncludeSelectedMember as Boolean, IncludeWithinSelectedGroup as Boolean, RemoveUnselectedGroup as Boolean, ReplaceMissing as String, ReplaceZeros as String, SuppressMissingColumns as Boolean, SuppressMissingRows as Boolean, SuppressSharedMembers as Boolean, SuppressZeroColumns as Boolean, SuppressZeroRows as Boolean, UseAliasTable as Boolean

QuerySection (Object)

Member of:

Sections collection

Description:

Represents a Query section.

Example 1:

This example shows how to build a Data Model using the Table Catalog object. It assumes that you are already connected to a database:

```
with (ActiveDocument.Sections["Query"].DataModel)
{
           Topics.RemoveAll()
           AutoJoin = false
//Create two new topics from tables in Table Catalog
           Catalog.Refresh()
           Table1 =Catalog.CatalogItems["WINE"]
           Table2 =Catalog.CatalogItems["WINE_SALES"]
           Topics.Add(Table1)
           Topics.Add(Table2)
           Field1 = Topics[1].TopicItems["Wine Id"]
           Field2 = Topics[2].TopicItems["Wine Id"]
//Create a new join by joining two TopicItems together
           Joins.Add(Field1,Field2,bqJoinSimpleEqual)
// Now add topic items to the request line
           for (I = 1; I <= Topics[1].TopicItems.Count; I++)
```

```
                        ActiveDocument.Sections["Query"].Requests.Add(
                        Topics[1].Name, Topics[1].TopicItems[I].DisplayName)
}
```

Example 2:

This example shows how to connect to an existing connection, remove all the limits, and process a query:

```
MyQuery = ActiveDocument.Sections["Query"]
MyQuery.DataModel.Connection.Username = "hyperion"
MyQuery.DataModel.Connection.SetPassword("hyperion")
MyQuery.DataModel.Connection.Connect()
MyQuery.Limits.RemoveAll()
MyQuery.Process()
RowReturned = ActiveDocument.Sections["Results"].RowReturned
Console.Writeln("Returned "+ RowReturned+" Rows!")
```

Methods:

Activate(), Copy(), CustomSQLFrom(CustomSQLStr as String), CustomSQLWhere(CustomSQLStr as String), Duplicate(), Export([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] Prompt as Boolean}, ExportToStream([optional] String Filename, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, Prompt as Boolean, [optional] Enclding as BqEncoding, ImportSQLFile(Filename as String, numColumns as Number), PrintOut([optional] From Page as Number, [optional] ToPage as Number, [optional]Copies as Number, [optional] Filename as String, [optional] Prompt as Boolean), Process(), ProcessStoredProc(), ProcessToTable(Tablename as String, ProcessType as BqProcessType, [optional] Grantee as String), Recalculate,() Remove(), ResetCustomSQL(), SetStoredProcParam(Parameter as Value, [optional] ParamIndex as Number)

Properties:

Read-only: Property Active As Boolean, Property LastPrinted as Date, Property LastSQLStatement as String, Property QuerySize as Number, Property Sorts As Sorts, Property Type As BqSectionType

Read-write: Property AutoProcess As Boolean, Property IncludeInProcessAll as Boolean, Property Name As String, PropertySequenceNum as Number, Property RowLimit as Number, Property RowLimitActive as Boolean, Property SaveResults As Boolean, Property ShowOutliner as Boolean, Property ShowSortLine as Boolean, Property TimeLimit As Number, Property TimeLimitActive as Boolean, Property UniqueRows As Boolean, Property Visible As Boolean

Objects:

DataModel As DataModel

Collections:

UserValue as UserValues, Limits As Limits, AggregateLimits As AggregateLimits, Items, Requests As Requests, SortItems as SortItems, AppendQueries As AppendQueries

Query Limit (Object)

Member of:

Fields collection

Description:

Sets a Query Limit field definition.

Example:

This example shows how to change the font color of the query limit in the Report section to red:

```
ActiveDocument.Sections["Report2"].Body.Fields["Query Limit"].Font.Color=bqRed
```

Methods:

Layer(BqLayer value), Spring(String Name), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

Query SQL (Object)

Member of:

Fields collection

Description:

Sets the last SQL (Structured Query Language) sent to the database when the Process button (in Query) was used.

Example:

This example shows how to identify the Query SQL field in an Alert box:

```
Alert(ActiveDocument.Sections["Report"].Body.Fields["Query SQL"].Name)
```

Methods:

Layer(Value as BqLayer), Spring (Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

(Live) RadarChart (Object)

Member of:

Dashboard objects

Description:

Represents a (Live) RadarChart object and its properties.

The following RadarChart object properties use constants in their parameters:

- Effect—BqLiveChartEffect
- Type—BqShapeType

Example:

This example shows how to set common (Live) RadarChart object attributes:

```
RadarChart.Comments = "Market Share"
RadarChart.Effect = bqLiveChartEffect3D
RadarChart.Locked = false
RadarChart.Name = "Sales"
RadarChart.SourceSectionName = "Results"
RadarChart.Visible = true
RadarChart.Facts.Add("Amount Sales")
RadarChart.CategoryAxis.Labels.Visible
RadarChart.CategoryAxis.Labels.Font.Color = 16711680
RadarChart.Marker.Style = bqMarkerStyleDiamond
```

Methods:

None

Read-write:: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Effect as BqLiveChartEffect, Locked as Boolean, Name as String, SourceSectionName as String, Visible as Boolean

Read only:: Type as BqShapeType

Objects and Collections:

UserValues collection, Placement object, Data object, Subcomponents object

RecentFiles (Collection)

Member of:

Application object

Description:

Collection of strings, that represent the list of open Interactive Reporting documents.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example prints lists of recent files to the Console window:

```
for (j = 1; j <= RecentFiles.Count ; j++)
Console.Writeln( "File #"+j+" ="+ RecentFiles[j])
```

Methods:

Function Item(Index As Number)

Properties:

Read-only: Property Count As Number

ReferenceLine (Object)

Member of:

ReferenceLines collection

Description:

Represents an individual Chart section reference line. Reference lines are horizontal or vertical lines drawn in the Chart diagram to indicate user-defined static values.

Note the behavior of a reference line:

- The reference line created for an item on the X fact axis is drawn vertically. It is drawn horizontally for an Y fact axis
- For Bubble charts, both the Axis non-fact value or fact based value can be placed on the Y-axis or X-axis since the line is associated with the bubble radius.
- A Pie chart cannot have any type of reference line.

- The Stacked Bar and Stacked Area charts both allow axis non fact and fact based value reference lines.
- The fact based reference line is always associated with some Fact column.
- When the column is removed from the Chart data layout, the corresponding reference line is also removed. When the Fact is hidden or focused, , the reference line is also hidden, or focused.

Exanple:

This example show how to return and write the label text and value associated with a reference line to the Console:

```
Console.Write(ActiveDocument.Sections["Chart"].ReferenceLines[1].GetLabelText())
```

Methods:

GetLabelText([optional] Number z), GetValue([optional] Number z), Remove(), SetValue(Value as Number)

Properties:

Read write: Axis as BqChartAxisType, DrawingOrder as BqDrawingOrder, FactIndex as Number, Function as BqDataFunction, LabelFormat as String, LegendFormat as String, NumberFormat as String, ShowInLegend as Boolean, ShowLabel as Boolean

ReferenceLines (Collection)

Member of:

ChartSection object

Description:

Represents a collection of Chart section reference line objects. Reference lines are horizontal or vertical lines drawn in the diagram to indicate user-defined static values.

Example:

This example shows how to add a reference line to Amount Sales fact value:

```
ActiveDocument.Sections["Chart"].ReferenceLines.Add("Amount")

ActiveDocument.Sections["Chart8"].ReferenceLines.Add("Amount_Sales")
```

Methods:

Add(FactNameOrIndex as value, [optional]BqChartAxisTypeAxis, Item(Index as Number, RemoveIndex as Number), RemoveAll()

Properties:

Read only: Property Count as Number

Objects:

ReferenceLine object

ReportCharts (Collection)

Member of:

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

Description:

Represents all *smart* Chart objects in the Report section.

When you insert a Chart object in the report section, a proportional copy is placed in every band instance. Any chart dropped into a header/footer that is owned by data is focused by that piece of data. Smart Charts are smart because only the corresponding section of the embedded report appears in each band instance.

Example:

This example shows how to count and display in an Alert box the number of smart Chart reports.

```
Alert(ActiveDocument.Sections["Report"].Body.Charts.Count)
```

Methods:

Item(NameOrIndex as Name)

Properties:

Read-only: Count as Number

Objects:

ReportChart object

ReportChart (Object)

Member of:

ReportChart collection

Description:

Represents a Chart object in the Charts collection of the Report section.

This object corresponds to inserting a Smart Chart in the Report section. When you insert a Chart object in the Report section, a proportional copy is placed in every band instance. Any Chart dropped into a header/footer that is owned by data is focused by that piece of data. Smart Charts are smart because only the corresponding section of the embedded report appears in each band instance.

Example:

This example shows how to spring a Chart report and a Pivot report in the body band of the report:

```
ActiveDocument.Sections["Report"].Body.Charts["Chart"].Spring("Pivot")
```

Methods:

Layer(Value as BqLayer), String(Name as String), UnSpring()

Properties:

Read-only: Name as String

ReportFooter (Object)

Member of:

ReportSection object

Description:

Represents the attributes of the report footer group. Typically, the report footer is a summarizing band of information and that is printed only on the last page of the report.

Example:

This example shows how to add a rose fill color to the report footer:

```
ActiveDocument.Sections["Report"].ReportFooter.Fill.Color = 16751052
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects and Coll.:

LineFormat object, FillFormat, Tables, Fields, Shapes, Shapes, Pivots collection, Pivot collection, Chart collection

ReportGroup (Object)

Member of:

ReportSection object

Description:

Represents the attributes of the top level from which to structure data in a report. When you drag an item from the Catalog pane into the Report Group data layout, Interactive Reporting, automatically supplies a group header band and adds a label inside the band that identifies the group. A group header categorizes data into repeating collections of records in a header band. A ReportGroup (Object) can also be added to a group footer band in addition to or instead of the group header band.

Example:

This example shows how to remove the objects in the ReportGroup:

```
ActiveDocument.Sections["Report"].Groups["Report Group1"].Remove()
```

Methods:

Move(LabelNameBefore as String), Remove()

Properties:

Read-only: Name as String

Objects/Coll.:

ReportGroup Header, ReportGroup Footer, LineFormat, FillFormat, Tables, Fields, Shapes, Pivots, Charts

ReportHeader (Object)

Member of:

ReportSection object

Description:

Represents the attributes of the report header group. Typically, the report headers is a summarizing band of information. The report header prints on the very first page of the report only.

Example:

This example shows how to instruct Interactive Reporting, not to split the report header band when a break is encountered. If Interactive Reporting, does encounter a break, the entire report header will be moved to the next page:

```
Documents["Salesreport.bqy"].Sections["Report"].ReportHeader.KeepTogether
```

Methods:

None

Properties:

Read-write: KeepTogether as Boolean, KeepWithNext as Boolean, PageBreak as BqPageBreak, Visible as Boolean

Objects:

LineFormat object, FillFormat object, Tables collection, Fields collection, Shapes collection, Pivots collection, Chart collection

ReportName Field (Object)

Member of:

Fields collection

Description:

Returns or sets the report name field.

Note: Be sure to include the Recalculate (Method) when using this object.

Example:

This example shows how to concatenate the name of the report and the current date:

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

Methods:

Layer(Value as BqLayer), Spring(Name as Name), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, Vertical Alignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType (Number)

Objects/Coll.:

UserValues, FillFormat, Font

ReportPivot (Object)

Member of:

ReportPivot collection

Description:

Represents a Pivot object in the Pivot collection of the Report section. This object corresponds to inserting a Smart Pivot report in the Report section. When you insert a Pivot object in the Report section, a proportional copy is placed in every band instance. Any Pivot dropped into a header/footer that is owned by data is focused by that piece of data. Smart Pivots are smart because only the corresponding section of the embedded report display in each band instance.

Example:

This example shows how to layer a Smart Pivot report to the front of a stack:

```
ActiveDocument.Sections["Report"].Body.Pivots["Pivot"].Layer(bqLayerFront)
```

Methods:

Layer(Value as BqLayer), Spring(Name as String), UnSpring()

Properties:

Read-only: Name as String

ReportPivots (Collection)

Member of:

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

Description:

Represents all *Smart* Pivot objects in the report section.

When you insert a Pivot object in the report section, a proportional copy is placed in every band instance. Any Pivot dropped into a header/footer that is owned by data is focused by that piece of data. Smart Pivot reports are smart because only the corresponding section of the embedded report displays in each band instance.

Example:

This example shows how to count the number of Pivot reports inserted in the Body band of the report:

```
Alert(ActiveDocument.Sections["Report"].Body.Pivots.Count)
```

Methods:

Item(NameOrIndex as Name)

Properties:

Read-only: Count as Number

Objects:

ReportPivot object

ReportSection (Object)

Member of:

Sections collection

Description:

Represents a Report section.

Example 1:

This example shows how to activate the Report section in the document:

```
ActiveDocument.Sections["Report"].Activate()
```

Example 2:

This example shows how to turn off the page headers for the first page in a report:

```
if ( PageNm==1)
{' '}
else
{"Query Processed:  "+ Format(new Date(), "d-mmm-yyyy")}
```

Example 3:

This example shows how to set the print orientation of the report:

```
ActiveDocument.Sections["Report"].Orientation = bqOrientationPortrait
ActiveDocument.Sections["Report"].Recalculate()
```

Methods:

Activate(), Copy(), Duplicate(), Export([optional] Filename As String, FileFormat As BqExportFileFormat [optional], [IncludeHeaders As Boolean]), ExportToStream([optional]Filename As String, [optional] FileFormat As BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean, [optional) Encoding as BqEncoding} , PrintOut([optional] [FromPage As Number], ToPage As Number, Copies [optional], Filename. [optional] Boolean Prompt], Recalculate(), Remove()

Properties:

Read-only: Property Active As Boolean, Property LastPrinted as Date, Property Type As BqSectionType

Read-write: Property BottomMargin, CSSExport as Boolean, Property LeftMargin, Property Name As String, Property Orientation as BqOrientation, Property RightMargin, Property ShowOutliner As Boolean, Property ShowSortLine as Boolean, Property SuspendCalculation as Boolean, Property TopMargin, Property Visible As Boolean

Object and Coll.:

UserValues, Groups collection, ReportHeader, ReportFooter, Body

ReportTable (Object)

Member of:

ReportTable collection

Description:

Represents a specific table object within a specific Report section object.

In the user interface, tables are created with dimension columns and fact columns, where the distinction is typically text versus numeric content. These tables are quite flexible structures in that several tables may be introduced into each band; originating from the same or different result sets in the document.

Example:

This example shows how to simulate the look of a green bar report by alternating the color scheme of every other row between green and white:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].BackgroundColor = bqLightGreen
ActiveDocument.Sections["Report"].Body.Tables["Table"].BackgroundAlternateColor =
bqWhite
ActiveDocument.Sections["Report"].Body.Tables["Table"].BackgroundAlternateFrequency = 1
```

Methods:

Item(NameOrIndex as Value)

Properties:

Read-only: Property Count as Number

Objects:

Dimension collection, Facts collection, SortItems collection

ReportTables (Collection)

Member of:

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

Description:

Represents all the table objects in a specific report section object.

Example:

This example uses the Count property to determine the number of tables in the Body band of the report and write it to the Console window:

```
Console.Write(ActiveDocument.Sections["Report"].Body.Tables.Count)
```

Methods:

Spring as Sting Name, UnSpring

Properties:

Read-write: Property BackgroundAlternateColor as BqColorType, BackgroundAlternateFrequency as Number, BackgroundColor as BqColor Type, BackgroundShowAlternate Color as Boolean, BorderColor as BqColorType, BorderWidth as Number

Read-only: Property Name as String

Request (Object)

Member of:

Requests collection

Description:

Represents a request line item.

Example:

This example prints the display name and data type of each item on the request line:

```
var count = ActiveDocument.Sections["Query"].Requests.Count
for(i=1;i<=count;i++)
{
       myRequest = ActiveDocument.Sections["Query"].Requests[i]
       switch(myRequest.DataType)
{
                  case 1:
                            myType = "String"
                            break;
```

```
                case 2:
                          myType = "Integer"
                          break;
                case 3:
                          myType = "Number"
                          break;
                case 4:
                          myType = "Date"
                          break;
                default:
                          myType = "Unknown"
}
Console.Write(myRequest.DisplayName +" DataType ="+myType+"\r\n")
}
```

Methods:

Remove()

Properties:

Read-only: Property SQLName As String

Read-write: Property DataType As BqDataType, Property DisplayName As String, Property SQLName as String, Property Visible As Boolean

Requests (Collection)

Member of:

QuerySection object

Description:

Represents all topic items on the request line.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to remove all request line items and add items based on topics queries:

```
with(ActiveDocument.Sections["Query"])
            {
            Requests.RemoveAll()
            for (I = 1; I <= DataModel.Topics[1].TopicItems.Count; I++)
                        {
                        TopicName = Topics[1].Name
```

```
                    TopicItemName = Topics[1].TopicItems[I].DisplayName
                    Requests.Add(TopicName,TopicItemName)
                }
        }
```

Methods:

Add(TopicName As String, TopicItemName As String) As Request, AddComputedItem(Name As String, Expression As String, Type As BqDataType) As Request, Item(NameOrIndex) As Request, RemoveAll()

Properties:

Read-only: Property Count As Number

ResourceManager (Object)

Member of:

Document object

Description:

Represents the Resource Manager utility used to load, manage and share pictures in theInteractive Reportingdocument file. By storing one copy of the image and referencing the copy as a resource when used elsewhere, the Resource Manager reduces the size of an Interactive Reporting document file, which in turn reduces the memory footprint required to open some documents. Each Interactive Reporting document file has its own Resource Manager. A new picture is added to the Resource Manager from disk by using an Import feature. In pre 9.3 release Interactive Reporting document file, Resource Manager images are not backward compatible when images are: When an Interactive Reporting document (BQY) is created and saved using 9.3.x releases, all images are saved in a centralized format (the Resource Manager). As a result, the images cannot be read or displayed in prior releases of Interactive Reporting. This is because the relocation and rationalization of the images are not understood by releases prior to 9.3.x. The exception is images from an Interactive Reportingdocument file saved in a pre 9.3 Release, which have been referenced once: not copied, merged or duplicated, that is, placed in Dashboard or Report workspace as a Picture graphic in traditional fashion. To retain compatibility with earlier releases, you can run the RevertImageResources script. This script provides a facility to undo the relocation and image merging, and thereby return the Interactive Reporting document file to the pre-9.3 pre-Resource Manager format. This script is included with the 9.3. Release of Dashboard Development Services. It can also be downloaded from the Hyperion Developer Network. This script can only be used on the desktop, and not in Workspace, as it relies on the COM feature of Interactive Reporting Studio. Certain new features make documents ineligible for conversion to the non-Resource Manager format: for example if the 9.3Interactive Reporting document file has Bubble and Scatter charts, then these are lost when theInteractive Reporting document file is opened in the older version of. For information on running the RevertImageResources script, see the Hyperion System 9 BI+ 9.3. Readme.

Example:

This

Objects and Collections:

Images collection

Result Limit (Object)

Member of:

Fields collection

Description:

Sets a Results limit field definition.

Example:

This example shows how to add a second Results section limit field to an existing Results section limit field programmatically:

```
ActiveDocument.Sections["Report"].Body.Fields["Result Limit"].Formula=" \"State\" +
LocalLimitValues(\"Results\", \"State Province\",\"\",\",\")  + \"  \"+   \"City \" +
LocalLimitValues(\"Results\", \"City\",\"\",\",\")"
```

Methods:

Layer(Value as BqLayer), Spring(Name As String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment, Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat object, FillFormat object, FontFormat object

Results (Collection)

Member of:

DMCatalog object

Description:

Represents all local results sets in Table Catalogs.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to get a count of the LocalResults in the Table Catalog.

```
ResultSetCount=ActiveDocument.Sections["Query2"].DataModel.Catalog.Results.Count
```

Methods:

Item(NameOrIndex) As DMResult

Properties:

Read-only: Property Count As Number

Results (Object)

Member of:

Results collection

Description:

Represents a results set in a Table Catalog.

Note: The Add method for CubeQuery "download to results" dependent reporting sections (Chart, Pivot, Table, Report) does not accept the CubeQuery "Query" section name as the dependent section argument. This is different from use of the Add method with a relational Query section. With CubeQuery, you must use the downloaded "Results" section as the dependent section argument.

Example:

This example shows how to add local results set to Data Models:

```
//Results Object
ResultSetName=ActiveDocument.Sections["Query2"].DataModel.Catalog.Results["Results2"].Name
```

Methods:

Activate(), Copy(), Duplicate, Export([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] Prompt as Boolean, ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as

Boolean), GetCell(nRow as Number, nCol as Number), PrintOut([optional] FromPage as Number, [optional] ToPage as Number, [optional] Copies as Number, [optional] Filename as String, [optional] Prompt as Boolean), Recalculate{}, Remove{}

Properties:

Read-only: Property Active as Boolean, Property RowCount as Number, Property Type as BqSectionType

Read-write: Property BackgroundAlternateColor as BQColorType, Property BackgroundAlternateFrequency as Number, Property BackgroundColor as BqColorType, BackgroundShowAlternateColor as Boolean, Property BorderColor as BqColorType, Property BorderWidth as Number, Property CSSExport as Boolean, Property ExportWithoutQuotes as Boolean, Property HTMLExportBreakRowCount as Number, Property, Property HTMLVerticalPageBreakEnabled as Boolean, Property HTMLVerticalPageBreakUnits as BqHTMLPageBreakUnits, Property LastPrinted, Property Name as String, Property OfficeHTMLFormulasEnabled as Boolean, Property ShowOutliner as Boolean, Property ShowRowNumbers as Boolean, Property ShowSortLine as Boolean, Property Visible as Boolean

Collections:

UserValues as UserValuesColumns As Columns, Limits as Limits, SortItems as SortItems

Objects:

ParentSection

RightAxis (Object)

Member of:

ValuesAxis object

Description:

Represents all the right values axis properties in a chart.

Example:

This example shows how to set basic properties for the right axis:

```
with(ActiveDocument.Sections["Chart"].ValuesAxis)
{
RightAxis.AutoScale = true
RightAxis.ShowLabel = false
RightAxis.LabelText = "Right Axis"
}
```

Methods:

None

Properties:

Read-write: Property AutoScale As Boolean, Property LabelText As String, Property ScaleMax As Number, Property ScaleMin As Number, Property ShowLabel As Boolean

ScatterChart (Object)

Member of:

ChartSection object

Description:

Represents a scatter chart, which is useful for emphasizing scientific or statistical similarities rather than differences in your analysis. Scatter charts illustrate the relationship between pairs of numerical or quantitative values, which are combined into individual data points along the horizontal (y axis) and a vertical (x axis) axis. Data points are plotted in uneven intervals.

Note: When the Chart type is set to Scatter, only items appropriate for Scatter chart display in the Script Editor Object Model browser.

Scheduler (Object)

Member of:

Events collections

Description:

Enables the execution of a Scheduler document event in the 8.0 client environment.

Note: The ExecuteOnStartup (Property), ExecuteOnShutDown (Property), ExecuteOnPreProcess (Property), and ExecuteOnPostProcess (Property) under the Scheduler object must have a default setting of "False" if the document was created before version 8.0.

Example:

This example shows how to disable all document events when running in the Scheduler environment:

```
Documents["Sample1.bqy"].Events["Scheduler"].ExecuteOnPostProcess=false
Documents["Sample1.bqy"].Events["Scheduler"].ExecuteOnPreProcess=false
Documents["Sample1.bqy"].Events["Screduler"].ExecuteOnShutDown=false
Documents["Sample1.bqy"].Events["scheduler"].ExecuteOnStartUp=false
```

Properties:

Read-write: ExecuteOnPostProcess as Boolean, ExecuteOnPreProcess as Boolean, ExecuteOnShutDown as Boolean, ExecuteOnStartUp as Boolean.

Read-only: Name as String

Section (Object)

Member of:

Sections collection

Description:

Represents the base object from which all section objects are derived. It is used to access object in a section other than the one containing the script.

Example:

This example shows how to activate the *Sales Chart* section, activate the legend, use *International Sales Report* as title, make the chart type horizontal bar chart, and export the chart to intlchrt.htm.

```
myChart = ActiveDocument.Sections["Sales Chart"]
myChart.Activate()
myChart.ShowLegend = true
myChart.Title  = "International Sales Report"
myChart.ChartType = bqChartTypeHorizontalBar
```

Methods:

Activate(), Copy(), CustomSQLFrom(String CustomSQLStr), CustomSQLWhere(String CustomSQLStr), Duplicate(), Export(Filename As String, FileFormat As BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] Prompt as Boolean), ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean), ImportSQLFile(Filename as String, numColumns as Number), PrintOut([optional] FromPage as Number, [optional] ToPage as Number, [optional] Copies as Number, [optional] Filename as String, [optional Prompt as Boolean), Process(), ProcessStoredProc(), ProcessToTable[Tablename as String, ProcessType as BqProcessType, [optional] Grantee as string, Recalculate(), Remove(), ResetCustomSQL{}, SetStoredProcParam(Value parameter, [optional] ParamIndex as Number)

Properties:

Read-only: Property Active As Boolean, Property LastSQLStatement as String, Process QuerySize as Number, Property Type As BqSectionType

Read-write: Property AutoProcess as Boolean, Property IncludeInProcessAll, Property Index As Number, Property Name As String, Property ProcessSequenceNum as Number, Property RowLimit as Number, Property Property RowLimitAcitve as Boolean, Property SaveResults as

Boolean, Property ShowOutliner as Boolean, Property ShowSortLine as Boolean, Property TimeLimit as Number, Property TimeLimitActive as Boolean, Property UniqueRows as Boolean, Property Visible As Boolean

Constants:

The FileFormat method of the Section (Object) uses the BqExportFileFormat constant group. The BqExportFileFormat constant group consists of these values:·

- bqExportFormatCSV
- bqExportFormatExcel2
- bqExportFormatExcel5
- bqExportFormatHTML
- bqExportFormatJPEG
- bqExportFormatLotus123
- bqExportFormatPDF
- bqExportFormatText
- bqExportFormatOfficeHTML
- bqExportFormatOfficeMHTML

Sections (Object)

Member of:

Toolbars collection

Description:

Represents the Sections toolbar in the application. The Sections toolbar is used to provide commands that are specific to each Interactive Reporting document section.

If the Sections (Object) has any of its associated properties accessed from EPM Workspace, the script commands are ignored, no exceptions are thrown, and any scripts continue.

Constants:

The Type property of the Toolbar (Object) uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows you enable the Sections toolbar in the application:

```
//Syntax for turning on the Sections toolbar
Toolbars["Sections"].Visible=true;
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

Sections (Collection)

Member of:

Document object

Description:

Represents all sections in single documents.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Note: When you use the Add (method) to add a new section, note the following. It is recommended that an Interactive Reporting document file (.bqy) have no more two hundred sections to ensure smooth performance.

Example:

This example shows how to create report and query sections.

In report sections (Chart and Pivot), you must set the*Section Dependency* parameter. You must associate Charts and Pivots with a Query or Results set.

```
MySection = ActiveDocument.Sections.Add(bqChart,"Query")
or
MySection = ActiveDocument.Sections.Add(bqPivot,"Results")
MySection.Name = "New Chart"
//Adding Queries does not require a section dependence
MySection = ActiveDocument.Sections.Add(bqQuery)
```

Methods:

Add(SectionType As BqSectionType, [SectionDependency as String]) As Section, ImportDataFile(FileName As String, Format As BqImportDataFileFormat), Item(Value NameOrIndex) As Section

Properties:

Read-only: Property Count As Number, Property QueryCount as Number

Objects:

Query object, Results object, Pivot object, Chart object, and Dashboard object

SelectedList (Collection)

Member of:

ControlsListBox object

Description:

Represents all selected list box items.

Tip: All collections have a "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items from a particular index or with a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how count the number of selected items in a list box.

```
Alert(ListBox1.SelectedList.Count)
```

Methods:

String Item(Number nIndex), Number ItemIndex(Number nIndex)

Properties:

Read-only: Property Count As Number

Objects:

SelectedList object

SelectedList (Object)

Member of:

ListBoxControl object

Description:

Represents a selected items within a list box.

Example:

This example shows how to add the selected items from a list box control to a preexisting results limit:

```
ActiveDocument.Sections["Results"].Limits[1].SelectedValues.RemoveAll()
for(I = 1; I <= ListBox.SelectedList.Count;I++)
{
          NewLimitValue = ListBox.SelectedList[I]
          ActiveDocument.Sections["Results"].Limits[1].SelectedValues.Add(NewLimitValue)
}
```

Methods:

Item(Index As Number) As String, ItemIndex(Index As Number) As Number

Properties:

Read-only: Property Count As Number

SelectedValues (Collection)

Member of:

LimitValues object

Description:

Represents selected limit values.

Note: Because values are called from database, the Add (Method) is unavailable for the AvailableValues (Collection). For ""CustomValues (Collection)" on page 60, the Add (Method) adds a value to the list. For the "SelectedValues (Collection)" on page 198, the Add (Method) adds a value to the selected list. The AddAll (Method) of the SelectedValues (Collection) selects all values of AvailableValues or CustomValues (Collection).

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example shows how to take every value from the AvailableValues (Collection) and add them to the SelectedValues (Collection). This is essentially the same as performing a select all values and transferring the selection in the Limit User Interface:

```
LimitCount = ActiveDocument.Sections["Results"].Limits[1].AvailableValues.Countfor
(i=1;i<=LimitCount;I++){MyVal =
Add(ActiveDocument.Sections["Results"].Limits[1].AvailableValues[i]ActiveDocument.Sectio
ns["Results"].Limits[1].SelectedValues.Add(MyVal))}
```

Methods:

Add(Value ValueItem), AddAll(), Item(Number Index), RemoveAll()

Properties:

Read-only: Property Count as Number,

Session (Object)

Member of:

Application object

Description:

Refers to the current Web browser's session variables. The Session (Object) contains three collections, which logically group a browser's different type of data variables. The Session (Object) applies to the Web plug-ins but is visible in the client server product to support script testing. To activate the Session object, you must include the key value pair JScript=enable in the URL. Please refer to "URL (Object)" on page 232 and "Form (Object)" on page 94 for more information.

Do not use this object model syntax in Interactive Reporting documents to be deployed in EPM Workspace:

- Session Active
- Session.Form.Add()
- Session.Form.Item()
- Session.URL.Add()
- Session.URL.Item()
- Session.Cookies.Add()
- Session.Cookies.Item()

Note: If a form launches the Interactive Reporting Web Client with the "withnav" variant of the Smartcut, the Session.Form collection items is inaccessible. In this instance, use withnav_get or withnav_run. This behavior can be set as the default mode for a Smartcut by editing the ws.conf line to show: WebClient.UserInterface.Navigation.ShowNavigationBar=true

Example:

This script shows determines if a session is active and process the session variables:

```
//Session.Active = true if the script is running in the plug-in and JScript=enable
if (Session.Active)
        Alert("Your web username is ="+ Session.Cookies["HYPERIONUSER"], "Web Username")
else
         Alert("You are not running a plug-in or you have not added the JScript=enable
key value pair to your URL")
```

Methods:

None

Properties:

Read-only: Property Active As Boolean

Collections

Form as Form, Cookies as Cookies, URL as URL

Shape (Object)

Member of:

Shapes collection

Description:

Represents a Dashboard graphic item in a Shapes (Collection). Certain properties apply only to specific shape objects. For example, the Picture Effect property applies only to a picture object. If you refer to properties that do not apply to an object, no action occurs.

Example:

This example shows how to change the properties of drawing objects in a Dashboard section. The example assumes that the script is running from the same Dashboard section. This provides access to the drawing objects by name:

```
Line1.DashStyle=bqDashStyleDotDash
Line1.LineWidth = 3
//note you may use Hex values instead of enumerated types for any color property
Rect1.BorderColor = bqBlue
```

```
Rect1.Line.DashStyle=bqDashStyleDotDash
TextLabel.Text = "Welcome to Dashboard Scripting"
TextLabel.Font.Style = bqFontStyleBoldItalic
```

Methods:

OnClick()

Properties:

Read-only: Property Active As Boolean, Property Group As String, Property Name As String, Property RowCount As Number, Property RowNumber As Number, Property Type As BqShapeType

Read-write: AccessibilityText (Property), Alignment As BqHorizontalAlignment, AppendObjectText (Property), Property Text As String, Property VerticalAlignment As BqVerticalAlignment, Property Visible As Boolean

Objects

Fill As Fill, Line As Line, EventScripts collection

Shapes (Collection)

Member of:

DashboardSection object

Description:

Represents all graphic objects in a Dashboard tab.

Objects included in the Shapes (Collection) are:

- CommandButton
- RadarChart
- PieChart
- LineChart
- FunnelChart
- BlockChart
- BarChart
- TrafficLight
- Thermometer
- Bullet
- Speedometer
- Slider

- HyperLink
- EmbeddedBrowser
- TextBox
- DropDown
- ListBox
- RadioButton
- CommandButton
- Picture
- TextLabel
- Oval
- RoundRect
- VtLine
- HzLine
- Line
- Rect

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example writes the number of shapes in a report section to the Console window:

```
Console.Writeln(ActiveDocument.Sections["Report"].Body.Shapes.Count)
```

Methods:

Item(Value as NameOrIndex) As Shape

Properties:

Read-only: Property Count As Number

SharedLibrary (Object)

Member of:

Application object

Description:

Represents an external, dynamically linked library.

Example:

This example shows how to call a function from a local DLL:

```
MyLibrary = Application.LoadSharedLibrary("c:\\temp\mydll.dll")
MyLibrary.Call("SetTransaction","String",Value1)
```

Methods:

Call(sFunctionName As String, sArgumentType As String, [arg1], [arg2], [arg3], [arg4], [arg5], [arg6], [arg7], [arg8])

SideLabels (Collection)

Description:

For information on the SideLabels (Collection) used in the Pivot section, see the "PivotLabels (Collection)" on page 165.

For information on the SideLabels (Collection) used in the OLAPQuery section, see the "OLAPLabels (Collection)" on page 141.

SideLabelValues (Array)

Member of:

Dashboard Pivot Embedded Section Object (ESO) in Active Mode only

Description:

Returns a JavaScript array (and not an Interactive Reporting Object Model array) of side label values . Each array element contains the value of the label at successively lower levels. For example, if there are two side labels in the Pivot, Year and Month, then the array element at index 1 in the array might contain "2001" and the next element at index 2 would contain a value like "November". Because JavaScript arrays are zero-based, the element at index 0 (zero) contains an empty string value.

The SideLabelValues (Array) is accessible for a Dashboard Embedded Section Object (ESO) only. It is available even if the Pivot contains no data. In addition, it is functional only when the Pivot is in Active mode.

The SideLabelValues (Array) is refreshed when a Pivot ESO cell is clicked in the user interface, or the OnCellDoubleClick() method used. An array is used due to the uncertainty of the SideLabel hierarchy. The array forces the user to get the value to ensure that it is current. It avoids using the standard x.y.z.name structure of an object that could become ambiguous or incorrect given changes of the Pivot side labels.

The array is read-only. Changing array values does not effect Pivot side label values.

The array returns a string which reflects the names of the Pivot side labels in sequence. Values are returned using the length accessor. JavaScript arrays are normally 0-based. However, the]0] value always returns an empty string. If there are three sideop labels, the array returns … SideLabelValues[0] =<empty string>, …SideLabelValues[1] =<First Side Label Name>, … SideLabelValues[2] =<Second Side Label Name>, …SideLabelValues[3] =<Last Side Label Name>.

If a Pivot side label does not exist, the SideLabelValues[1] returns “undefined”; however no exception is generated and scripts execute. The same behavior occurs when values are called that exceed the length of the array. The array behaves the same regardless of data at the selected intersection.

Values returned by the array are unformatted. Use scripts to apply formatting.

Iterating through the SideLabelValues (Array) does not persists (dirty) in the document or retrieve the value. The value in the array reflects the names of the Pivot side labels in sequence.

- Separately from Pivot values in the document
- Within the application

The SideLabelValues (Array) is available in Interactive Reporting Studio and the Interactive Reporting Web Client. It is unavailable in EPM Workspace.

Note: You cannot use graphics to identify selected cells using this array.

Example:

This example shows how to access top label array data:

```
var ActionName = "SideLabelValues Array"

TextBox1.Text ="Start " + ActionName

try

{

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues Default is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues length is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues.length

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues[0] is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues[0]

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues[1] is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues[1]

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues[2] is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues[2]
```

```
PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nSideLabelValues[3] is: " +
ActiveSection.Shapes["Pivot1"].SideLabelValues[3]
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}
TextBox1.Text ="End " + ActionName
```

The results of the above script are displayed as follows:

SideLabelValues Default is: ,4 SideLabelValues length is: 2 SideLabelValues[0] is: SideLabelValues[1] is: 4 SideLabelValues[2] is: undefined SideLabelValues[3] is: undefined

Related Topics

"TopLabelValues (Array)" on page 223

Slider (Object)

Member of:

Dashboard objects

Description:

Represents the slider control object that drives the indicator on an associated gauge

The current value of the Slider is determined by the relative location between the end points of the track or the minimum and maximum values defined for the slider. The Slider allows for a continuous range of values between its minimum and maximum values, but you can also set the snap interval parameter to specify intervals between the minimum and maximum value. A Slider can show tick marks, which are independent of the assigned values, at specified intervals along the track. The slider has a horizontal orientation by default, but it can be set to a vertical orientation by setting the value of the direction parameter to vertical. The slider track stretches from end to end and the tick marks are placed from left to right just above the track.

When associating a slider with a gauge, note that the slider can control multiple gauges in the same dashboard section, and there is no need to have cross dashboard controls. The Slider should use the data set from a section, but can reference any data type except for a BLOB and CLOBs. The data set a can only be imported from Result, Table, Pivot, OLAP Query and CubeQuery sections.

Example:

This example shows how to add the tooltip "East Coast Sales", unlock the slider object, use data from the Results set, set the slider to a realistic theme, and make the slider visible:

```
SalesSlider.Comments = "East Coast Sales"
SalesSlider.Locked = false
SalesSlider.SourceSectionName = "Results"
SalesSlider.Subtype = "realistic"
SalesSlider.Visible = true
```

Methods:

OnSelection

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Locked as Boolean, Name as String, SelectedIndex as Number, SourceSectionName as String, Subtype as String, Visible as Boolean

Read only:: SelectedValue as String, Type as BqShapeType

Objects:

UserValues collection, Placement object, Subcomponents object, Themes collection, Data object

SmartViewClient (Object)

Member of:

Events collections

Description:

Enables the execution of a SmartViewClient document event in the 9.0 client environment.

Note: The ExecuteOnStartup (Property), ExecuteOnShutDown (Property), ExecuteOnPreProcess (Property), and ExecuteOnPostProcess (Property) under the SmartViewClient object must have default settings of "False" if the document was created before version 8.0.

Example:

This example shows how to disable all document events when running in the Scheduler environment:

```
Documents["Sample 1.bqy"].Events["SmartViewClient"].ExecuteOnPostProcess=false
Documents["Sample 1.bqy"].Events["SmartViewClient"].ExecuteOnPreProcess=false
Documents["Sample 1.bqy"].Events["SmartViewClient"].ExecuteOnShutdown=false
Documents["Sample 1.bqy"].Events["SmartViewClient"].ExecuteOnStartup=false
```

Properties:

Read-write: ExecuteOnPostProcess as Boolean, ExecuteOnPreProcess as Boolean, ExecuteOnShutDown as Boolean, ExecuteOnStartUp as Boolean.

Read-only: Name as String

SortItems (Collection)

Member of:

QuerySection object, ResultsSection object, TableSection object

Description:

Collection of sorts within a Query, Results, or Table section.

In the Query section, sort line objects must be columns on the Request line since only these can be placed on the Sort Line. In the Results and Table section, sort line objects must be columns in the Results set.

The SortItems (Collection) provides you with the ability to create Sort Line objects (column names), add them to the Sort Line, specify a sort order, and force an immediate sort (for Results and Table).

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or using a specific name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical: `myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Constants:

The SortItems property uses the BqSortOrder constant group, which consists of these values:·

bqSortAscend

bqSortDescend

Example 1 (Query Sorts):

The example shows how to select sort objects (columns) in Query sections:

```
ActiveDocument.Sections["Query"].Sort
```

Example 2 (Query Sorts):

This following example shows how to add, modify and remove all sort line object(s) (columns) in Query sections:

```
ActiveDocument.Sections["Query"].Sort.Add()
//ActiveDocument.Sections["Query"].Sort.Add(Column_Name)
ActiveDocument.Sections["Query"].Sort.Item()
//ActiveDocument.Sections["Query"].Sort.Item(Value)
ActiveDocument.Sections["Query"].Sort.RemoveAll()
```

Example 3 (Query Sorts):

This example shows how to count sort items in the Query section:

```
ActiveDocument.Sections["Query"].Sort.Count
```

Example 4 (Query Sorts):

This example shows how to remove a *Product Id* sort from the Sort line in the Query section:

```
ActiveDocument.Sections["Query"].SortItems["Product Id"].Remove()
```

Example 5 (Query Sorts):

This example shows how to use ascending sort order in Query sections:

```
ActiveDocument.Sections["Query"].Sort["1"].SortOrder = bqSortAscend;
```

Example 6 (Results Sorts):

This example shows how to select sort objects (columns) in Results sections:

```
ActiveDocument.Sections["Results"].Sort
```

Methods:

Add(Request item as String), Item(Value as NameOrIndex), RemoveAll(), SortNow()

Properties:

Read-only: Property Count As Number

SortItems (Object)

Member of:

QuerySection object, ResultsSection object, TableSection object

Description:

Represents the individual Sort Line item used in a sort condition. In the Query section, sort line items must be columns that are on the Request line because these are the only items that can be placed on the Sort line. In the Results and Table section, sort line items must be columns in the Results set.

Constants:

The SortItem (Object) uses the SortOrder property. The SortItems property uses the BqSortOrder constant group, which consists of the bqSortAscend and bqSortDescend values.

Methods:

Remove()

Properties:

Read-Write: Property SortOrder as BqSortOrder

Speedometer (Object)

Member of:

Dashboard objects, Shapes collection

Description:

Represents an individual speedometer gauge in the Dashboard section.

Example:

This example shows how to make the speedometer gauge visible, add a comment (tooltip), and display the fact amount:

```
Speedometer.Visible = true
Speedometer.Comments = "Sales"
Alert(Speedometer.Facts["Amount"].Name)
```

Properties:

Read-write:: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Locked as Boolean, SourceSectionName as String, Subtype as String, Visible as Boolean

Read only:: Type as Number

Objects and Collections:

UserValues collection, Placement object, Subcomponents object, Themes collection, Data object

Standard (Object)

Member of:

Toolbars collection

Description:

Represents the Standard toolbar in the application. The Standard toolbar contains icons for commonly used operations, such as the *Open* and *Save* commands. The availability of an icon depends on the active section

Constants:

The Type property of the Toolbar (Object) uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows how to enable the Standard toolbar in the application:

```
//Syntax for turning on the Standard toolbar
Toolbars["Standard"].Visible=true;
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

Subcomponents (Object)

Member of:

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

Description:

Represents objects that are visible, such as tickmarks and categories, in a (Live) Chart or gauge.

Example:

This example shows how to make the legend and tickmarks available for the Bullet gauge.

```
ActiveDocument.Sections["Dashboard"].Shapes["Bullet"].Legend.Visible = true
ActiveDocument.Sections["Dashboard"].Shapes["Bullet"].TickMark.Visible = true
```

Properties:

None

Methods:

None

Objects:

CategoryAxis object, ColorRanges collection, FactsAxis object, Legend object, Marker object, NumericRange object, TickMark object, Title object, ValueLabels object

TableFact (Object)

Member of:

TableFacts collection

Description:

Sets the measurable or quantifiable fact objects that makes up the body of the report. Interactive Reporting quantifies values by group header and dimension. If you have a descriptive numeric value that should not be calculated, such as Retail Price or Target Sales, use the table dimension object instead of a fact object.

Example 1:

This example shows how to remove the *Unit Sales* object from table facts:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts["Unit Sales"].Remove()
```

Example 2:

This example shows how to align horizontal text to the left within a fact column:

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

Example 3:

This example shows how not to display the column total for the *Unit Sales* fact object:

```
ActiveDocument.Sections["Report"].Body.Tables
["Table"].Facts["Unit Sales"].ShowColumnTotal = false
```

Methods:

Move(LabelNameBefore as String), Remove()

Properties:

Read-write: BackgroundAlternateColor as BqColorType, BackgroundAlternateFrequency as Number, BackgroundColor as BqColorType, BackgroundAlternateColor as Boolean, DataFunction as BqDataFunction, HorizontalAlignment as BqHorizontalAlignment, NumberFormat as String, ShowColumnTotal as Boolean, SuppressDuplicates as Boolean, TextWrap as Boolean, VerticalAlignment

Read-only: Name as String

TableFacts (Collection)

Member of:

ReportTable object

Description:

Represents all table fact objects in the report section.

Note: If you use the Add (Method), Move (Method), and Remove (Method) with this collection, and the SuspendCalculation (Property) is true (default), use the Recalculate (Method) to force recalculate the Report section.

Example 1:

This example shows how add the *Unit Sales* column to the table:

```
ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts.Add("Unit Sales")
ActiveDocument.Sections["Report"].Recalculate()
```

Example 2:

This example shows how to use the AddComputed (Method) to divide the *Amount Sales* column by the *Unit Sales* column and display the results in a new computed column called *My Computed*:

```
var myStr ="Tables(\"Results\").Columns(\"Amount_Sales\").Sum(currBreak)  /
Tables(\"Results\").Columns(\"Unit_Sales\").Sum(currBreak)";
ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts.AddComputed("MyComputed",
myStr,bqDataTypeNumber)
ActiveDocument.Sections["Report"].Recalculate()
```

Example 3:

This example shows you to how to count the number of tables in the body of the Report section:

```
Alert(ActiveDocument.Sections["Report"].Body.Tables["Table"].Facts.Count," Number of
Tables")
ActiveDocument.Sections["Report"].Recalculate()
```

Methods:

Add(NewFact as String, [optional] MoveBeforeName as String), AddComputed(Name as String, Expression as String), Item(NameOrIndex as Value), ModifyComputed(OldName as String, NewName as String, Expression as String), RemoveAll()

Properties:

Read-only: Count as Number

Table (Object)

Member of:

Shapes (Collection)

Description:

Represents an embedded Results or Table section object within the Shapes collection .

Example:

This example shows how to print the names of all the columns to the Console window.

```
MyResults = ActiveDocument.Sections["Dashboard"].Shapes["Results"]
ColumnCount = MyResults.Columns.Count
for (I=1;I<= ColumnCount;I++)
Console.Write("Column#"+I+":"+MyResults.Columns[I].Name+"\r\n")
```

Methods:

Activate(), Copy(), Duplicate(), Export([optional] Filename As String, [optional] FileFormat As BqExportFileFormat [optional], IncludeHeaders As Boolean, [optional] Prompt As Boolean), GetCell(nRow As Number, nCol As Number) As Value, PrintOut([optional] FromPage As Number], [ToPage As Number], [Copies As Number], [Filename As String], [Prompt As Boolean]), Recalculate(), Remove()

Properties:

Read-only: Property Active As Boolean, Property Name as String, Property RowCount As Number, Property RowNumber as Number, Property Type As BqSectionType

Read write: Property Alignment as BqHorizontalAlignment, Property ScrollbarsAlwaysShown as Boolean, Property ShowOutliner as Boolean, Property ShowRowNumbers as Boolean, Property Text as String, Property VerticalAlignment as BqVerticalAlignment, Property Visible as Boolean

Collections:

Limits as Limits, Columns As Columns, SortItems as SortItems, TableSection as Object

TableSection (Object)

Member of:

Sections collection

Description:

Represents a Results or Table section object within a document.

Example:

This example shows how to print the names of all the columns to the Console window.

```
MyResults = ActiveDocument.Sections["Results"]
ColumnCount = MyResults.Columns.Count
for (I=1;I<= ColumnCount;I++)
Console.Write("Column#"+I+":"+MyResults.Columns[I].Name+"\r\n")
```

Methods:

Activate(), Copy(), Duplicate(), Export([optional] Filename As String, [optional] FileFormat As BqExportFileFormat [optional], IncludeHeaders As Boolean, [optional] Prompt As Boolean), ExportToStream([optional] Filename as String, [optional] FileFormat as BqExportFileFormat, [optional] IncludeHeaders as Boolean, [optional] DataStreaming as Boolean, [optional] Prompt as Boolean), GetCell(nRow As Number, nCol As Number) As Value, PrintOut([optional] FromPage As Number, [optional] ToPage As Number, [optional] [Copies As Number, [optional] Filename As String, [optional] Prompt As Boolean), Recalculate(), Remove()

Properties:

Read-only: Property Active As Boolean, Property LastPrinted as Date, Property RowCount As Number, Property Type As BqSectionType

Read-write: Property BackgroundAlternateColor as BqColorType, Property BackgroundAlternateFrequency as Number, Property BackgroundColor as BqColorType, Property BackgroundShow AlternateColor as Boolean, Property BorderColor as BqColorType, Property BorderWidth as Number, Property CSSExport as Boolean, Property ExportWithoutQuotes as Boolean, Property HTMLExportBreakRowCount as Number, Property HTMLVerticalPageBreakEnabled as Boolean, Property HTMLVerticalPageBreakUnits as BqHTMLPageBreakUnits, Property Name As String, Property ShowOutliner as Boolean, Property ShowRowNumbers As Boolean, Property ShowSortLine as Boolean, Property Visible As Boolean

Collections:

UserValues as UserValues, Limits As Limits, Columns As Columns, SortItems as SortItems

Objects:

ParentSection

TargetFact (Object)

Member of:

Facts collection

Description:

This object represents an individual Bullet, Slider, Speedometer, Thermometer and TrafficLight, (Live) Funnel Chart, and (Live) LineChart target fact object. It does not have any methods, and only includes a read-only Name property. TargetFact objects can be selected from the database if a column is available, or you can create a computed item that reflects a target amount. Data functions can be applied to fact values to aggregate values.

Example:

This example :shows how to display the name of the name of the target fact in an Alert box:

```
Alert(Speedometer.TargetFacts["Targeted Sales"].Name)
```

Methods:

None

Properties:

Read-Write: Property Name as String, DataFunction as Number

Objects:

None

TargetFacts (Collection)

Member of:

Data object

Description:

Represents all Bullet, Slider, Speedometer, Thermometer, TrafficLight, (Live) Funnel Chart, and (Live) LineChart target fact objects. TargetFact objects can be selected from the database if a column is available, or you can create a computed item that reflects a target amount. Data functions can be applied to fact values to aggregate values.

Example:

This example shows how to remove the target fact Unit Sales and adds the target fact Amount Sales:

```
Speedometer.TargetFacts.Remove("Unit Sales")
Speedometer.TargetFacts.Add("Amount Sales")
```

Methods:

Add(ItemName As String), Item (Value as NameOrIndex) Remove(Value as NameOrIndex), RemoveAll()

Properties:

Read-only: Property Count As Number

Targets (Object)

Member of:

(Live) LineChart object

Description:

Represents a target range object and its properties. The target range object indicates the unit of deviation and its value is expressed as either a percentage or absolute value from the actual target. This object is only available for a (Live) LineChart in an area chart format.

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
```

Methods:

None

Properties:

Read-write: Property Unit as BqLiveChartTargetUnit, Property Value as Number, Property Visible as Boolean

Objects:

Colors collection

Theme (Object)

Member of:

Themes collection

Description:

Represents an individual theme type for a selected 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). The theme object name for the gauge is retrieved from the widget.xml file.

Example:

This example shows how to read the theme types for a speedometer and write the results to the Console Window:

```
Console.Writeln(Speedometer.Theme.SelectedTheme)
Speedometer.Theme.SelectedTheme = "realistic"
Console.Writeln(Speedometer.Theme.SelectedTheme)
Speedometer.Theme.SelectedTheme = "simplistic"
Console.Writeln(Speedometer.Theme.SelectedTheme)
Console.Writeln(Speedometer.Theme[1].Name) (always returns realistic)
Console.Writeln(Speedometer.Theme[2].Name)(always returns simplistic)
```

Methods:

None

Properties:

Read-write: Name as string

Objects:

None

Themes (Collection)

Member of:

Bullet object, Slider object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Description:

Represents an array of existing theme types for a selected 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).

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"
```

Methods:

Item(Value as NameOrIndex)

Properties:

Read only:Count as number

Read-write: SelectedTheme as string

Objects:

Theme object

Thermometer (Object)

Member of:

Dashboard objects, Shapes collection

Description:

Represents an individual thermometer gauge in the Dashboard section.

Example:

This example shows how to name the thermometer gauge to "Sales Thermometer", disable sthe auto-scaling for the numeric ranges shown on the thermometer gauges, and sets the ranges manually:

```
Thermometer.Name = "Sales Thermometer"
Thermometer.NumericRange.AutoScale = false
Thermometer.NumericRange.MinorInterval = 10
Thermometer.NumericRange.MinScale = 1
Thermometer.NumericRange.MajorInterval =20
Thermometer.NumericRange.MaxScale = 100
```

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property),: Comments as String, Locked as Boolean, SourceSectionName as String, Subtype as String, Visible as Boolean

Read only: :Type as Number

Objects and Collections:

UserValues collection, Placement object, Subcomponents object, Themes collection, Data object

ThinClient (Object)

Member of:

Events collections

Description:

Enables or disables execution of an EPM Workspace (thin client) document event in the 8.0 client environment. This object's properties take a Boolean value, which by default is true.

Example:

This example shows how to disable all document events when running in an EPM Workspace(web-browser) environment:

```
Documents["Sample1.bqy"].Events["ThiClient"].ExecuteOnPostProcess=false
Documents["Sample1.bqy"].Events["ThinClient"].ExecuteOnPreProcess=false
Documents["Sample1.bqy"].Events["ThinClient"].ExecuteOnShutDown=false
Documents["Sample1.bqy"].Events["ThinClient"].ExecuteOnStartUp=false
```

Properties:

Read-write: ExecuteOnPostProcess as Boolean, ExecuteOnPreProcess as Boolean, ExecuteOnShutDown as Boolean, ExecuteOnStartUp as Boolean.

Read-only: Name as String

TickMark (Object)

Member of:

Bullet object, Speedometer object, Thermometer object, TrafficLight object

Description:

Represents an individual tickmark object for a selected gauge. Tickmarks are shown as short lines indicating specific values along a scale.

Example:

This example shows how to enable tick marks on the speedometer gauge:

```
ActiveDocument.Sections["Dashboard"].Shapes["Speedometer"].TickMark.Visible = true
```

Methods:

None

Properties:

Visible as Boolean

Objects:

None

Time Field (Object)

Member of:

Fields collection

Description:

Sets the current time in the HH:MM AM/PM format.

Example:

This example shows how to reposition the Time Field (Object) behind another object (such as a shape object):

```
ActiveDocument.Sections["Sales Report"].PageFooter.Fields["Time
Field"].Layer(bqLayerBack)
```

Methods:

Layer(Value as BqLayer), Spring(Name as String), UnSpring()

Properties:

Read-write: Property Formula as String, Property HorizontalAlignment as BqHorizontalAlignment (Number), Property Text as String, Property TextWrap as Boolean, Property VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Property Name as String, Property Type as BqShapeType (Number)

Objects/Col.:

UserValues, LineFormat, FillFormat, Font

TimeNow Field (Object)

Member of:

Fields collection

Description:

Sets the current time in the HH:MM:SS format.

This object represents the time when the TimeNow field is first added to the report, and it never changes.

Example:

This example shows how to concatenate the string: *Last Updated on:* and the date on which the TimeNow field was added to the report:

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

Methods:

Layer(Value As BqLayer), Spring(Name as String), UnSpring()

Properties:

Read-write: Formula as String, HorizontalAlignment as BqHorizontalAlignment (Number), Text as String, TextWrap as Boolean, VerticalAlignment as BqVerticalAlignment (Number)

Read-only: Name as String, Type as BqShapeType

Objects:

LineFormat, FillFormat, Font

Title (Object)

Member of:

Subcomponents object

Description:

Represents the title associated with a gauge or (Live) Chart.

Example:

This example shows how to set the traffic light gauge title to "July Sales" and make it visible:

```
TrafficLight.Title.Text = "July Sales"
TrafficLight.Title.Visible = true
```

Methods:

None

Properties:

Read-write:

Text as String, Visible as Boolean

Objects:

Font

Toolbar (Object)

Member of:

Toolbars collection

Description:

Represents a toolbar in the application.

Constants:

The Type property of the Toolbar (Object) uses the BqToolbars constant, which consists of these values:

- bqToolbarStandard = 1
- bqToolbarFormat = 2
- bqToolbarSections = 3
- bqToolbarNavigation = 4
- bqToolbarPaging = 5

Example:

This example shows how to hide all the toolbars in the application.

```
for(I = 1; I <= Application.Toolbars.Count;I++)
{
            MyToolbar = Application.Toolbars[I]
            MyToolbar.Visible = false
}
```

Methods:

None

Properties:

Read-only: Property Name As String, Property Type As BqToolbars

Read-write: Property Visible As Boolean

Toolbars (Collection)

Member of:

Application object

Description:

Represents all toolbars in an application.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to conceal all toolbars in an application:

```
for(I = 1; I <= Application.Toolbars.Count;I++)
Application.Toolbars[I].Visible = false
```

Methods:

Item(Value As NameOrIndex) As Toolbar

Properties:

Read-only: Property Count As Number

Objects:

Standard, Formatting, Sections, Navigation, Paging

TopLabels (Collection)

Description:

For information on the TopLabels (Collection) used in the Pivot section, see the "PivotLabels (Collection)" on page 165

For information on the TopLabels (Collection) used in the OLAPQuery section, see the "OLAPLabels (Collection)" on page 141.

TopLabelValues (Array)

Member of:

Dashboard Pivot Embedded Section Object (ESO) in Active Mode only

Description:

Returns a JavaScript array (and not an Object Model array) of top label values. Each array element contains the value of the label at successively lower levels. For example, if there are two top labels in the Pivot, Year and Month, then the array element at index 1 in the array might contain "2001" and the next element at index 2 would contain a value like "November". Because JavaScript arrays are zero-based, the element at index 0 (zero) contains an empty string value.

The TopLabelValues (Array) is accessible for a Dashboard Embedded Section Object (ESO) only. It is available even if the Pivot contains no data. In addition, it is only functional when the Pivot is in Active mode.

The TopLabelValues (Array) is refreshed each time a Pivot ESO cell is clicked in the user interface, or the OnCellDoubleClick() method is used. An array is used in this case due to the uncertainty of the TopLabel hierarchy at any given time. The array forces the user to iterate through and get the value each time to ensure it is up to date. It purposefully avoids the standard x.y.z.name structure of an object which could become ambiguous or incorrect with any change of the Pivot Top Labels.

The array is read-only. An attempt to change the values in the array has no effect, scripts continue and this action does not change any of the Pivot top label values.

The value returned by the array should be a string which reflects the names of the Pivot Top Labels in sequence. The number of values is returned using the length accessor. JavaScript arrays are normally 0-based, but for purposes of consistency with other Object Model objects, the]0] value always returns an empty string. If there are three Top labels, the array returns … TopLabelValues[0] =<empty string>, …TopLabelValues[1] =<First Top Label Name>, … TopLabelValues[2] =<Second Top Label Name>, …TopLabelValues[3] =<Last Top Label Name>.

When no Pivot top label exists, the TopLabelValues[1] returns the value “undefined”; however no exception is generated and scripts continue to execute. The same behavior occurs when a value exceeding the length of the array is called. The array functions the same whether or not data exists at the selected intersection.

Any value returned by the array is not formatted. Any formatting that you want applied to the value should be scripted additionally.

Iterating through the TopLabelValues (Array) does not persist (dirty) in the document; nor does retrieving the value. The value in the array reflects the names of the Pivot top labels in sequence. It does not persist separately from the Pivot values or in the application.

The TopLabelValues (Array) is available in Interactive Reporting Studio and Interactive Reporting Web Client. It is unavailable in EPM Workspace.

Note: You cannot use graphics to identify a selected cell using this array.

Example:

This example shows how to access top label array data:

```
var ActionName = "TopLabelValues Array"

TextBox1.Text ="Start " + ActionName

try

{

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues Default is: " +
ActiveSection.Shapes["Pivot1"].TopLabelValues
```

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues length is: " + ActiveSection.Shapes["Pivot1"].TopLabelValues.length

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues[0] is: " + ActiveSection.Shapes["Pivot1"].TopLabelValues[0]

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues[1] is: " + ActiveSection.Shapes["Pivot1"].TopLabelValues[1]

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues[2] is: " + ActiveSection.Shapes["Pivot1"].TopLabelValues[2]

PivotESOTextBox.Text = PivotESOTextBox.Text + "\r\nTopLabelValues[3] is: " + ActiveSection.Shapes["Pivot1"].TopLabelValues[3]

}

catch(e)

{

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

}

TextBox1.Text ="End " + ActionName

The results of the script are displayed as follows:

TopLabelValues Default is: ,15 TopLabelValues length is: 2 TopLabelValues[0] is: TopLabelValues[1] is: 15 TopLabelValues[2] is: undefined TopLabelValues[3] is: undefined

Related Topics

"SideLabelValues (Array) " on page 203

Topic (Object)

Member of:

DataModelSection object

Description:

Represents a topic in a Data Model or Query section.

Example 1:

This example shows how to print the names of all the topics in a Data Model to the Console window:

```
with(ActiveDocument.Sections["Query"].DataModel)
{
         TopicsCount = Topics.Count
         for(I=1;I<= TopicsCount;I++)
   Console.Write(Topics[I].DisplayName+"\r\n")
```

```
}
```

Example 2:

This example shows how to remove the *Wine Sales* topic from a query and send an alert of the modification:

```
ActiveDocument.Sections["Query"].DataModel.Topics["Wine Sales"].Remove()
Alert("Query results have been modified")
```

Methods:

Remove()

Properties:

Read-only: Property PhysicalName As String, Property Type As BqTopicType

Read-write: Property DisplayName As String, Property View As BqTopicView

Collections:

TopicItems As TopicItems

TopicItem (Object)

Member of:

Topic object

Description:

Represents a field within a topic.

Example:

This example shows how to print the names of all the topics and topic items in a Data Model to the Console window:

```
with(ActiveDocument.Sections["Query"].DataModel)
{
TopicsCount = Topics.Count
for(I=1;I<= TopicsCount;I++)
{
       Console.Write(Topics[I].DisplayName+"\r\n")
       TopicItemsCount = Topics[I].TopicItems.Count
        for(j=1;j<= TopicItemsCount;j++)
                       Console.Write(Topics[I].TopicItems[j].DisplayName )
             }
}
```

Methods:

None

Properties:

Read-only: Property PhysicalName As String

Read-write: Property DisplayName As String, Property Visible As Boolean

TopicItems (Collection)

Member of:

Topic object

Description:

Represents all fields in a topic.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to print all topics names and items in a Data Model to the Console window:

```
with(ActiveDocument.Sections["Query"].DataModel)
{
TopicsCount = Topics.Count
for(I=1;I<= TopicsCount;I++)
{
Console.Write("\r\nTopic - "+Topics[I].DisplayName+"\r\n")
TopicItemsCount = Topics[I].TopicItems.Count
for(j=1;j<= TopicItemsCount;j++)
Console.Write(Topics[I].TopicItems[j].DisplayName)
}
}
```

Methods:

Item(NameOrIndex) As TopicItem

Properties:

Read-only: Property Count As Number

Topics (Collection)

Member of:

DataModel object

Description:

Represents all topics in a Data Model.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:
`myItem = Documents[1] myItem = Documents.Item(1) myItem = Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")`

Example:

This example shows how to print the names of all topics item in a Data Model to the Console window:

```
with(ActiveDocument.Sections["Query"].DataModel)
{
TopicsCount = Topics.Count
for(I=1;I<= TopicsCount;I++)
{
Console.Write(Topics[I].DisplayName)
TopicItemsCount = Topics[I].TopicItems.Count
for(j=1;j<= TopicItemsCount;j++)
Console.Write(Topics[I].TopicItems[j].DisplayName)
}
}
```

Methods:

Add(TableObject As DMCatalogItem) As Topic, Item(NameOrIndex) As Topic, RemoveAll()

Properties:

Read-only: Property Count As Number

TrafficLight (Object)

Member of:

Dashboard objects, Shapes collection

Description:

Represents an individual traffic light gauge in the Dashboard section.

Example:

This example shows how to display the source section name of the traffic light object in a text box.:

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

var ActionName = "SourceSectionName"

TextBox1.Text ="Start " + ActionName

if (TextBox5.Text == "")
{

TextBox1.Text ="Step 1"

try
{
var TargetWidget = eval('ActiveSection.Shapes["' + LBTargetWidget.SelectedList.Item(1) +
'"]')
TextBox3.Text =  "SourceSectionName is: " + TargetWidget.SourceSectionName
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}

}

else
{
TextBox1.Text ="Step 2"

try
{
var TargetWidget = eval('ActiveSection.Shapes["' + LBTargetWidget.SelectedList.Item(1) +
'"]')
TargetWidget.SourceSectionName = eval(TextBox5.Text)
}
catch(e)
{
TextBox2.Text = "Caught: " + e.toString()
}


TextBox1.Text ="Step 3"

try
{
var TargetWidget = eval('ActiveSection.Shapes["' + LBTargetWidget.SelectedList.Item(1) +
'"]')
TextBox3.Text =  "SourceSectionName is: " + TargetWidget.SourceSectionName
}
catch(e)
```

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

Methods:

None

Properties:

Read-write: AccessibilityText (Property), AppendObjectText (Property), Comments as String, Locked as Boolean, SourceSectionName as String, Subtype as String, Visible as Boolean

Read only: Type as BqShapeType

Objects and Collections:

UserValues collection, Placement object, Subcomponents object, Themes collection, Data object

TrendLines (Collections)

Member of:

ChartSection object

Description

Represents a collection of trend line used to track trends in a data series graphically.

For this collection, the Add (Method) creates and adds a fact-bound reference line. The FactIndex is the index of the fact in the Fact array (1 for axis-bound reference lines). Axis – should be passed for axis-bound reference lines and for Scatter/Bubble charts.

Example:

This example shows how to add a trend line using the fact index value for Amount Sales:

```
ActiveDocument.Sections["Chart"].TrendLines.Add(1)
```

Methods:

Add(FactNameOrIndex as value, [optional] axis as BqChartAxisType), Item (Index as number), Remove (Index as number), RemoveAll

Properties:

Read only: Count as Number

Objects:

TrendLine object

TrendLine (Object)

Member of:

TrendLine collection

Description:

Represents an individual trend line used to track trends in a data series graphically. Interactive Reporting supports trend lines modeled after linear regression analysis. Generally the trend line is represented as a slanted line that crosses the diagram. For example, the trend line can demonstrate an increase or decrease of values over time. It may be accompanied with the calculated goodness of fit (R-squared) value.

Trend lines can be layered on top of the chart graphics (or Z axis for 3 D charts), or positioned to the background. When data is processed to create the trend line, facts from all pages of the chart are included.

Trend lines are always fact based, and only one trend line can be associated with a single fact column. In Scatter and Bubble charts, the trend line is bound to the data series. Multiple trend lines can be created when multiple data series exist if the series has been populated by the Group By feature.

Example 1:

This example shows how to retrieve the label text of the trend line and display it in an Alert box:

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

Example 2:

This example shows how to set the trend line behind the chart elements, and set the trend line on the X axis.

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

Methods:

GetLabelText([optional] Number z), GetRSquared[optional] Number z), Remove()

Properties:

Read write: Axis as BqChartAxisType, DrawingOrder as BqDrawingOrder, FactIndex as Number, LabelFormat as String, LegendFormat as String, NumberFormat as String, ShowInLegend as Boolean, ShowLabel as Boolean

Objects:

Style object, Font object

URL (Object)

Member of:

Session object

Description:

Represents a list of key value pairs generated from a GET method invocation in the current browser. URL key value pairs are variables, which are appended to the end of a URL in a Web browser.

For example:

```
http://www.yourserver.com&name=test&version=6.0&jscript=enable
```

has two key value pairs, Name and Version. The URL (Object) provides read-only access to these variables. Because URLs are browser-based, this collection applies only to the plug-in products. However, the URL (Object) is exposed in the client server products to assist in developing plug-in scripts.

Note: Do not use Session.URL.Add () and Session.URL.Item() in Interactive Reporting documents to be deployed in EPM Workspace.

Tip: All collections have the "Item(NameOrIndex)" method. This is the default method for all collections that returns collection items at a particular index or by name. Use brackets ([]) to represent calls to the Item (Method). For example, these statements are identical:

```
myItem = Documents[1] myItem = Documents.Item(1) myItem =
Documents["StartUp.bqy"] myItem = Documents.Item("StartUp.bqy")
```

Example:

This example shows how to read and use values from URLs in scripts running on the plug-in.

```
http://www.yourserver.com&app=hyperionquery&group=pm&userid=2020&jscript=enable/
// Write the URL information to the console window.
BaseURL = Application.URL
Console.Write ("The URL of my server is = "+BaseURL)
Console.Write ("The value App variable is = " + Session.URL["App"])
Console.Write ("The value Group variable is = " + Session.URL["Group"])
Console.Write ("The value UserID variable is = " + Session.URL["UserID"])
```

Methods:

Add(Key As String, Value As String), Item (Key As String) As String

UserValues (Collection)

Member of:

(Live) BarChart, object, Bullet object, Document collection, (Live) FunnelChart object, (Live) LineChart, (Live) PieChart object, (Live) RadarChart object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Description:

Enables you to create a user-defined or arbitrary value to an object in the Object Model. A user-defined value, in this context, is not explicitly defined in the Object Model or other software components of the product. It is a value defined solely by the Designers using the Object Model.

The user-defined value takes the form of a named string value. It is created from the JavaScript core object *String*. The individual characters in a string can range in value from 0 to 255.

Any user-defined value persists across uses of the document. That is, if the document is saved after a user-defined value has been created, the value is still present and accessible when the document is reopened.

This feature is available for Interactive Reporting documents to be deployed in the Interactive Reporting Studio, Interactive Reporting Web Client and the EPM Workspace. The persistence of user-defined values does not effect the Oracle Enterprise Performance Management Workspace, Fusion Edition unless the end user chooses to save the document state. Persistence works with Interactive Reporting Web Client in all its modes.

Depending on the amount of data stored as a user-defined value with the document, Interactive Reporting may take extra time to read, write, compress and decompress.

For detailed information on how to add and maintaining user-defined values through the Object Model, see "Adding and Maintaining the UserValue Object" on page 245.

Uses:

The UserValues object is effective for:

- Data that is temporary and needs to be saved and reused. For example, if you have a Dashboard form that collects parameters from the user by way of various controls (combo boxes, list boxes, etc.), use the UserValues object to recall the settings for the end user the next time the user accesses the document.
- External software that recognizes a Oracle's Hyperion® Interactive Reporting Studio or Interactive Reporting Web Client document format. In this case, the UserValue object could be used to pass information into the document for use when the document is opened. The persistence of values also enables the external software programs to use the information as data intrinsically relevant.
- OLE-enabled external software. The UserValues object can use the Object Model to store data in an Interactive Reporting document (.bqy) document, which can be used in turn when the document is opened to control the behavior of the document. On the other hand, documents can store data that can be ready by OLE-enabled software and used to alter the program's behavior accordingly.

Methods:

Add(String Name, [optional] String Value), Item(Value NameOrIndex), RemoveAll

Properties:

Read-only: Count as Number

Objects:

`UserValue object`

ValuesAxis (Object)

Member of:

ChartSection object

Description:

Logically represents all the properties of a charts values axis.

Example:

This example shows how to set some basic properties for the left axis:

```
with(ActiveDocument.Sections["Chart"])
{
ValuesAxis.LeftAxis.AutoScale = true
ValuesAxis.LeftAxis.ShowLabel = false
ValuesAxis.RightAxis.AutoScale = true
ValuesAxis.RightAxis.ShowLabel = false
ValuesAxis.RightAxis.LabelText = "Right Axis"
}
```

Methods:

None

Properties:

Read-write: Property AdjustableScale as Boolean, Property ShowIntervalTickmarks As Boolean, Property ShowIntervalValues As Boolean, Property ShowValuesAtRight As Boolean

Objects:

LeftAxis As LeftAxis, RightAxis As RightAxis

ValueLabels (Object)

Member of:

BarChart object, BlockChart object, Bullet object, Control object, FunnelChart object, LineChart object, PieChart object, RadarChart object, Slider object, Speedometer object, Thermometer object, TrafficLight object

Description:

Represents an individual value label object for a gauge or (Live) chart object. Setting this property is equivalent to setting value label number properties on the Number dialog box.

Example:

This example shows how to enable the value labels and set the number format for the value labels to $#,##0.00.:

```
BarChart.ValueLabels.Visible= true
BarChart.ValueLabels.NumberFormat = NumberFormat = "$#,##0.00"
```

Methods:

None

Properties:

Read-write: NumberFormat as String, Visible as Boolean

Objects:

Font object

WebClientDocument (Object)

Member of:

Documents collection, Application object

Applies to

Interactive Reporting Web Client

Description:

Represents an Interactive Reporting document that has been opened inside Oracle's Hyperion® Interactive Reporting Web Client . This object, which is based on a document object, has similar methods and properties. A WebClientDocument object also has methods and properties that are specific to Web environments.

Methods:

Activate(),Close([SaveChanges As Boolean]), Import(Filename As String, FileType As Number), ImportSQLFile(Filename As String), Save([bCompressed As Boolean]), SaveAs([Filename As String], [bCompressed As Boolean]), Send(To As String, [CC As String], [Subject As String], [Message As String], [SaveResults As Boolean], [Compressed As Boolean]) As Number, SetODSPassword(Password as String)

Properties:

Read-only: Property AdaptiveState as BqAdaptiveState, Property Active As Boolean, Property LastSaved As Date, Property Name As String, Property Path As String, Property Url as String

Read-write: Property ShowCatalog As Boolean, Property ShowSectionTitleBar As Boolean, Property Username as String

Collections:

Sections As Sections

XAxisLabel (Object)

Member of:

LabelsAxis object

Description:

Represents a Chart X axis label. This object's properties directly affect the display of the X axis and correspond to the options provided on the Label Axis tab of the Properties dialog box.

Example:

This example shows how to modify the properties of the X axis label:

```
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.AutoFrequency = true
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.LabelFrequency = 3
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.LabelText = "X Axis"
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.ShowLabel = true
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.ShowTickmarks = false
ActiveDocument.Sections["Chart1"].LabelsAxis.Xaxis.ShowValues = true
ActiveDocument.Sections["Chart1"].LabelsAxis.XAxis.TickmarkFrequency = 4
```

Methods:

None

Properties:

Read-write: Property AutoFrequency As Boolean, Property LabelFrequency As Number, Property LabelText As String, Property ShowLabel As Boolean, Property ShowTickmarks As Boolean, Property ShowValues As Boolean, Property TickmarkFrequency As Number

XCategories (Collection)

Description:

See "Body (Object)" on page 36on page 36.

XCategory (Object)

Member of:

CategoryItems collection

Description:

Represents a Chart X axis. This object's properties directly affect the display of the X axis and the X Categories in the Outliner.

Example:

This example shows how to sort by facts for *Quarter* X categories:

```
ActiveDocument.Sections["BooksChart"].XCategories["Quarter"].SortByFact()
```

Methods:

Hide(), SortByFact(FactName by String, SortFunction BqSortFunction, [optional] BqSortOrder SortOrder), SortByLabel{[optional] SortOrder as BqSortOrder }

Properties:

Read-only: Property Name as String, SortFactName as String, SortFunction as Number, SortOrder as Number

XFacts (Collection)

Member of:

ChartSection object

Description:

The XFacts (Collection) is a collection of XFact objects, each of which represents a fact object on the X axis of a scatter or bubble chart.

Example:

This example show how to remove "Unit Sales" and add "Amount Sales".

```
ActiveDocument.Sections["Chart2"].XFacts.Remove("Unit_Sales")
ActiveDocument.Sections["Chart2"].XFacts.Add("Amount_Sales")
```

Methods:

Add(Item Name as String), AddComputedItem(Name as String, Expression as String, [optional] Index as Number), Item(Value as Name or Index), Remove(Value as Name or Index), RemoveAll()

Properties:

Read Only AxisType as BQChartAxisType constant, Count as Number

Constants:

The AxisType (Property) uses the BqChartAxisType constant group, which consists of these values:

- bqChartXAxis
- bqChartYAxis
- bqChartZAxis

XFact (Object)

Member of:

XFacts (Collection)

Description:

Represents a fact object on the X axis for a scatter or bubble chart.

Example:

This example show how to hide "Amount Sales" from the X Axis of the Outliner.

```
ActiveDocument.Sections["Chart2"].XFacts["Amount Sales"].Hide()
```

Methods:

Hide(), UnHide()

Properties:

Read only: Name as String

XLabels (Object)

Member of:

ChartSection object

Description:

Represents a label value on the X axis. This object's properties directly affect the display of the label value on the X axis and correspond to the options provided on the Chart menu or shortcut menu.

Note: You must specify the label values in an array before using the FocusSelection (Method), HideSelection (Method), and UnHide (Method).

Example:

This example shows how to modify the label value on the X-axis:

```
var OArray = new Array()
OArray[0]= ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(1)
OArray[1]= ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(3)
var ZArray = new Array()
ZArray[0]= ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(2)
ZArray[1]= ActiveDocument.Sections["Chart"].XLabels.LabelValues.Item(4)
ActiveDocument.Sections["Chart"].XLabels.FocusSelection(OArray)
ActiveDocument.Sections["Chart"].XLabels.HideSelection(ZArray)
ActiveDocument.Sections["Chart"].XLabels.UnhideAll(ZArray)
```

Methods:

DrillInto(NameOrIndex As Value, DrillName As String), FocusSelection(ItemArray As Value), HideSelection(ItemArray As Value), UnhideAll()

Properties:

Read-only: Property Count as Number

Objects:

LabelValues As LabelValues

YFacts (Collection)

Member of:

ChartSection (Collection)

Description:

Collection of YFact objects, each of which represents a fact object on the Y axis of a scatter or bubble chart.

Example:

This example shows you how to add the values from Amounts Sales to the Y Axis in the data layout.

```
ActiveDocument.Sections["Chart2"].YFacts.Add("Amount_Sales")
```

Methods:

Add(Item Name as String), AddComputedItem(Name as String, Expression as String, [optional] Index as Number), Item(Value as Name or Index), Remove(Value as Name or Index), RemoveAll

Properties:

Read Only AxisType as BqChartAxisType constant, Count as Number

Constants:

The AxisType (Property) uses the BqChartAxisType constant group, which consists of these values:

- bqChartXAxis
- bqChartYAxis
- bqChartZAxis

YFact (Object)

Member of:

YFacts (Collection)

Description:

Represents a fact object on the Y axis for a scatter or bubble chart.

Example:

This example show how to hide “Amount Sales” from the Y Axis of the Outliner.

```
ActiveDocument.Sections["Chart2"].YFacts["Amount Sales"].Hide()
```

Methods:

Hide(), UnHide()

Properties:

Read only: Name as String

YLabels (Object)

Member of:

ChartSection object

Description:

Represents a label value on the Y axis. This object's properties directly affect the display of the label on the Z axis and correspond to the options provided on the Chart menu or shortcut menu.

Note: You must specify the label values in an array before using the FocusSelection (Method), HideSelection (Method), and UnHide (Method).

Example:

This example shows how to modify the label value on the Y axis:

```
var OArray = new Array()
OArray[0]= ActiveDocument.Sections["Chart"].YLabels.LabelValues.Item(1)
OArray[1]= ActiveDocument.Sections["Chart"].YLabels.LabelValues.Item(3)
var ZArray = new Array()
ZArray[0]= ActiveDocument.Sections["Chart"].YLabels.LabelValues.Item(2)
ZArray[1]= ActiveDocument.Sections["Chart"].YLabels.LabelValues.Item(4)
ActiveDocument.Sections["Chart"].YLabels.FocusSelection(OArray)
ActiveDocument.Sections["Chart"].YLabels.HideSelection(ZArray)
ActiveDocument.Sections["Chart"].YLabels.UnhideAll(ZArray)
```

Methods:

DrillInto(Value as NameOrIndex, DrillName As String), FocusSelection(Value As ItemArray), HideSelection(Value as ItemArray), UnhideAll()

Properties:

Read-only: Property Count as Number

Objects:

LabelValues As LabelValues

ZAxisLabel (Object)

Member of:

LabelsAxis object

Description:

Represents a Chart's Z axis. This object's properties directly affect the display of the Z axis label.

Example:

This example shows how to modify the properties of the Z axis label:

```
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.AutoFrequency = true
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.LabelFrequency = 3
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.LabelText = "X Axix"
```

```
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.ShowLabel = true
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.ShowTickmarks = false
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.ShowValues = true
ActiveDocument.Sections["Chart1"].LabelsAxis.ZAxis.TickmarkFrequency = 4
```

Methods:

None

Properties:

Read-write: Property LabelText As String, Property ShowLabel As Boolean, Property ShowTickmarks As Boolean, Property ShowValues As Boolean

ZCategories (Collection)

Description:

See "Body (Object)" on page 36on page 36.

ZCategory (Object)

Member of:

CategoryItems collection

Description:

Represents a Chart's Z axis. This object's properties directly affect the display of the Z axis and the Z Categories in the Outliner.

Example:

This example shows how to sort the "Quarter" X categories by label:

```
ActiveDocument.Sections["BooksChart"].XCategories["Quarter"].SortByLabel()
```

Methods:

Hide(), SortByFact(String FactName, BqSortFunction SortFunction, [optional] BqSortOrder SortOrder), SortByLabel{[optional] BqSortOrder SortOrder}

Properties:

Read-only: Property Name as String, SortFactName as String, SortFunction as Number, SortOrder as Number

ZFacts (Collection)

Member of:

ChartSection object

Description:

Collects ZFact objects, each of which represents a fact object on the Z axis (size) of bubble charts.

Example:

This example shows how to create a computed value entitled Double Sales on the Z Axis (Size), which doubles the amount in the Unit Sales bubble.

```
ActiveDocument.Sections["Chart"].ZFacts.AddComputedItem("Double_Sales", "Unit_Sales *2",
2)
```

Methods:

Add(Item Name as String), AddComputedItem(Name as String, Expression as String, [optional] Index as Number), Item(Value as Name or Index), Remove(Value as Name or Index), RemoveAll

Properties:

Read Only: AxisType as BqChartAxisType constant, Count as Number

Constants:

The AxisType (Property) uses the BqChartAxisType constant group, which consists of these values:

- bqChartXAxis
- bqChartYAxis
- bqChartZAxis

ZFact (Object)

Member of:

ZFacts (Collection)

Description:

Represents a fact object on the Z axis (Size) for a bubble chart. The ZFact (Object) a value that defines the size or width dimension of a bubble in proportion to the amount of data.

Example:

This example show how to hide “Amount Sales” from the Size Axis of the Outliner

```
ActiveDocument.Sections["Chart2"].ZFacts["Amount Sales"].Hide()
```

Methods:

Hide(), UnHide()

Properties:

Read only: Name as String

ZLabels (Object)

Member of:

ChartSection object

Description:

Represents a label value on the Z axis. This object's properties directly affect the display of the label on the Z axis and correspond to the options provided on the Chart menu or shortcut menu.

Note: You must specify the label values in an array before using the FocusSelection (Method), HideSelection (Method), and UnHide (Method).

Example:

This example shows how to modify the label value on the Z axis:

```
var OArray = new Array()
OArray[0]= ActiveDocument.Sections["Chart"].ZLabels.LabelValues.Item(1)
OArray[1]= ActiveDocument.Sections["Chart"].ZLabels.LabelValues.Item(3)
var ZArray = new Array()
ZArray[0]= ActiveDocument.Sections["Chart"].ZLabels.LabelValues.Item(2)
ZArray[1]= ActiveDocument.Sections["Chart"].ZLabels.LabelValues.Item(4)
ActiveDocument.Sections["Chart"].Zlabels.FocusSelection(OArray)
ActiveDocument.Sections["Chart"].Zlabels.HideSelection(ZArray)
ActiveDocument.Sections["Chart"].Zlabels.UnhideAll(ZArray)
```

Methods:

DrillInto(NameOrIndex As Value, DrillName As String), FocusSelection(ItemArray As Value), HideSelection(ItemArray As Value), UnhideAll()

Properties:

Read-only: Property Count as Number

Objects:

LabelValues As LabelValues

Adding and Maintaining the UserValue Object

Use this section to add and maintain a UserValue object. For syntax, purposes and examples, see the "UserValues (Collection)" on page 233. This section describes how to add, retrieve, and remove UserValue objects.

Adding a UserValue object

➤ To set a user-defined value for a selected object:

1 **Define a string which contains the desired value.**

For example, type:

```
MyString.
```

2 **Invoke the Add (Method) to create an item in the UserValues (Collection) to store this value.**

A string containing the name for this value is the only parameter needed for the Add (Method) (that is, Object.UserValues.Add(name,[optional]value).

The Add (Method) adds the named item to the collection so that it can be accessed through the object model and can be written and read as part of the Interactive Reporting document format. The UserValue object has one method: Remove. It also has three properties as described below:

- **Name** — A read-only property which returns the name of an UserValue object. The Add method creates the item with this property set to the string value specified as the argument to the method. This feature is useful when you need to enumerate the items in the collection, such as in a for loop based on the Count property of the object's UserValues (Collection).
- **Value** — A property of the string type, which is read/write. The Add method creates the item with the Value (Property) set to the empty string ("") unless the optional second parameter, which is of type string, sets the value.
- **ReadOnly** A property of the Boolean type, which is read-only. The Add (Method) creates the item with the read-only property set to false.

  The Add (Method) increments the Count (Property) of the object's UserValues (Collection) by one.

  When the document containing this item is saved, it stores the collection containing the item so that it is documentable. Consequently, any external software systems can follow the documentation to add user-defined values directly into an Oracle's Hyperion® Interactive Reporting document format.

  Errors resulting from adding the UserValue object get written to the Console window. Typically, errors result from these conditions:

- Memory that cannot be allocated to hold the item
- An item of the same name already exists.

3 **Assign or reassign the Value property of the UserValue object. For example, you might type:**

```
Object.UserValues[name].Value=sMyString.
```

Retrieving a UserValue object

➤ To retrieve the user defined value for any specific object:

1 **Define a string that contains the desired value.**

2 **Access the Value property for the desired item in the UserValues (Collection) of the object, for example:**

```
SMyString=Object.UserValues[name].Value.
```

If the named/indexed item does not exist in the UserValues (Collection) for the specified object and the error occurs outside of a "try-catch" block, the message: "No such property" is written to the Console window, and the script fails.

To continue executing a script when a value cannot be identified, include the Value syntax in a *try-catch* block.

Removing a UserValue object

➤ To remove the user defined value for any specific object:

➤ Invoke the Remove method for the item.

For example, type:

```
Object.UserValues[name].Remove()
```

The UserValue object has one method: Remove. The Remove (Method) behaves accordingly:

- Removes the named property for the object from the document and decrements the collection's Count property by one.
- Scripts that reference the user-defined values fails unless the value is identified within a *try-catch* block. If the value is not identified within a *try-catch* block, the error message "Script Execution Error "%" (where % represent items not found) is written to the Console window.
- If the named item value has its ReadOnly property set to true, the error message "Script Execution Error "%" (where % represents items that cannot be removed) is written to the Console window.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z