Inserting Objects

Because of the asynchronous nature of operations within the JS-Client implementation of OWrite, many operations, such as inserting objects, must be handled with this in mind. To avoid such operations turning into difficult asynchronous management task, we designed *$insert()* in such a way that all properties for the object can be specified as part of the insert action. In order to prepare the data for an object, we have provided a set of helper methods that can be used to populate a javascript object variable with all relevant properties. This approach has greatly simplified the inserting of even the most complex of objects.

This document serves as a reference guide for inserting different types of objects. It is organised into two sections. The first section explains the helper methods and their parameters. The second section provides example code and descriptions for all object properties for each type of object that can be inserted into an OWrite document.

Please use this reference in conjunction with the OWrite/JS-OWrite documentation and the OWrite/JS-OWrite document manager examples.

The documentation can be viewed at

https://supportpublic.brainydata.com/documentation/html/owrite/

The examples can be downloaded from

https://demos.brainydata.com/download.htm

2019 © Brainy Data Ltd

Helper Methods

There are a total of six helper methods that prepare the object data for insertion. Two are essential for all types of objects, whereas the remaining four methods help create the object data for table fields with rows, cells and cell content. This section lists each helper method, describing its function, its parameters, the return value and syntax. Code examples are provided in the *Object Reference* section.

$initparams(pObjType)

When preparing for any type of object, the first method that one must call is *$initparams*.

Parameters:

pObjType - specifies one of the kWriObjType… constants.

Returns:

the initialised Javascript object variable which is used with subsequent calls to $addparams().

Syntax:

```
Do $cinst.$objs.OWrite.$initparams(kWriObjTypeTable) Returns params
```

$addparams(pParams,pName,pValue[,pName,pValue,…])

After calling $initparams, this function can be called repeatedly to add additional object properties to the Javascript object variable. Object properties are added using two parameters each that specify the name-value pair for the property. Multiple properties can be added in a single call.

Parameters:

pParams - the Javascript object variable returned by *$initparams*.

pName - the property name

pValue - the property value

Returns:

returns kTrue if the properties were successfully added.

Syntax:

```
Do $cinst.$objs.OWrite.$addparams(params,
                 "ObjName","table1",
                 "ObjTableCellSpacing",2000,
                 "ObjTableFlags",'PageHeaders')
```

$addparamstblrow(pParams,pRowNo,pName,pValue[,pName,pValue,…])

For use with table objects only, this method adds the data for a table row to the Javascript object variable, or updates the specified table row with the specified properties.

Parameters:

pParams - the Javascript object variable returned by *$initparams*.

pRowNo - must be zero the first time this method is called to add a row, subsequent calls to update the properties of the row must specify the row number returned by the first call to this method.

pName - the property name

pValue - the property value

Returns:

returns the row number for the row that was added or updated.

Syntax:

```
Do $cinst.$objs.OWrite.$addparamstblrow(params,0,
                ”ObjTableRowType",kWriTblRowHeader,
                "ObjTableRowFlags",'')
                Returns rowNo
```

$addparamstblcell(pParams,pRowNo,pCellNo,pName,pValue[,pName,pValue,…])

For use with table objects only, this method adds the data for a table cell to the Javascript object variable, or updates the specified table cell with the specified properties.

Parameters:

pParams - the Javascript object variable returned by *$initparams*.

pRowNo - must specify the row number returned by the first call to the $addparamstblrow.

pCellNo - must be zero the first time this method is called to add a cell, subsequent calls to update the properties of the cell must specify the cell number returned by the first call to this method.

pName - the property name

pValue - the property value

Returns:

returns the cell number for the cell that was added or updated.

Syntax:

```
Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
                "ObjWidth",objWidth,
                "ObjHeight",8000,
                "ObjFlags",'AutoSize')
                Returns cellNo
```

$addparamstblcellcontent(pParams,pRowNo,pCellNo,pContentText,pName,pValue[,pName,pValue,…])

For use with table objects only, this method adds the data for the specified table cell content to the Javascript object variable, or updates the specified table cell content with the specified properties.

Parameters:

pParams - the Javascript object variable returned by *$initparams*.

pRowNo - must specify the row number returned by the call to the $addparamstblrow.

pCellNo - must specify the cell number returned by the call to $addparamstblcell.

pContentText - specifies the text for the cell content.

pName - the property name

pValue - the property value

Syntax:

```
Do $cinst.$objs.OWrite.$addparamstblcellcontent(params,rowNo,cellNo,"",
                  "StyleSuper","Normal",
                  "Bold",kTrue,
                  "TextColor",rgb(0,0,0),
                  "SpaceBefore",0,
                  "SpaceAfter",0)
```

Object Reference

This section will list the properties for each object and provide sample code on how to prepare the object data for insertion.

Note: properties marked with an asterisk are required during a call to $insert.

Calculated Field

The calculated field is typically used to merge data from DB table columns directly into the text-flow of a document. The key properties for this object are the calculation to retrieve the data and the various display options when the field has not been merged. It has very few formatting properties for merging data as the merged data takes on the style of the current text run or the merged data consists of RTF which can provide its own formatting.

Calculated Field Properties	
ObjCalc *	(character/calculation) The calculation that is executed during a merge
ObjClickCalc	(character/calculation) The calculation that is executed when the object is clicked.
ObjDisplay	(character/plain text) The text to display when not merged.
ObjFillColor *	(integer/colour constant or RGB value) The object fill colour when not merged.
ObjFlags	(character/comma list any combination of “GenClick, ShowRTF, ShowingRTF, StripEmpty”) Please refer to the section entitled ‘Object Flags’ at the end of the Object Reference section.
ObjHeight	(integer/owrite units 1/1000mm) Used only when merged text is to be clipped to the bounding box of the field. Default is zero.
ObjName *	(character/plain text) Used to identify the object during certain events.
ObjToolTip	(character/plain text) The text that is displayed in a tooltip when the mouse hovers over the object.
ObjUserData	(any-data-type/undefined) Developer specific data for the object
ObjWidth	(integer/owrite units 1/1000mm) Used only when merged text is to be wrapped and limited to the bounding box of the field. Default is zero.

Calculated Field Sample Code

```
;  prepare our object parameters for the call to OWrite.$insert

;  start a new set of parameters for inserting a calculated field
Do $cinst.$objs.OWrite.$initparams(kWriObjTypeCalc) Returns params

;  specify the fields attributes
Do $cinst.$objs.OWrite.$addparams(params,”ObjName”,pName,"ObjCalc",pCalc)
Do $cinst.$objs.OWrite.$addparams(params,”ObjDisplay”,pDisplay)
Do $cinst.$objs.OWrite.$addparams(params,"ObjFillColor",rgb(196,196,196))

;  insert the object at current position
Do $cinst.$objs.OWrite.$insert(kWriObjTypeCalc,params,kWriInsertOver,
                               con("Insert",kWriObjTypeCalc)) Returns ok
Quit method ok
```

Info Field

The info field is a special field that displays the current page number, the document page count, the current date or the current time. It has very few formatting properties as the text displayed by the field takes on the style of the current text run.

Info Field Properties	
ObjInfoType *	(integer/kWriObjTypeInfo… constant)
ObjFillColor *	(integer/colour constant or RGB value) The object fill colour for display while editing. This is not shown when the document is printed or exported.

Info Field Sample Code

```
;  prep params for inserting a doc info object
Do $cinst.$objs.OWrite.$initparams(pObjType) Returns params
Do $cinst.$objs.OWrite.$addparams(params,"ObjInfoType",kWriObjTypeInfoDate)
Do $cinst.$objs.OWrite.$addparams(params,"ObjFillColor",rgb(200,255,200))

;  insert the object at current position
;  (last two parameters are async message and data for evAsyncDone event)
Do $cinst.$objs.OWrite.$insert(kWriObjTypeInfo,params,kWriInsertOver,
                               "Insert",kWriObjTypeInfo) Returns ok
Quit method ok
```

Picture / Calculated Picture Field

The picture or calculated picture field is the first of our more complex objects. As well as the basic set of properties, picture and calculated picture fields have box and floating positioning properties in addition to standard inline properties, which we have already encountered with calculated fields.

Note: some properties only apply to calculated pictures which will be marked with *(CP)* and some properties only apply to ordinary picture fields which will be marked with *(P)*.

Picture (P) & Calculated Picture (CP) Properties	
ObjAlign	(integer/kWriObjAlign… constant) Applies to in-line pictures and specifies the vertical positioning relative to the text.
ObjCalc (CP)	(character/calculation) The calculation that is executed during a merge
ObjClickCalc	(character/calculation) The calculation that can be executed on the server when the object is clicked or some other data such as an URL which could be handled on the client. See evObjClick.
ObjData (P)	(binary/png) The image data in raw png format. For calculated pictures this should be empty.
ObjDataSrc (P)	(character/calculation) The calculation that can be executed on the server to fetch the external source for the image. Using ObjDataSrc reduces the document size by allowing you to store image data external to OWrite documents. This external image data can then also be shared by multiple documents. See evGetDataFromSource.

Picture (P) & Calculated Picture (CP) Properties	
ObjDisplay (CP)	(integer/icon-id) The picture to display when not merged. This is currently ignored by JS-OWrite and the supplied ow_defimage.png image is used.
ObjFlags	(character/comma list any combination of “GenClick,LockAspect,UseClickCalc”) Please refer to the section entitled ‘Object Flags’ at the end of the Object Reference section.
ObjFormat*	(integer/kWriObjFmt… constant) Specifies the floating option of the object; i.e. in-line, wrap, etc.
ObjHeight*	(integer/owrite units 1/1000mm) The current display height of the object.
ObjHorzOffset	(integer/owrite units 1/1000mm) The horizontal offset of the object from its anchor point which is typically the beginning of the nearest paragraph above the image.
ObjName	(character/plain text) Used to identify the object during certain events.
ObjOrigHeight*	(integer/owrite units 1/1000mm) The original height of the image. This needs to be calculated based on the number of pixels and the image’s intended DPI.
ObjOrigWidth*	(integer/owrite units 1/1000mm) The original width of the image. This needs to be calculated based on the number of pixels and the image’s intended DPI.
ObjToolTip (CP)	(character/plain text) The text that is displayed in a tooltip when the mouse hovers over the object.
ObjUserData	(any-data-type/undefined) Developer specific data for the object.
ObjVertOffset	(integer/owrite units 1/1000mm) The vertical offset of the object from its anchor point which is typically the beginning of the nearest paragraph above the image.
ObjWidth*	(integer/owrite units 1/1000mm) The current display width of the object.
ObjWDBottom	(integer/owrite units 1/1000mm) The text wrapping distance from the bottom edge.
ObjWDLeft	(integer/owrite units 1/1000mm) The text wrapping distance from the left edge.
ObjWDRight	(integer/owrite units 1/1000mm) The text wrapping distance from the right edge.
ObjWDTop	(integer/owrite units 1/1000mm) The text wrapping distance from the top edge.

Picture Sample Code

```
;  prep params for inserting a picture object
Do $cinst.$objs.OWrite.$initparams(kWriObjTypePict) Returns params
```

```
;  set floating and click properties
Do $cinst.$objs.OWrite.$addparams(params,
              ”ObjFormat",kWriObjFmtWrap,
              “ObjFlags",'GenClick,LockAspect,UseClickCalc',
              “ObjName”,’BD_LOGO’,
              “ObjClickCalc”,“https://www.brainydata.com”)
```

```
;  set distance from anchor point 10 mm horizontally & vertically
Do $cinst.$objs.OWrite.$addparams(params,
              ”ObjHorzOffset",10000,
              "ObjVertOffset",10000)
```

```
;  display image at half its original size
Calculate img_dpi as 150
Calculate orig_width as img_px_width * 25400 / img_dpi
Calculate orig_height as img_px_height * 25400 / img_dpi
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjOrigWidth", orig_width,
              "ObjOrigHeight", orig_height,
              "ObjWidth", orig_width/2,
              "ObjHeight", orig_height/2)
```

```
;  add the image data
Do $cinst.$objs.OWrite.$addparams(params,"ObjData",ivBDLogoPNG)
```

```
;  insert the object at current position
;  (last two parameters are the async message and data for evAsyncDone event)
Do $cinst.$objs.OWrite.$insert(kWriObjTypePict,params,kWriInsertOver,
              "Insert",kWriObjTypePict) Returns ok
Quit method ok
```

Text-box

OWrite text boxes have a very similar set of properties to that of a plain picture or calculated picture field. Text boxes can have calculations and they can float, although they cannot be set to in-line floating. Text boxes also have additional box properties, such as borders and inner margins that separate their content from the borders.

Text-box Properties	
ObjCalc	(character/calculation) The calculation that is executed during a merge. This is empty for simple text boxes that are edited manually.
ObjDisplay	(character/plain text) The text to display when not merged and the text box has a calculation for merging.
ObjFlags	(character/comma list any combination of “AutoSize, NoEnter, StripEmpty, ShowRTF”) Please refer to the section entitled ‘Object Flags’ at the end of the Object Reference section.
ObjFormat*	(integer/kWriObjFmt… constant) Specifies the floating option of the object; i.e. Behind text, Wrap, etc.
ObjHorzOffset	(integer/owrite units 1/1000mm) The horizontal offset of the object from its anchor point which is typically the beginning of the nearest paragraph above the image.
ObjName	(character/plain text) Used to identify the object during certain events.
ObjToolTip	(character/plain text) The text that is displayed in a tooltip when the mouse hovers over the object.
ObjUserData	(any-data-type/undefined) Developer specific data for the object.
ObjVertOffset	(integer/owrite units 1/1000mm) The vertical offset of the object from its anchor point which is typically the beginning of the nearest paragraph above the image.
ObjWDBottom	(integer/owrite units 1/1000mm) The text wrapping distance from the bottom edge.
ObjWDLeft	(integer/owrite units 1/1000mm) The text wrapping distance from the left edge.

ObjWDRight	(integer/owrite units 1/1000mm) The text wrapping distance from the right edge.
ObjWDTop	(integer/owrite units 1/1000mm) The text wrapping distance from the top edge.
Box Properties	see section 'Text-Box and Table-Cell Box Properties' below for additional properties.

Text-box Sample Code

```
;  prep params for inserting a picture object
Do $cinst.$objs.OWrite.$initparams(kWriObjTextbox) Returns params

;  set floating properties
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjFormat",kWriObjFmtWrap,
              "ObjFlags",'AutoSize')

;  set calculation properties
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjName",'Address',
              "ObjCalc",'$cinst.$getAddress()',
              "ObjDisplay",'Client Address',
              "ObjTooltip",'Field merges client's full address')

;  set distance from anchor point 10 mm horizontally & vertically
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjHorzOffset",10000,
              "ObjVertOffset",10000)

;  display box at 3cm wide with minimum height of 2cm
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjOrigWidth", 30000,
              "ObjOrigHeight", 20000,
              "ObjWidth", 30000,
              "ObjHeight", 20000,
              "ObjRectWidth", 30000,
              "ObjRectHeight", 20000)

;  add the image data
Do $cinst.$objs.OWrite.$addparams(params,"ObjData",ivBDLogoPNG)

;  insert the object at current position
;  (last two parameters are async message and data for evAsyncDone event)
Do $cinst.$objs.OWrite.$insert(kWriObjTypeTextbox,params,kWriInsertOver,
              "Insert",kWriObjTypeTextbox) Returns ok
Quit method ok

Quit method ok
```

Text-Box and Table-Cell Box Properties

The box properties that commonly apply to text-box and table-cell objects.

Text Box / Table Cell - Box Model Properties	
	Border and Fill
ObjBordColor	(integer/colour constant or RGB value) The object border colour.

ObjBordLnSize	(integer/points in fractions of hundred) The border line thickness.
ObjBordLnStyle	(integer/kWriLine… constant) The border line style.
ObjBordSides	(integer/kWriFrame… constant) The border frame options.
ObjBordStyle	(integer/kWriBord… constant) The border style.
ObjFillColor	(integer/colour constant or RGB value) The object fill colour.
	Inner Margins
ObjBottomMargin	(integer/owrite units 1/1000mm) The inner margin between bottom border and text
ObjLeftMargin	(integer/owrite units 1/1000mm) The inner margin between left border and text
ObjRightMargin	(integer/owrite units 1/1000mm) The inner margin between right border and text
ObjTopMargin	(integer/owrite units 1/1000mm) The inner margin between top border and text
	Box Dimensions
ObjHeight*	(integer/owrite units 1/1000mm) Minimum height of object
ObjWidth*	(integer/owrite units 1/1000mm) Fixed width of object
ObjRectHeight	(integer/owrite units 1/1000mm) Actual current height of object
ObjRectWidth	(integer/owrite units 1/1000mm) Actual current width of object
ObjOrigHeight	(integer/owrite units 1/1000mm) Should be same as ObjHeight during insert
ObjOrigWidth	(integer/owrite units 1/1000mm) Should be same as ObjWidth during insert

Table, Table Rows and Table Cells

In order to insert an entire table, the properties for the table, the properties for each table row and the properties for each table cell within each row must be prepared. This may seem daunting at first, but a systematic explanation followed by an example will reveal its simplicity.

Table Properties	
ObjCalc	(list/list) The calculation that is executed during a merge. This can be empty for tables that are edited manually.
ObjName	(character/plain text) Used to identify the object during certain events.
ObjTableAlign	(integer/kWriStyParaJst… constant) Horizontal alignment within page margins
ObjTableCellSpacing	(integer/owrite units 1/1000mm) spacing between cells
ObjTableFlags	(character/comma list any combination of “PageHeaders, PageFooters, AutoExtend, MultiDataRows”) Please refer to the section entitled ‘Object Flags’ at the end of the Object Reference section.
ObjTableIndent	(integer/owrite units 1/1000mm) indent from left page margin
ObjToolTip	(character/plain text) The text that is displayed in a tooltip when the mouse hovers over the object.

ObjUserData	(any-data-type/undefined) Developer specific data for the object.

Table Row Properties	
ObjTableRowFlags	(character/comma list any combination of “CanSplit”) Please refer to the Flags table towards the end of the Object Reference section.
ObjTableRowType	(integer/kWriTblRow… constant) The row type, header, normal or footer.

Table Cell Properties	
ObjCalc	(character/calculation) The calculation that is executed during a merge. This is empty for cells that have no calculation and are edited manually.
ObjDisplay	(character/plain text) The text to display when not merged and the cell has a calculation for merging.
ObjFlags	(character/comma list any combination of “AutoSize, NoEnter, StripEmpty”) Please refer to the section entitled ‘Object Flags’ at the end of the Object Reference section.
ObjName	(character/plain text) Used to identify the object during certain events.
ObjToolTip	(character/plain text) The text that is displayed in a tooltip when the mouse hovers over the object.
Box Properties	see section ‘Text-Box and Table-Cell Box Properties’ below for additional properties.

Table Cell Content (standard text formatting properties)	
Bold	(boolean/kTrue or kFalse) Text bold state
FirstIndent	(integer/owrite units 1/1000mm) Paragraph first line indent, offset from left indent
Font	(character/plain text) Text font name
FontSize	(Integer/points) Text size
Italic	(boolean/kTrue or kFalse) Text italic state
Justify	(integer/kWriStyParaJst… constant) Paragraph justification
LeftIndent	(integer/owrite units 1/1000mm) Paragraph left indent from page margin
LineSpacing	(integer/0 to 2) Paragraph line spacing 0=normal, 1 = 1.5, 2= double
RightIndent	(integer/owrite units 1/1000mm) Paragraph right indent from page margin
SpaceAfter	(integer/points) Paragraph spacing top
SpaceBefore	(integer/points) Paragraph spacing bottom
StyleSuper	(character/plain text) Paragraph style name
Tabs	(character/plain text) Paragraph tabs specified as a list of tab codes followed by tab position in OWrite units 1/1000mm separated by a space
TextColor	(integer/colour constant or RGB value) Text colour

Underline	(boolean/kTrue or kFalse) Text underline state

Table Sample Code

```
;  prep params for inserting a table object
Do $cinst.$objs.OWrite.$initparams(kWriObjTypeTable) Returns params

;  set main table properties
Do $cinst.$objs.OWrite.$addparams(params,"ObjName","table1",
              "ObjTableCellSpacing",2000,
              "ObjTableFlags",'PageHeaders')

;  set TABLE calculation that populates and returns an Omnis list
Do $cinst.$objs.OWrite.$addparams(params,
              "ObjCalc","$cinst.$getDataList('fClients', 'fClients.seq,
              fClients.title,
              fClients.name, fClients.address1, fClients.address2,
              fClients.city, fClients.state, fClients.postcode,
              fClients.country')")

;  add our HEADER row ($addparamstblrow returns the new row number)
Do $cinst.$objs.OWrite.$addparamstblrow(params,0,
              "ObjTableRowType",kWriTblRowHeader,
              "ObjTableRowFlags",'') Returns rowNo

;  using a for loop to insert FOUR CELLS for the HEADER row
For cellNo from 1 to 4 step 1

  ;  add cell basic properties for cell [cellNo]
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjWidth",pick(cellNo,0,15000,25000,50000,80000),
              "ObjHeight",5000,
              "ObjFlags",'') Returns cellNo

  ;  add cell border properties
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjBordStyle",kBorderPlain,
              "ObjBordLnStyle",kWriLineSolid,
              "ObjBordLnSize",100,
              "ObjBordColor",rgb(128,0,0),
              "ObjBorderSides",
                   kWriFrameLeft+kWriFrameRight+kWriFrameTop+kWriFrameBottom)

  ;  add cell fill and inner margins
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjFillColor",rgb(200,200,200),
              "ObjTopMargin",0,
              "ObjLeftMargin",1000,
              "ObjBottomMargin",0,
              "ObjRightMargin",1000) Returns cellNo

  ;  add cell content
  Do $cinst.$objs.OWrite.$addparamstblcellcontent(params,rowNo,cellNo,
              pick(cellNo, "", "Seq", "Title", "Name", "Address"),
              "StyleSuper","Normal",
              "Bold",kTrue,
              "TextColor",rgb(0,128,0),
              "SpaceBefore",0,
              "SpaceAfter",0)
```

```
End For

;  using a for loop to insert FOUR CELLS for the NORMAL/DATA row
For cellNo from 1 to 4 step 1

  ;  add cell basic properties for cell [cellNo]
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjWidth",pick(cellNo,0,15000,25000,50000,80000),
              "ObjHeight",8000,
              "ObjFlags",'AutoSize') Returns cellNo

  ;  add cell border properties
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjBordStyle",kBorderPlain,
              "ObjBordLnStyle",kWriLineSolid,
              "ObjBordLnSize",100,
              "ObjBordColor",rgb(128,0,0),
              "ObjBorderSides",
                   kWriFrameLeft+kWriFrameRight+kWriFrameTop+kWriFrameBottom)

  ;  add cell fill and inner margins
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjFillColor",rgb(200,200,200),
              "ObjTopMargin",0,
              "ObjLeftMargin",1000,
              "ObjBottomMargin",0,
              "ObjRightMargin",1000) Returns cellNo

  ;  add cell calculation info
  Do $cinst.$objs.OWrite.$addparamstblcell(params,rowNo,cellNo,
              "ObjName",pick(cellNo,"","Seq","Title","Name","Address"),
              "ObjCalc",pick(cellNo,"","$ref.seq","$ref.title",
                   "$ref.name", "con($ref.address1,chr(13),$ref.city,chr(13),
                   $ref.state,chr(13),$ref.postcode,chr(13),$ref.country)"),
              "ObjDisplay",pick(cellNo,"","Seq","Title","Name","Address"),
              "ObjToolTip",objTooltip) Returns cellNo

  ;  add cell content
  Do $cinst.$objs.OWrite.$addparamstblcellcontent(params,rowNo,cellNo,
              pick(cellNo, "", "Seq", "Title", "Name", "Address"),
              "StyleSuper","Normal",
              "Bold",kFalse,
              "TextColor",rgb(0,0,0),
              "SpaceBefore",0,
              "SpaceAfter",0)

End For

;  insert the object at current position
;  (last two parameters are async message and data for evAsyncDone event)
Do $cinst.$objs.OWrite.$insert(kWriObjTypeTable,params,kWriInsertOver,
              "Insert",kWriObjTypeTable) Returns ok
Quit method ok
```

Object Flags

This section lists and describes the flags that can be used with the ObjFlags, ObjTableRowFlags and ObjTableFlags properties as detailed above.

Generic Object Flags	
AutoSize	Object will grow vertically to fit the text or image.
GenClick	Object generates click events when clicked
LockAspect	Locks the aspect ratio when sizing picture fields
NoEnter	Cell cannot be entered and edited
ShowRTF	Instead of showing result data as formatted text, its RTF source is revealed.
StripEmpty	Empty lines are stripped from result data
UseClickCalc	Use click calculation (instead of object calculation) when object is clicked

Table Object Flags	
AutoExtend	New rows will be added when the user tabs off the last cell
MultiDataRows	Multiple normal rows will be used to display a single line from a result list
PageFooters	Table will generate page footers during evaluation
PageHeaders	Table will generate page headers during evaluation

Table Row Flags	
CanSplit	Table rows can be split across page boundaries during evaluation

Document History

08 April 2019: gramatical corrections
22 February 2019: first publication