Bug Fixes

ID : 1237 **Fixed in version :** 3.5.0

Short Description: owrite export to html replacing '&' with '&'

Full Description: I think this is a change to behaviour as things that work are now not. My call to get the html out of owrite is

Do
iOWrite.$savedata(binary,kWriFmtHTML,kTrue,kWriHtmlNoMargins,kTrue,kWriHtmlSizeAdjust,kFalse,kWriHtmlRawSupport,kTrue,kWriHtmlNoAutoSize,kTrue,kWriHtmlBgColor,iOWrite.$papercolor)

all the '&' in the exported 'html' seems to be replaced by &

for example, the .Header_&_Footer style is .Header_&_Footer and that may or may not affect the page rendering... I don't know

.Header_&_Footer\t{text-align:left;margin-top:0;padding-top:0pt;margin-bottom:0;padding-bottom:0pt;font-family:Times New Roman;font-size:10pt;font-style:normal;font-weight:normal;text-decoration:none;vertical-align:top;vertical-align:top;color:#000000;text-decoration:none;}\r\n\t.Header_&_Footer_table

the more important part is my rendering of pictures. I am creating a URL that is:
https://127.0.0.1/TheatreManager/1/picture?id=2&pmb_seq=217&tick=2278715

and it is coming out of the owrite exporter with the the '&' replaced by '&' as follows:

<img src="https://127.0.0.1/TheatreManager/1/picture?id=2&pmb_seq=217&tick=2278715
etc..

the problem with the & in this case is that it makes the parameters that follow in the URL ineffective .... the browsers and our servers won't process the parameters where the & is. I can fix it temporarily by replacing the & after the export...

but I'm not sure that a global change that the html export function is doing is wise in this case.

Comments : OWrite has always converted the '&' to the html "&" escape sequence within an image's SRC attribute (we have tested this as far back as version 2.7) . This was probably an oversight but it appears that all browsers and servers handle this correctly so it was never noticed. As it is not necessary to convert '&' characters when correctly used to append parameters to an image's URL we have changed this behaviour.

IMPORTANT: The URL of an image that is provided during export to HTML, must be valid URL and the '?' and '&' characters should only be used to append additional parameters to an URL.

NOTE: This change was previously reported to have been included in the 3.3.0 release, but this is incorrect. The change was added to a 3.2.5 patch but was missing from the 3.3.0 release. It has now been added to the 3.5.0 release.

ID :	1274	**Fixed in version :**	3.3.5
Short Description:	Fonts with style 'Normal' do not render correctly		

Full Description: Fonts with the style 'Normal' set do not render correctly, i.e. the text does not look right and there are large gaps between the words.

Comments : OWrite failed to map the style name 'Normal' (typically 'Regular' would be used) to the regular font of the specified family.

ID :	1277	**Fixed in version :**	3.3.6
Short Description:	Crash on load Binary		

Full Description: We have a problem with the last version of oWrite non-visual object.
We need to convert some text into an oWrite binary field.
We use the non visual oWrite object to load the text with the $loaddate(text, kwriftmtext) instruction and then saving the result on the binary variable calling the $savedata method.
These operations are correct, but when I try to reload the binary variable saved into the oWrite field, Omnis crashes.

In the attached example library, you can find exactly what I've described.
In the toolbar there is a button "text and nv owrite object" that invokes the method $nv_obj_owrite on oWrite field. As you can see, the last $loaddata unexpectedly quits the program.

The problem occurs both on WIN (seven and 8), I'm using Omnis 5.1.

Comments : This crash was caused by an empty font name in the 'Normal' style of the document. When one loads plain text data into a non-visual OWrite object, OWrite will create a document with a default 'Normal' style and a default 'Header & Footer' style if headers & footers are turned on.

Both styles initialise their font setting from the Omnis default System font. It appears that in Studio 5.1 Omnis returns an empty font name in some circumstances. Our testing indicates that this is not the case in all versions of Omnis and is intermittent, although we have been unable to specifically pin down under what circumstances Omnis returns an empty font name.

Normally this does not matter as OWrite considers loading data that may contain an empty font name. However, version 3.0 released in 2012 introduced major changes to font handling and we discovered a small section of the new font handling code that was not testing for empty font names. This has now been rectified and empty font names should no longer cause OWrite to crash.

However, being left with documents that contain invalid font settings in their default styles is not desirable and we recommend the following options to create valid documents from text data.

Option 1: load a template document containing a default set of styles and insert the plain text (recommended option)

```
Do OWriteObj.$loaddata(template_doc,kWriFmtDefault)
Do OWriteObj.$::insert(plain_text_data,kWriObjTypeText)
```

Option 2:load plain text data and modify the default styles

```
Do OWriteObj.$loaddata(plain_text_data,kWriFmtText)
Do OWriteObj.$getstylelist(list)
For list.$line from 1 to list.$linecount step 1
   Do OWriteObj.$editstylename.$assign(list.C1)
   Do OWriteObj.$curfont.$assign('Arial')
   Do OWriteObj.$curfontsize.$assign(10)
End For
Do OWriteObj.$editstylename.$assign('')
```

ID :	1278	**Fixed in version :**	3.5.0
Short Description:	Bookmarks not exported to HTML		

Full Description: Bookmarks added via the $curbookmark property appear not to be exported to HTML. The bookmark text does not appear to have a name tag surrounding it as in the text.

Comments :

ID : 1272 **Implemented in version :** 3.3.5

Short Description: $saveData command

Full Description: When we merge oWrite documents, with Omnis 5.1 on Windows, we usually get documents of 100 MB.
The $saveData command, crashes the program with the "insufficient memory" message (see the attached png) when the function is called with the kWriFmtDefault and kfalse as second and third parameter.

If the function is called with the kWriFmtDefault and ktrue parameters, the document loading is correct (albeit with very high and unacceptable waiting times).

Comments : In response to this report we have made a small change to the loading of data which can substantially increase performance during a $loaddata call and we have redesigned our document merge function in order to optimize the merging process and to rectify some anomalies such as unnecessary style creation and memory issues.

The third parameter for $mergedocs used to be a boolean parameter that simply specified if the documents should be separated by a page break. This parameter has been altered to an integer with a set of constants. The full set of supported constants are

```
kWriMergePageBreak
kWriMergeNoPagenate
kWriMergeUseExistingStyles
```

In order to merge many documents fast, it is crucial to avoid the unnecessary loading and saving of intermediate documents. The best approach is to use a non-visual OWrite document object to accumulate all documents to be merged. The first document can be loaded via the $loaddata method whereas all subsequent documents can be merged into the object using the non-static $mergedocs function.

Correct use of optimised merge method:

```
Do OWriteObj.$loaddata(doc_list.1.doc)
For doc_list.$line from 2 to doc_list.$linecount step 1
   Do OWriteObj.$mergedocs(#NULL,doc_list.doc,kWriMergeNoPagenate) Returns result
End for
Do OWriteObj.$savedata(merged_doc,kWriFmtDefault,kTrue)
```

In order for the merge to be fast, the third parameter must specify the new merge option kWriMergeNoPagenate. This stops OWrite form carrying out any time-costly pagination on the loaded data. Pagination is not required if all we are doing is merge a number of documents. However, it is impossible to further manipulate an un-paginated document, i.e. set the selection and change some content. The only valid action after all the documents have been merged is to save the document to the OWrite binary format as shown in the example above (please note, specifying kTrue for the third parameter, although not required, will make saving and loading of the merged document faster and will reduce the binary size). If further manipulation is required after a merge, the merged document can be saved and reloaded after which it can be manipulated normally. Another solution is to not specify the kWriMergeNoPagenate option for the last document that is to be merged.

In addition to the new merge option kWriMergeNoPagenate, we have also added the option kWriMergeUseExistingStyles. Without this option, when OWrite encounters two identically named styles during a merge which have different settings, OWrite renames the style being merged to Stylename~2, or Stylename~3, if 2 has been used already with this name, etc. However, when specifying kWriMergeUseExistingStyles, the conflicting style being merged is discarded and the existing style is used for the document content being merged. This may alter the appearance of documents that are merged but it is a neat way of bringing documents in-line with a common set of

styles.

Example:

```
   Do OWriteObj.$loaddata(empty_style_template_doc)
   For doc_list.$line from 1 to doc_list.$linecount step 1
      Do
OWriteObj.$mergedocs(#NULL,doc_list.doc,kWriMergeNoPagenate+kWriMergeUseExistingStyles)
Returns result
   End for
   Do OWriteObj.$savedata(merged_doc,kWriFmtDefault,kTrue)
```

The above example demonstrates how one can bring in line documents that are merged using a style template that contains a default set of styles. But the merge feature could also be used to bring individual documents in line for editing as part of the loading process.

Example

```
   Do OWriteWindowObj.$loaddata(empty_style_template_doc)
   Do
OWriteWindowObj.$mergedocs(#NULL,document_to_be_edited,kWriMergeUseExistingStyles)
```

In this case kWriMergeNoPagenate must not be specified so the document is ready for editing.

ID :	1275	**Implemented in version :**	3.4.0
Short Description:	Feedback while handling large documents		

Full Description: We have introduced the property $showprogress. If set to kTrue, OWrite will generate progress messages during time consuming loading, saving, importing, exporting or printing.

For the OWrite window object, the $event method is called with the new evProgress event. For non-visual document objects, the document object has to be inherited using an Omnis object class containing a method called $progress.

Both the event and method call will receive two parameters.
1) pProgressType - set to one of the new kWriProgress... constants
2) pProgressPercent - the percent complete value (0 to 100).

The following progress types are currently generated
   kWriProgressLoad: generated during $loaddata when loading with kWriFmtDefault
   kWriProgressSave: generated during $savedata when saving with kWriFmtDefault
   kWriProgressImport: generated during $loaddata when loading other than kWriFmtDefault
   kWriProgressExport: generated during $savedata when saving other than kWriFmtDefault
   kWriProgressPagenate: typically follows ...Load or ...Import once the document is loaded and is being formatted
   kWriProgressPrint: generated during printing of documents
   kWriProgressFinish: sent when the current process is completed
   kWriProgressCancel: sent when the current process did not complete due to an error

Example use:

```
   On evProgress
      ;  show progress of current operation (loading, saving, importing, exporting or printing)
      ;  this is helpful when dealing with larger documents (100 pages+)
      Switch pProgressType
         Case kWriProgressLoad
            Calculate $cinst.$statusbar.$panes.1.$text as "Loading document"
         Case kWriProgressSave
            Calculate $cinst.$statusbar.$panes.1.$text as "Saving document"
         Case kWriProgressImport
```

```
            Calculate $cinst.$statusbar.$panes.1.$text as "Importing document"
         Case kWriProgressExport
            Calculate $cinst.$statusbar.$panes.1.$text as "Exporting document"
         Case kWriProgressPrint
            Calculate $cinst.$statusbar.$panes.1.$text as "Printing"
         Case kWriProgressFinish
            ;  return to showing document info
            Calculate $cinst.$statusbar.$panes.1.$text as con("Page ",ivEdit.$pagenumber," of
",ivEdit.$::pagecount)
            Calculate $cinst.$statusbar.$panes.2.$text as "
         Case kWriProgressCancel
            Calculate $cinst.$statusbar.$panes.1.$text as "Action cancelled"
            Calculate $cinst.$statusbar.$panes.2.$text as "
            Quit event handler
      End Switch
      Calculate $cinst.$statusbar.$panes.2.$text as con(pProgressPercent,"% done
",jst("",con(pProgressPercent,"P|")))
```

Comments :