OWrite Expelled Gremlins

ID : 1935 **Fixed in version :** 5.3.0.0

Short Description: Assigning $papercontinuous crashes with permanent data documents

Full Description: In fact it is not an RTF document that causes me problem now but the original owr when I use $papercontinous to recalculate the pages and page numbers.
Please find attached the document.
When I load it, no problem.
But as soon as I do
Calculate iRefEdit.$papercontinuous as kTrue
Calculate iRefEdit.$papercontinuous as kFalse
To ensure the fields that may be in the document are evaluated and that pages are Ok, The system hangs and I must quit the app.
The doc has only the page and pages fields and (I suppose) no information to re-calculate the contents, so I deduct it should do nothing…
Is there a way to know if the document was saved with bMakeDataPermanent=ktrue so I could know that I must avoid using $papercontiuous ?

The client that reported this to me has a large number of documents that were saved and cannot be opened again.
On my PC, when I create a new document, I have no problem to open it again and do the re-pagination,
but on his system, even docs generated with last 5.2.0.0 and at least 2 pages hang when re-paginated.

Could you have a look at the problem and give me a hint ?

Comments : This was a difficult issue in regards to what the correct solution is.

When a document is saved with the parameter bMakeDataPermanent set to true, OWrite removes all calculations, result data and other evaluation related information, such as markers that show a table header row was inserted around page boundaries by using the $papercontinuous feature. Any result data is thus directly embedded in the document as if it was manually created by a user. For all intents and purposes, the document can no longer be evaluated and thus changed by assigning $evalcalcs.

Problem 1 ($evalcalcs)
==========
The first problem was that, because the current state of $evalcalcs is saved with the document, it will be true when the document is saved after evaluation and is set to true again when loaded. This allowed the $papercontinuous feature to work on the 'permanent' document.

We have now changed this so that when bMakeDataPermanent during a save is true, $evalcalcs is saved as false with the document, which makes sense, because the document is no longer in an evaluated state. The document is considered to be in a permanently data embedded state.

Problem 2 ($papercontinuous)
==========
The second problem was that this does not prevent existing documents that are in a confused state from crashing OWrite. The crash was caused by tables that had extra headers or footers embedded around page boundaries. Because the document was saved with bMakeDataPermanent set to true, the $papercontinuous feature failed to remove these extra headers or footers when entering continuous mode, which is vital for the operations that follow when exiting the continuous mode. These subsequent operations rely on the table only having one set of headers and footers at the beginning and end of the table. For all intents and purposes, the previously evaluated table has the appearance of a static table (manually entered by a user), which causes the crash. In any case, when the document was saved with bMakeDataPermanent set to true, $papercontinuous (even if it could remove the extra headers and footers) has no way of correctly re-inserting the page boundary headers and footers as all the result data (that these rows may have been relying on) is now missing from the document.

The solution was to add a simple test when assigning $papercontinuous, that checks the suitability of all tables in the document. If the order of header rows, data rows and footer rows is not as expected, the assignment will fail and the new warning kWriWarnPermanentData is added to the $docwarnings list. In your case, the generic code that re-applies page boundaries after loading a document can do

```
If (owrite_ref.$papercontinuous.$assign(kTrue))
    Do owrite_ref.$papercontinuous.$assign(kFalse)
End if
```

Problem 3 (knowing document is permanent)
==========
While investigating this issue, it became apparent there was no way of knowing if the document was saved with bMakeDataPermanent set to true. We have now added the read-only property $evalpermanent. This will return true if a loaded document was last saved with bMakeDataPermanent and $evalcalcs set to true.

Revised modifications (September 2022)
==========
Since the initial fix in May (version 5.2.1.0), we have slightly modified the behaviour as it was not in line with the behaviour of bMakeDataPermanent. In a nutshell, bMakeDataPermanent traditionally only affected the saved data. Unfortunately, our initial changes to resolve this issue affected also the loaded data by altering the $evalcalcs and $evalpermanent flags of the loaded document that is being saved. This is incorrect behaviour. We have now changed the code so that $evalcalcs and $evalpermanent only affects the saved data. This means that when saving a document $evalcalcs and $evalpermanent of the loaded document are not manipulated. Of course, when loading the document data of a document that was saved with bMakeDataPermanent, $evalcalcs and $evalpermanent of the newly loaded document will reflect this (i.e. $evalcalcs will be false and $evalpermanent will be true).

ID : 1940 **Fixed in version :** 5.2.1.2

Short Description: Omnis crashes during undo after merging horizontal cells

Full Description: With the example library (OWriteDocumentManager.lbs) :

1) create a new document
2) insert a new table (2 rows, 5 columns)
3) select the first and second column from the first row
4) right click on this selection and select "Table Options" / "Merge selected cells"
5) click on "Undo"
6) Omnis Studio crashes !

Comments : Resolved...

ID : 1941 **Fixed in version :** 5.2.1.2

Short Description: Omnis crashes when setting $papercontinuous to kFalse

Full Description: One of our clients reports a crash when creating a document. It occurs after evaluation when setting $papercontinuous to kfalse.
The doc has 2 tables. the 1st one spreads from page 1 to page 2 and 2nd table spreads from page 2 to page 3. Both tables have the property set to repeat the header on each page. (that's probably the problem...)

I have modified your example library to add button "repaginate" to the toolbar with the code you supplied for case 1935:

```
If (owrite_ref.$papercontinuous.$assign(kTrue))
    Do owrite_ref.$papercontinuous.$assign(kFalse)
End if
```

Import Document-5211-2tables.owr (evaluated doc built with oWrite 5211) and repaginate. Omnis

crashes.

Thank you for a fix.
Best regards
Joel

AttachedFile: Doc-OW5211-2Tables.zip

Comments : The crash can occur when paginating a table that has the properties $curtblpagefooters and $curtblpageheaders turned off, while the property $curtblrowevalcansplit is true for the data rows and a row can split across a page boundary. This can occur for both static tables and evaluated tables.

This crash does not appear to be reproducible on macOS using the provided example, but this could simply mean that it is intermittent in that it is dependent on circumstances such as the rows just fitted on the pages and did not require splitting.

ID : 1945 **Fixed in version :** 5.2.1.3

Short Description: oWrite crashes during startup

Full Description: I am trying to launch our Notarised version of Studio on a new M1 iMac. It has never had any other software running yet, out of the box.

Use our DMG file to copy the file, launch it, and it crashes. Both the Dev & the RT versions.

From Omnis Support:
'That crashed inside OWrite xcomp when it called GDIgetFontList(), I cannot see that function in our SDK so I assume it is Michael's function?'

I have tracked it down to oWrite R5.2.0 and if I remove that, all is well. Can you investigate please.

The rest of the team have no issues when they upgraded from older versions. I had some problems notarising this package, but it seems to work, but not on new iMacs. Does the crash log help at all?

Errors:
The provided email is linked to a maintenance only subscription. The specified request is not supported with this subscription.

Comments : This crash was the result of the system providing a postscript name of a font that does not appear to return a valid Family name resulting in a null string being returned. This was not expected and caused a crash in a subsequent system call.

ID : 1961 **Fixed in version :** 5.2.1.3

Short Description: Omnis crashes as soon as opened

Full Description: Refer to attached crash report. This is happening for a client who is on a Mac M1 processor. As oWrite is mentioned in the crash we're wondering if this is happening due to something with oWrite

Comments :

ID :	1962	**Fixed in version :**	5.3.0.0
Short Description:	oWrite crashes when using $holdupdates and $evalcalcs with non-visual object		

Full Description: if you try to eval calcs in a non-visual object it will crash Omnis if it is set to $holdupdates = true.

try the sample lib attached, i stripped down the code from our main app to investigate

Click the first button to load the attached sample binary with a calc field, then the next buttons will prep and run OK in the visual world, followed by non-visual. if you choose 9a it will crash, if you choose 9b it will work...

Took a while to find this out! I have a work around so not urgent

AttachedFile: Archive.zip

Comments :

ID :	1963	**Fixed in version :**	5.3.0.0
Short Description:	Omnis crashes when setting $papercontinuous to kFalse		

Full Description: while I had made sure the client runs oWrite 5.2.1.2 it seems there is still cases when pagination crashes Omnis.
The doc is built from same template as in case 1941, where two tables expands over pages.
I attach the document before setting $papercontinuous to kfalse.
My client runs under Omnis 8.1.6 win64 but I tested it with Omnis 10.2 with same result.

Thank you for a fix under both 8.1.6 and 10.2

AttachedFile: Document S029422.zip

Comments : There was a repagination problem when a the table headers of the table were separated from the first data row due to pagination. The special code that was meant to correct this prior to inserting additional headers and footers around new page breaks failed due to a logic problem.

ID :	1965	**Fixed in version :**	5.3.0.0
Short Description:	Notation with $:: no longer works in 10.2		

Full Description: When trying to code oWrite notation that uses notation names of Omnis internal notation, it used to be possible to do so by qualifying the notation with two colons as in $::insert(). Due to tokenization changes in more recent Studio 10.2 versions, this no longer works. Studio fails to tokenize such notation correctly resulting in a notation error when the code is executed. According to Omnis support, there is a work-around:

"Set the entry badNotationNameIsSyntaxError in the methodEditor section of config.json to false, to revert to the behaviour of earlier versions of 10.2."

This issue does not effect existing code as long as it is not edited with these latest versions.

However, to protect our developers when moving forward with Studio 10.2, we have made the difficult decision to rename the effected properties and methods, while maintaining the old names to protect existing code. We tried various other solutions, but they all proved unsatisfactory in different ways. Below is the list of property/method names and their replacements:

$paper -> $docpaper
$paperlength -> $docpaperlength
$paperwidth -> $docpaperwidth
$orientation -> $docorientation

$topmargin -> $doctopmargin
$leftmargin -> $docleftmargin
$bottommargin -> $docbottommargin
$rightmargin -> $docrightmargin
$pagecount -> $docpagecount
$pagenumber -> $docpagenumber
$showrulers -> $showpaperrulers
$insert() -> $docinsert()
$print() -> $docprint()

As already stated, existing code will continue to work, but the new notation names should be used from hereon if at all possible. If you choose to replace existing notation, be careful only to replace notation code that addresses oWrite.

Comments :