OWrite and Unicode

As developers consider moving to Studio 5 the move to unicode and its complications can no longer be avoided. OWrite has for a long time stored text within its binary format as UTF8, regardless of wether you are using Unicode or non-Unicode Omnis. So there are no conversion issues when moving to Studio 5.

However, there are complications that directly effect the way you code. When loading and saving data using the various formats supported by OWrite, these complications may arise.

When saving and loading RTF or HTML, you must now use binary variables. When saving and loading plain text as UTF-16 or UTF-8, you must also use binary variables.

These formats may have worked using text variables in non-Unicode Omnis, but they will not work in Unicode Omnis with text variables. A more detailed explanation follows.

RTF

When loading RTF documents, these are typically loaded from a disk file using file ops. RTF is encoded as 7 bit ASCII single byte characters, so when reading an RTF file, the data must now be read into an Omnis binary variable. Omnis text variables only store UTF-32 which are four byte characters. Reading RTF data into a text variable will corrupt the RTF.

Equally, when using $savedata to save RTF, this must be done to a binary variable, as OWrite will write out RTF as 7 bit ASCII single byte characters, which is the standard.

OWrite has some special code that will accept RTF as 4 byte characters when inserting text via calculations. This allows you to use Omnis text variables to build and store calculations for your OWrite fields that return RTF.

We are aware that some developers, against our recommendation, store their OWrite documents as RTF using a text field in their database. When converting your library and data files to Studio 5, this text will now be converted to UTF-32. In order to deal with this you must manually convert the RTF data to single byte characters when loading documents and to four byte characters when saving documents.

When loading, prior to calling $loaddata, use the Omnis function chartoutf8(). When saving after calling $savedata, use the function utf8tochar(). It is save to use the UTF-8 conversion functions as RTF only uses the low 7 bits of the single byte characters.

```
; load data example
Calculate binary_variable as chartoutf8(unicode_rtf_text)
Do OWriteRef.$loaddata(binary_variable,kWriFmtRTF)

; save data example
Do OWriteRef.$savedata(binary_variable,kWriFmtRTF)
Calculate unicode_rtf_text as utf8tochar(binary_variable)
```

May we add that we still do not recommend storing your master documents as RTF. If a text based version is required for searches, please consider storing both the OWrite binary format and a copy as plain text.

HTML

OWrite saves HTML in the two most commonly used encodings, ISO-8859-1 and UTF-8. It specifies the encoding in the HTML header via the charset attribute.

When running in a non-Unicode Omnis, OWrite will save HTML as ISO-8859-1, and when running in Unicode Omnis, OWrite will save HTML as UTF-8 or ISO-8859-1 if kWriSaveNonUnicode is specified.

Both these encodings are single byte character encodings and therefore are not compatible with text variables in Unicode Omnis. In a Unicode Omnis, HTML must be saved to a binary variable.

```
; save the default format ISO-8859-1 in non-unicode Omnis and UTF-8 in
; unicode Omnis
Do OWriteRef.$savedata(binary_variable,kWriFmtHTML)

; save as ISO-8859-1 in unicode Omnis
Do OWriteRef.$savedata(binary_variable,kWriFmtHTML,kFalse,kWriSaveNonUnicode,kTrue)
```

TEXT

When loading or saving plain text in a Unicode Omnis, the default is UTF-32. Additionally, OWrite can load and save UTF-16, UTF-8 and operating system characters, by specifying kWriTextFmtUTF16, kWriTextFmtUTF8 and kWriSaveNonUnicode in the $savedata and $loaddata methods. Please refer to the main documentation for a full description. Needless to say, when choosing these alternative formats, binary variables must be used for storage.

```
; in unicode Omnis you can save as UTF-32, UTF-16, UTF-8
; and operating system characters

; save as UTF-32
Do OWriteRef.$savedata(text_variable,kWriFmtText)
; save as UTF-16
Do OWriteRef.$savedata(binary_variable,kWriFmtText,kFalse,kWriTextFmtUTF16,kTrue)
; save as UTF-8
Do OWriteRef.$savedata(binary_variable,kWriFmtText,kFalse,kWriTextFmtUTF8,kTrue)
; save as platform character set
Do OWriteRef.$savedata(binary_variable,kWriFmtText,kFalse,kWriSaveNonUnicode,kTrue)
```