Convert Office Files in FileMaker

We have a few customers with document databases. They store Word, Excel, PowerPoint and other documents in a FileMaker database. To preview them in a container field, we need for all documents a preview picture. JPEG, PNG and few other image formats can be shown directly. But those office documents need to be converted and that needs an application to do layouts and page break.

That is where we use LibreOffice. It allows us to convert all the given file types:

- Microsoft Word document
- Microsoft Excel document
- Microsoft PowerPoint document
- Open Document documents
- RTF files
- HTML files

The HTML thing surprised me, but LibreOffice seems to have a way to import a HTML document into a LibreOffice text document. And then we can ask it to export it to a PDF, just like the others.

We got this example script for FileMaker in our PDFKit Print Documents.fmp12 example file:

 # Convert Office File in file PDFKit Print Documents

```
Set Variable [ $DocumentPath ; Value: GetAsText(PDFKit Print Documents::Document) ]
Set Variable [ $DocumentName ; Value: MBS( "Path.LastPathComponent"; $DocumentPath ) ]
# temp position for input file
Set Variable [ $TempFolder ; Value: MBS( "Folders.UserTemporary" ) ]
Set Variable [ $InputFolder ; Value: MBS( "Path.AddPathComponent"; $TempFolder; "Input-" & Get(UUID) ) ]
Set Variable [ $InputFile ; Value: MBS( "Path.AddPathComponent"; $InputFolder; $DocumentName ) ]
Set Variable [ $r ; Value: MBS( "Files.CreateDirectory"; $InputFolder ) ]
# and output
Set Variable [ $OutputFolder ; Value: MBS( "Path.AddPathComponent"; $TempFolder; "Output" & Get(UUID) ) ]
Set Variable [ $r ; Value: MBS( "Files.CreateDirectory"; $OutputFolder ) ]
#
# write input file
```

```
Set Variable [ $result ; Value: MBS( "Container.WriteFile"; PDFKit Print
Documents::Document; $InputFile) ]
If [ MBS("IsError") ]
	Exit Script [ Text Result: $result ]
End If
#
If [ MBS("IsWindows") ]
	# using LibreOffice on Windows
	Set Variable [ $AppPath ; Value: "C:\Program
Files\LibreOffice\program\soffice.exe" ]
Else If [ MBS("IsMacOS") ]
	# using LibreOffice on MacOS
	Set Variable [ $AppPath ; Value: "/Applications/LibreOffice.app/Contents/
MacOS/soffice" ]
Else If [ MBS("IsLinux") ]
	# using LibreOffice on Linux
	Set Variable [ $AppPath ; Value: "/usr/bin/soffice" ]
Else
	# cleanup & exit
	Set Variable [ $r ; Value: MBS( "Files.Delete"; $InputFile ) ]
	Set Variable [ $r ; Value: MBS( "Files.DeleteFolder"; $OutputFolder ) ]
	Set Variable [ $r ; Value: MBS( "Files.DeleteFolder"; $InputFolder ) ]
	Exit Script [ Text Result: "Platform not supported." ]
End If
#
Set Variable [ $shell ; Value: MBS( "Shell.New" ) ]
Set Variable [ $result ; Value: MBS( "Shell.Execute"; $shell; $appPath; "--convert-
to"; "pdf"; "--outdir"; $OutputFolder; $InputFile ) ]
If [ MBS("IsError") ]
	Set Variable [ $e ; Value: MBS("Shell.Release"; $shell) ]
Else
	# running
	Set Variable [ $e ; Value: MBS( "Shell.Wait"; $shell; 10) ]
	Set Variable [ $error ; Value: MBS( "Shell.ReadErrorText"; $shell; "UTF-8") ]
	Set Variable [ $output ; Value: MBS( "Shell.ReadOutputText"; $shell; "UTF-8")
]
	#
	Set Variable [ $files ; Value: MBS( "Files.List"; $outputFolder; 1; ".pdf" ) ]
	If [ ValueCount ( $files ) = 0 ]
		Set Variable [ $result ; Value: "Conversion failed: " & $error ]
	Else
		Set Variable [ $fileName ; Value: GetValue($files; 1) ]
		Set Variable [ $FilePath ; Value: MBS( "Path.AddPathComponent";
$outputFolder; $fileName ) ]
		#
```

```
                Set Variable [ $PDF ; Value: MBS( "Container.ReadFile"; $FilePath;
"auto"; $fileName) ]
                If [ MBS("IsError") ]
                        Set Variable [ $result ; Value: $PDF ]
                Else
                        Set Field [ PDFKit Print Documents::PDF ; $PDF ]
                        Set Variable [ $result ; Value: "OK" ]
                End If
        End If
End If
#
# cleanup
Set Variable [ $r ; Value: MBS( "Files.Delete"; $FilePath ) ]
Set Variable [ $r ; Value: MBS( "Files.Delete"; $InputFile ) ]
Set Variable [ $r ; Value: MBS( "Files.DeleteFolder"; $OutputFolder ) ]
Set Variable [ $r ; Value: MBS( "Files.DeleteFolder"; $InputFolder ) ]
#
Exit Script [ Text Result: $result ]
```

There are a few special things in that script. First we create a temp folder for both input and output using an UUID for each. This way we never have trouble with duplicate file names in those folders. And whatever file appears later in output file is our result. The use of UUIDs allow this to run in multiple scripts in parallel if needed. We use Container.WriteFile function to write the file into the input folder.

For the LibreOffice executable file, we need to have the full path. So on Windows we pick the exe in the program folder, for macOS the executable within the LibreOffice application. For Linux you install the right packages and then it is available in /usr/bin folder.

The Shell.Execute call passes each parameter as a parameter to the plugin, so the quoting is done right internally. Then the whole thing runs in the background and we wait up to 10 seconds with Shell.Wait function. On return we read any error messages and then look for the PDF file in the output folder.

Finally we have to do some cleanup with deleting temp files. In case the process would crash, you may just not get the PDF.

Please do not hesitate to contact us if you have questions.