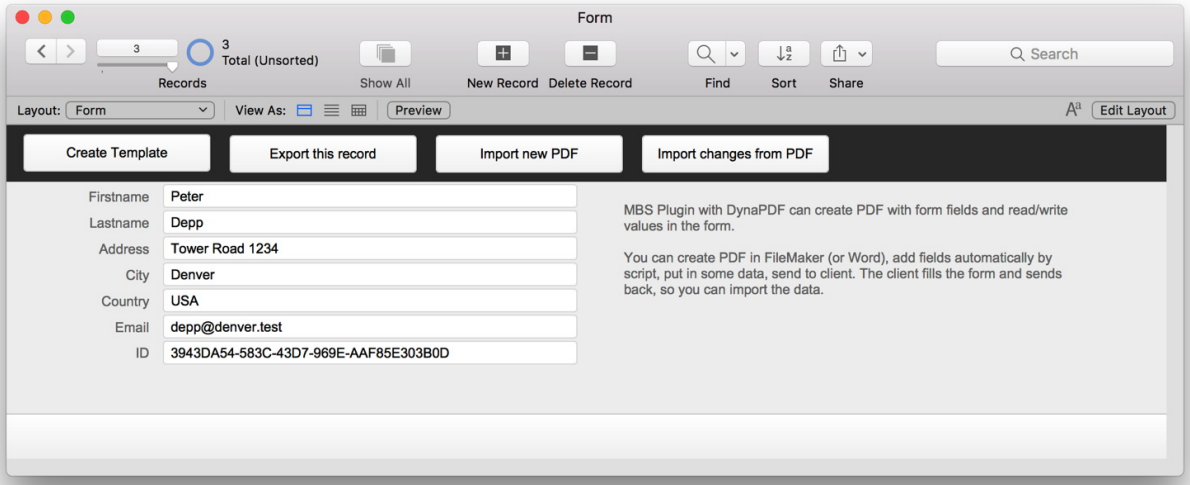
PDF Forms with MBS Plugin

As you may know, you can create PDF form fields with our MBS Plugin using the DynaPDF functions. The plugin can do a lot there and not just read/write form field values, but also create new fields. In the following article, we want to show you how we do this. Our goal is to dynamically create a template PDF based on a form file created in a word processor. We add fields with data, send it to a client and when we get the modified file back, we can import the new values.



We start with a sample database. This database has a table for people and the following fields. All just plain text fields and the ID field is set to be come an UUID. That is important as a random UUID makes it difficult to guess a valid ID, which would be easy if you just count up record IDs. We store the ID in the PDF and don't want someone to make up a valid ID. Here is the field list:

Firstname	Text	
Lastname	Text	
Address	Text	
City	Text	
Country	Text	
Email	Text	
ID	Text	Indexed, Auto-enter Calculation replaces existing value, Can't Modify Auto

DynaPDF Initialize

First thing we do in all scripts is to trigger the DynaPDF Initialize script if needed.

```
# Initialize DynaPDF if needed
If [ MBS("DynaPDF.IsInitialized") ≠ 1 ]
        Perform Script [ “InitDynaPDF” ]
End If
```

The DynaPDF initialization script more or less builds a file path for the dynapdf library file and calls DynaPDF.Initialize, so the plugin can load the library. As we use different library

files for Mac and Windows, you need to specify the right one. For a server, we usually put the file in the extension folder next to the plugin and just pass the file name, so the plugin locates the folder automatically.

Create fields

For our example we have an existing form.pdf file, which I just created in Pages and print as PDF. You can use whatever PDF file you like or just create one in FileMaker. First the script gets the path to the Form.pdf and builds our template.pdf path.

Next we start a new PDF session with DynaPDF.New and open the Form.pdf file using DynaPDF.OpenPDFFromFile function. Of course you an use our DynaPDF.OpenPDFFromContainer function to read from a container field. Next we call DynaPDF.ImportPDFFile to read all the pages. The DynaPDF.EditPage function opens the the first page for editing as we want to place the fields there. Next we define background and border colors for the new fields. Before we continue, here the script:

```
# Import form.pdf and add fields
#
Set Variable [ $FormPath ; Value: MBS( "Path.FileMakerPathToNativePath"; Substitute
( Get ( FilePath ); ".fmp12"; ".pdf")) ]
Set Variable [ $TemplatePath ; Value: Substitute ( $FormPath; "Form.pdf";
"Template.pdf") ]
#
# Clear current PDF document
Set Variable [ $pdf ; Value: MBS("DynaPDF.New") ]
# Load PDF from file
Set Variable [ $r ; Value: MBS("DynaPDF.OpenPDFFromFile"; $pdf; $FormPath) ]
If [ MBS("IsError") ]
	Show Custom Dialog [ "Failed" ; "Failed to read PDF file." & ¶ & $r ]
Else
	Set Variable [ $r ; Value: MBS("DynaPDF.ImportPDFFile"; $pdf) ]
	Set Variable [ $r ; Value: MBS("DynaPDF.EditPage"; $pdf; 1) ]
	Set Variable [ $r ; Value: MBS("DynaPDF.SetFieldBackColor"; $pdf;
MBS("DynaPDF.RGB"; 200; 200; 200)) ]
	Set Variable [ $r ; Value: MBS("DynaPDF.SetFieldBorderColor"; $pdf; 0 /* black */) ]
	#
	# We position fields where we find label text
	Set Variable [ $r ; Value: CreateTextField ( "Firstname" ; "Firstname:" ) ]
	Set Variable [ $r ; Value: CreateTextField ( "Lastname" ; "Lastname:" ) ]
	Set Variable [ $r ; Value: CreateTextField ( "Address" ; "Address:" ) ]
	Set Variable [ $r ; Value: CreateTextField ( "City" ; "City:" ) ]
	Set Variable [ $r ; Value: CreateTextField ( "Country" ; "Country:" ) ]
	Set Variable [ $r ; Value: CreateTextField ( "Email" ; "Email:" ) ]
	Set Variable [ $r ; Value: Let ( [ fieldName = "ID"; field =
MBS( "DynaPDF.CreateTextField"; $pdf; fieldName; -1; 0; 200; 120; 10; 300; 20 ); r =
MBS("DynaPDF.SetFieldFlags"; $pdf; field; "Hidden ReadOnly Invisible NoView") /* yes,
three ways to hide! */ ]; field ) ]
	#
	Set Variable [ $r ; Value: MBS("DynaPDF.EndPage"; $pdf) ]
	Set Variable [ $r ; Value: MBS("DynaPDF.OpenOutputFIle"; $pdf; $templatePath) ]
	If [ MBS("IsError") ]
```

```
                Show Custom Dialog [ "Failed" ; "Failed to create PDF file." & ¶ & $r ]
        End If
End If
Set Variable [ $r ; Value: MBS("DynaPDF.Release"; $pdf) ]
```

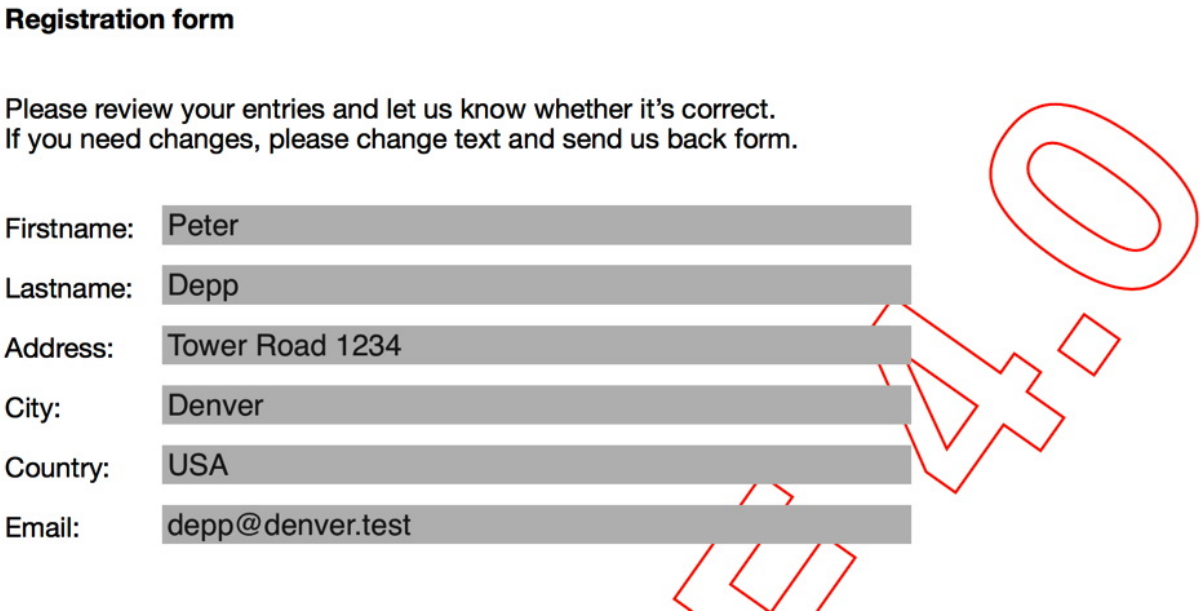
To create the fields, we use a pretty clever custom function:

```
CreateTextField( fieldName, findText )
Let (
[
found = MBS( "DynaPDF.FindText"; $pdf; fieldName; 0 );
posText1 = MiddleWords ( found ; 2 ; 1 );
posNum1 = MBS( "Math.TextToNumber"; posText1 );
posText2 = MiddleWords ( found ; 6 ; 1 );
posNum2 = MBS( "Math.TextToNumber"; posText2 );
field = MBS( "DynaPDF.CreateTextField"; $pdf; fieldName; -1; 0; 200; 120;
posNum1-3; 300; posNum2-posNum1+5 );
r = MBS("DynaPDF.SetFieldBorderStyle"; $pdf; field; "Bevelled")
]; field )
```

As you see, we first call DynaPDF.FindText function to locate the text of the label in the PDF page. We get a position back and pick the two y positions from the rectangle. As I test this in Germany and the numbers come in US format, I need to use Math.TextToNumber to convert from text to number. Once we have a position, we can simply place a field with the given name. Maximum text length is 200 here and position is 120 point from left and posNum1-3 points from bottom of page. 300 is the width of the field and the height is dynamically based on he text size of the label.

After we created the fields for the data, we continue in the script with an invisible field for the ID. There we use DynaPDF.SetFieldFlags to hide the field with 3 different flags. Yes, PDF standard defines three ways and we just use them all. Once all fields are completed, we close the page and than open an output file. Alternatively you can just call DynaPDF.Save and get the result in a container field.

Export PDF with data



The next script exports data from one record into a PDF file. We use FileDialog functions in MBS Plugin to ask for a file path, than import the template and create a new PDF file at the given file path. For each field, we simply call DynaPDF.SetTextFieldValue function and pass the field name and value. When the PDF is closed, the file is saved and you can email it to an user.

```
# Ask user for file location to save to
#
Set Variable [ $r ; Value: MBS("FileDialog.Reset") ]
Set Variable [ $r ; Value: MBS("FileDialog.SetMessage"; "Please select PDF file") ]
Set Variable [ $r ; Value: MBS( "FileDialog.SetNameFieldStringValue"; "Export" &
Get(RecordID ) & ".pdf") ]
Set Variable [ $r ; Value: MBS("FileDialog.SaveFileDialog") ]
If [ $r ≠ "OK" ]
        Exit Script [ Text Result:    ]
End If
#
Set Variable [ $OutputPath ; Value: MBS("FileDialog.GetPath"; 0) ]
Set Variable [ $TemplatePath ; Value: MBS( "Path.FileMakerPathToNativePath"; Substitute
( Get ( FilePath ); "Form.fmp12"; "Template.pdf")) ]
#
#
# Clear current PDF document
Set Variable [ $pdf ; Value: MBS("DynaPDF.New") ]
# Load PDF from file
Set Variable [ $r ; Value: MBS("DynaPDF.OpenPDFFromFile"; $pdf; $TemplatePath) ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Failed" ; "Failed to read PDF file." & ¶ & $r ]
Else
        Set Variable [ $r ; Value: MBS("DynaPDF.ImportPDFFile"; $pdf) ]
        Set Variable [ $r ; Value: MBS("DynaPDF.OpenOutputFile"; $pdf; $outputPath) ]
        If [ MBS("IsError") ]
                Show Custom Dialog [ "Failed" ; "Failed to create PDF file." & ¶ & $r ]
        End If
```

```
        # Write form values into form
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "Firstname";
Form::Firstname; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "Lastname";
Form::Lastname; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "Address";
Form::Address; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "City";
Form::City; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "Country";
Form::Country; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "Email";
Form::Email; ""; "left" ) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.SetTextFieldValue"; $pdf; "ID"; Form::ID;
""; "left" ) ]
End If
Set Variable [ $r ; Value: MBS("DynaPDF.Release"; $pdf) ]
```

Import updated data

When you get back the file, you can use the following script to import the data for an existing record. The script for creating a newer record is a bit easier and included in sample database.

The key here is that we use DynaPDF.GetField to get the text from the ID field. This could be anything as some user may have changed it. We use the text only for the search to find a record with a matching ID. If we find such a record, we update the values, otherwise we show an error message to the user:

```
# Ask user for file path
Set Variable [ $r ; Value: MBS("FileDialog.Reset") ]
Set Variable [ $r ; Value: MBS("FileDialog.SetMessage"; "Please select PDF file") ]
Set Variable [ $r ; Value: MBS("FileDialog.OpenFileDialog") ]
If [ $r ≠ "OK" ]
        Exit Script [ Text Result:    ]
End If
#
Set Variable [ $path ; Value: MBS("FileDialog.GetPath"; 0) ]
#
#
# Clear current PDF document
Set Variable [ $pdf ; Value: MBS("DynaPDF.New") ]
# Load PDF from file
Set Variable [ $r ; Value: MBS("DynaPDF.OpenPDFFromFile"; $pdf; $path) ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Failed" ; "Failed to read PDF file." & ¶ & $r ]
Else
        Set Variable [ $r ; Value: MBS("DynaPDF.ImportPDFFile"; $pdf) ]
        #
        # Look for record by ID
        Set Variable [ $ID ; Value: MBS( "DynaPDF.GetField"; $pdf; "ID"; "Value" ) ]
        Set Error Capture [ On ]
```

```
	# Find Records		Form::ID: [==$ID]
	Perform Find [ Restore ]
	If [ Get ( FoundCount ) = 1 ]
		# Update values
		Set Field [ Form::Firstname ; MBS( "DynaPDF.GetField"; $pdf; "Firstname";
"Value" ) ]
		Set Field [ Form::Lastname ; MBS( "DynaPDF.GetField"; $pdf; "Lastname";
"Value" ) ]
		Set Field [ Form::Address ; MBS( "DynaPDF.GetField"; $pdf; "Address";
"Value" ) ]
		Set Field [ Form::City ; MBS( "DynaPDF.GetField"; $pdf; "City"; "Value" ) ]
		Set Field [ Form::Country ; MBS( "DynaPDF.GetField"; $pdf; "Country";
"Value" ) ]
		Set Field [ Form::Email ; MBS( "DynaPDF.GetField"; $pdf; "Email";
"Value" ) ]
	Else
		Show Custom Dialog [ "Record not
found." ; MBS( "DynaPDF.GetField"; $pdf; "Firstname"; "Value" ) & "
" & MBS("DynaPDF.GetField"; $pdf; "Lastname"; "Value" ) & ¶ & $ID ]
	End If
	Show All Records
End If
Set Variable [ $r ; Value: MBS("DynaPDF.Release"; $pdf) ]
```

Conclusion

We hope you got the idea about how the form field functions work in our plugin. If needed, the plugin can create checkboxes with choice for picking one or multiple options, create popup menus and lists to pick values. You an have combo boxes, which provide text entry as well as picking predefined values from a list. DynaPDF can even add buttons with custom javascript actions, e.g. to verify user input in the Adobe Reader application. The group field allow to group several fields together in a form. Finally all can be customized to your style with fonts, colors and borders.

The example project Form.fmp12 is available with MBS Plugin 8.2. Or email us for a copy.