# Create Barcodes on Server from FileMaker Go

We recently got a query how to create the scripts to handle barcode creation everywhere with MBS Plugin. This includes passing the request to a FileMaker Server if no local plugin is installed:

- in FileMaker Pro with MBS Plugin to create barcode locally.
- in FileMaker Pro or Go without MBS Plugin and using the plugin installed on server
- in a scheduled script on FileMaker Server with MBS Plugin installed
- in WebDirect on FileMaker Server
- in FileMaker Data API on FileMaker Server, where we need to trigger script in server side scripting engine.
- in FileMaker iOS SDK with either local plugin or server side plugin

The following script runs on the FileMaker Server and creates the barcode there if the MBS Plugin is installed. Pass JSON as script parameter with the barcode parameters. The script then returns a JSON object text with either error or barcode entry filled. But see yourself:

```
# This script is run on server to provide barcode to clients without MBS Plugin
Set Variable [ $JSON ; Value: Get(ScriptParameter) ]
#
Set Variable [ $result ; Value: "" ]
If [ GetAsText(MBS("Version")) = "?" ]
	# no MBS Plugin
	Set Variable [ $result ; Value: JSONSetElement ( $JSON ; "error" ; "No MBS Plugin on server installed."; JSONString ) ]
Else
	# with MBS Plugin
	#
	# generate barcode based on JSON
	Set Variable [ $img ; Value: MBS("Barcode.GenerateJSON"; $JSON) ]
	If [ MBS("IsError") = 0 ]
		# Create PNG image file as container value
		Set Variable [ $Image ; Value: MBS( "GMImage.WriteToPNGContainer"; $img; "barcode.png") ]
		If [ MBS("IsError") = 0 ]
			# Return image base64 encoded
			Set Variable [ $result ; Value: JSONSetElement ( $JSON ; "barcode" ; Base64Encode ( $Image ); JSONString ) ]
```

```
                Else
                        # Pass back error
                        Set Variable [ $result ; Value: JSONSetElement ( $JSON ;
"error" ; $Image; JSONString ) ]
                End If
                # free memory
                Set Variable [ $r ; Value: MBS( "GMImage.Destroy"; $img ) ]
        Else
                # Pass back error
                Set Variable [ $result ; Value: JSONSetElement ( $JSON ; "error" ;
$img; JSONString ) ]
        End If
End If
Exit Script [ Text Result: $result ]
```

The script above is called with Perform Script on Server and below follows the script to run it. This script tests if MBS Plugin is installed locally and if the plugin is missing passes the request to the server. Otherwise it calls the above script directly to create barcode:

```
# This script is run locally in FileMaker Pro or Go to create barcode via server
Set Variable [ $JSON ; Value: Get(ScriptParameter) ]
Set Variable [ $result ; Value: "" ]
If [ GetAsText(MBS("Version")) = "?" ]
        # no MBS Plugin, so pass to server
        #
        Perform Script on Server [ Specified: From list ; “Create Barcode on
Server” ; Parameter: $JSON ; Wait for completion: On ]
        Set Variable [ $errorCode ; Value: Get(LastError) ]
        If [ $errorCode ≠ 0 ]
                # report error back to caller
                Set Variable [ $error ; Value: "Failed to trigger script on server with
error " & $errorCode ]
                Set Variable [ $result ; Value: JSONSetElement ( $JSON ; "error" ;
$error; JSONString ) ]
        Else
                # pass result back to caller
                Set Variable [ $result ; Value: Get(ScriptResult) ]
        End If
Else
        # with MBS Plugin
        #
        Perform Script [ Specified: From list ; “Create Barcode on Server” ;
Parameter: $JSON ]
        Set Variable [ $result ; Value: Get(ScriptResult) ]
```

```
End If
Exit Script [ Text Result: $result ]
```

Finally here the test script to generate the barcode and put it in the container field:

```
# Build JSON to call
// Set Variable [ $JSON ; Value: Barcode Generation JSON::JSON ]
Set Variable [ $JSON ; Value: JSONSetElement ( "{}" ; "symbology" ; "QRCode";
JSONString ) ]
Set Variable [ $JSON ; Value: JSONSetElement ( $JSON ; "Text" ; "Hello World";
JSONString ) ]
Set Variable [ $JSON ; Value: JSONSetElement ( $JSON ; "Scale" ; "4";
JSONNumber ) ]
#
# run script to create barcode
Perform Script [ Specified: From list ; “Create Barcode” ; Parameter: $JSON ]
Set Variable [ $value ; Value: Get(ScriptResult) ]
#
# check result
Set Variable [ $barcode ; Value: JSONGetElement ( $value ; "barcode" ) ]
If [ Length ( $barcode ) > 0 ]
        Set Field [ Barcode Generation JSON::Image ; Base64Decode ( $barcode;
"barcode.png" ) ]
Else
        Set Variable [ $error ; Value: JSONGetElement ( $value ; "error" ) ]
        Show Custom Dialog [ "Failed to create barcode" ; $error ]
End If
```

As you see we either take the JSON from field or for a test here build it with JSONSetElement. Then it calls the script and if this is successful, shows the barcode in an image container. If the script failed it shows the error message to the user. By passing result back as JSON we can easily distinguish between error and success states. And as with most JSON scripts we write, we pass back the input JSON in the result, so you would see the parameters passed when you get the error message in the JSON.