Batch sending Emails in FileMaker via MBS Plugin

As you may know you can use CURL functions in MBS Plugin to send emails. We include examples to show you how to send with attachments, html text and inline graphics. A recent example coming with 7.0pr2 showed you how to batch send emails. But that example sends emails one by one and each time with a new connection. I already improved the example here to reuse connections which helps a lot on speed. Still the big problem is that network transfers with uploads take time. The script waits while the Kilobytes for the email go through the network cables.

Background processing

Luckily our plugin offers a function for cURL to run a transfer in background: CURL.PerformInBackground. Using this function we can start a cURL transfer and the script continues. We can prepare and send the next email while the last one is still uploading. Now you need to be careful and manage the connections and their status. To show you how this can work, we show you what we did in our updated example. Here is an excerpt from the script where we look for a free cURL session:

```
...
#Find a non busy curl connection
Set Variable [$index; Value:1]
Loop
	If [$curls[$index] = ""]
		#found free index, setup new CURL session
		Set Variable [$curl; Value:MBS("CURL.New")]
		Set Variable [$curls[$index]; Value:$curl]
		Set Variable [$r; Value:MBS("CURL.SetOptionURL"; $curl; "smtp://" &
$SMTPServer)]
		Set Variable [$r; Value:MBS("CURL.SetOptionUsername"; $curl;
$SMTPUser)]
		Set Variable [$r; Value:MBS("CURL.SetOptionPassword"; $curl;
$SMTPPass)]
		...
		Exit Loop If [1]
	Else
		#found used index
		If [MBS( "CURL.IsRunning"; $curls[$index] ) = 0]
			#found used index which is done
			Perform Script [“HandleFinishedCURL”; Parameter:
$curls[$index]]
			Set Variable [$curl; Value:$curls[$index]]
			Exit Loop If [1]
		End If
	End If
	Set Variable [$index; Value:$index + 1]
	If [$index = 9]
		#Wait
		Set Variable [$index; Value:1]
		Pause/Resume Script [Duration (seconds): ,01]
	End If
End Loop
...
```

As you see we use $curls[] for the connections and have up to 8 of them. How many we actually use depends on network speed. The slower the network, the more connections keep running. So when we find an empty slot, we can create a new session and store it in the $curls array and use the new one. If an existing connection is not running, we can trigger the script HandleFinishedCURL handle that. Than we can reuse the cURL session. If no session is available ($index = 9), we wait a bit and loop again.

Now, we have a free cURL session. We use the tag on the cURL session to store the ID for the current record. Tags are often used in MBS Plugins to associate a value with a given object. Later in the HandleFinishedCURL we can get the ID back and know which record needs updating. Next we call CURL.PerformInBackground to start transfer in a background thread:

```
...
Set Variable [$result; Value:MBS("CURL.SetTag"; $curl; Recipients::ID)]
Set Variable [$result; Value:MBS("CURL.PerformInBackground"; $curl)]
...
```

Later when we looped over all recipients, we need to wait for all cURL sessions to finish. So we loop over all 8 indices and check if they are running. In case one is still running, we wait a bit and when it is done, we run the HandleFinishedCURL script again. This way we make sure we wait for all emails being sent.

```
#Wait for all to close
Set Variable [$index; Value:1]
Loop
	If [$curls[$index] = ""]
		#found free index, no used
	Else
		#found used index, wait for finished
		Loop
			Exit Loop If [MBS( "CURL.IsRunning"; $curls[$index] ) = 0]
			Pause/Resume Script [Duration (seconds): ,1]
		End Loop
		#handle result
		Perform Script [“HandleFinishedCURL”; Parameter: $curls[$index]]
	End If
	Set Variable [$index; Value:$index + 1]
	Exit Loop If [$index = 9]
End Loop
```

As usual you need to cleanup. With our IsEmpty() check we can just write that inline in a set variable script step. I wrote 8 lines, but that could all be in one calculation.

```
# Cleanup
Set Variable [$r; Value:If(IsEmpty($curls[1]); 0; MBS("CURL.Cleanup"; $curls[1]))]
Set Variable [$r; Value:If(IsEmpty($curls[2]); 0; MBS("CURL.Cleanup"; $curls[2]))]
Set Variable [$r; Value:If(IsEmpty($curls[3]); 0; MBS("CURL.Cleanup"; $curls[3]))]
Set Variable [$r; Value:If(IsEmpty($curls[4]); 0; MBS("CURL.Cleanup"; $curls[4]))]
Set Variable [$r; Value:If(IsEmpty($curls[5]); 0; MBS("CURL.Cleanup"; $curls[5]))]
Set Variable [$r; Value:If(IsEmpty($curls[6]); 0; MBS("CURL.Cleanup"; $curls[6]))]
Set Variable [$r; Value:If(IsEmpty($curls[7]); 0; MBS("CURL.Cleanup"; $curls[7]))]
Set Variable [$r; Value:If(IsEmpty($curls[8]); 0; MBS("CURL.Cleanup"; $curls[8]))]
```

Now we show you our referenced HandleFinishedCURL script. First we get the cURL session ID from the script parameter. Than we query the tag, so we have the ID of the recipient. And we query debug messages and status code. If status code is zero, the email was sent. Else we should review the error messages soon. The example will now use SQL command to log the success or failure to the database.

```
Set Variable [$curl; Value:Get(ScriptParameter)]
Set Variable [$RecipientID; Value:MBS("CURL.GetTag"; $curl)]
#get debug messages
Set Variable [$DebugMessages; Value:MBS("CURL.GetDebugAsText"; $curl)]
Set Variable [$Status; Value:MBS( "CURL.ErrorCode"; $curl )]
...
```

We hope you enjoy the example. It'll be included in the next 7.0pr release. Or email me for a copy.