Send Audit Log via CURL in background

When we saw the new OnWindowTransaction, we got the idea to pass the JSON we receive to a web server to log the changes outside the FileMaker database. We had a similar thing with using MongoDB functions in a recent blog post: **OnWindowTransaction and MongoDB**. By storing the JSON outside, we can do the audit log on a different server. Whether you pass the JSON to another FileMaker Pro via our WebHook functions, a PHP script on your web server or directly send it to AWS to put in a database.

From the point of view of MBS FileMaker Plugin we just use the CURL functions. And there we have the CURL.PerformInBackground function to perform a HTTP Post in the background without slowing down the GUI. When the request is sent, we get either an expression evaluated or a script triggered. There we may check if the transfer worked, maybe do something special if it failed. But on the end we always free the curl object.
Take a look on the script:

```
Set Variable [ $json ; Value: Get(ScriptParameter) ]
Set Variable [ $URL ; Value: "https://yourdomain.com/yourscript.php" ]
#
Set Variable [ $curl ; Value: MBS("CURL.New") ]
# configure transfer
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; $URL) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionPostFields"; $curl; $json) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionHTTPHeader"; $curl; "Content-
Type: application/json") ]
#
# Let the evaluate then free the transfer
# Set Variable [ $r ; Value: MBS("CURL.SetFinishedEvaluate"; $curl;
"MBS(\"CURL.Release\"; $$ID$$)" ) ]
# or run a script to handle result
Set Variable [ $r ; Value: MBS("CURL.SetFinishedScript"; $curl; Get(FileName);
"Transfer Finished" ) ]
#
# run in background thread asynchronously
Set Variable [ $r ; Value: MBS("CURL.PerformInBackground"; $curl) ]
```

You can have multiple transfers running parallel and no user has to wait for a script to finish, since it ends with the call to CURL.PerformInBackground. With the finish script or expression, we can inspect the result and free the curl session. Since curl has a connection cache, it may reuse the connection automatically. Whether you use the finished script, you get the $curl as parameter. In the evaluate, you use $$ID$$ as placeholder for the plugin to put the $curl value. Of course you can use a Let statement to bundle multiple calls there. In both cases

you can check response code, the error code, the effective URL and other values. If you like, you can use CURL.SetTag in first script and CURL.GetTag in second script to pass whatever data you need.

Here is the sample Transfer Finished script:

```
Set Variable [ $curl ; Value: Get(ScriptParameter) ]
#
# check result?
Set Variable [ $Error ; Value: MBS( "CURL.ErrorCode"; $curl ) ]
Set Variable [ $ResponseCode ; Value: MBS( "CURL.GetResponseCode";
$curl ) ]
Set Variable [ $EffectiveURL ; Value: MBS( "CURL.GetEffectiveURL"; $curl ) ]
#
Set Variable [ $r ; Value: MBS("Curl.Release"; $curl) ]
```

If the script doesn't run, please check your privilege set for the option "Validate cross-file plug-in access (fmplugin)" and make sure it is checked to allow the script trigger.

The example file will be included with the MBS FileMaker Plugin. Please do not hesitate to contact us with your questions.