How long do you wait for Insert From URL to finish?

Did you know you can run CURL commands in background with MBS FileMaker Plugin?

Let's start with a simple Insert From URL script like the following one. $url has the URL to query and $json the data to send to whatever web API you like to query. The headers include authentication and content types. Nothing special. The query runs and stores result in $$result and a log into $$debug. Here is the script:

```
Set Variable [ $options ; Value: "-X POST --header \"Authorization: Basic <<API KEY>>\" --header \"Content-Type: application/json\" --header \"Accept: application/json\" --trace $$debug --data @$json " ]
Insert from URL [ Select ; With dialog: Off ; Target: $$result ; $url ; cURL options: $options ; Do not automatically encode URL ]
```

After this is run through, you may inspect Get(LastError) as well as $$result and $$debug. And depending on how long your query runs, the FileMaker script may pause. If the server isn't answering, it may wait 90 seconds for a timeout (unless changed). Especially if you upload images over slow connections, the user may have to wait.

Let's do the same transfer with the CURL functions in our MBS FileMaker Plugin. We create a new CURL sessions, fill in the options one by one. Then we call perform where the plugin does the transfer. This may take a while like FileMaker and provide the result and debug log similar to FileMaker:

```
Set Variable [ $curl ; Value: MBS("CURL.New") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; $url) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionPostFields"; $curl; $json; "UTF-8") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionHTTPHeader"; $curl; "Authorization: Basic <<API KEY>>¶Content-Type: application/json¶Accept: application/json") ]
# perform transfer
Set Variable [ $error ; Value: MBS("CURL.Perform"; $curl) ]
# get result
Set Variable [ $$result ; Value: MBS("CURL.GetResultAsText"; $curl; "UTF-8") ]
Set Variable [ $$debug ; Value: MBS("CURL.GetDebugMessages"; $curl) ]
Set Variable [ $r ; Value: MBS("CURL.Cleanup"; $curl) ]
```

Since this is all just MBS calls, you stuff them all into one big Let() statement and execute anywhere without running a script:

```
Let([
curl = Value: MBS("CURL.New");
r = Value: MBS("CURL.SetOptionURL"; curl; $url);
r = Value: MBS("CURL.SetOptionPostFields"; curl; $json;
"UTF-8");
r = Value: MBS("CURL.SetOptionHTTPHeader"; curl;
"Authorization: Basic <<API KEY>>¶Content-Type: application/
json¶Accept: application/json");
error = Value: MBS("CURL.Perform"; curl);
$$result = Value: MBS("CURL.GetResultAsText"; curl;
"UTF-8");
$$debug = Value: MBS("CURL.GetDebugMessages"; curl);
r = Value: MBS("CURL.Release"; curl)
]; error)
```

To run it in the background, we use the CURL.PerformInBackground function. This will schedule the transfer on a new thread. FileMaker can do whatever it likes, e.g. show a dialog, but the transfer will continue. We have to set a script or expression to trigger when finished. Here is the script:

```
Set Variable [ $curl ; Value: MBS("CURL.New") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; $url) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionPostFields"; $curl; $json; "UTF-8")
]
Set Variable [ $r ; Value: MBS("CURL.SetOptionHTTPHeader"; $curl;
"Authorization: Basic <<API KEY>>¶Content-Type: application/json¶Accept:
application/json") ]
# perform transfer in background
Set Variable [ $result ; Value: MBS( "CURL.SetFinishedScript"; $curl;
Get(FileName); "curl Download done" ) ]
Set Variable [ $r ; Value: MBS("CURL.PerformInBackground"; $curl) ]
```

You write a script to continue after the transfer. We pass you the CURL reference number and you can use it to query status. The example script below queries the debug messages, the text from the web service. Especially the response code may tell you whether the query was successful (2xx code) or failed (code ≥ 400).

```
Set Variable [ $curl ; Value: Get(ScriptParameter) ]
Set Variable [ $$debug ; Value: MBS("CURL.GetDebugMessages"; $curl) ]
Set Variable [ $$result ; Value: MBS("CURL.GetResultAsText"; $curl; "UTF-8") ]
Set Variable [ $ResponseCode ; Value: MBS("CURL.GetResponseCode"; $curl) ]
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl) ]
```

You can use background transfers to upload larger files or sending emails. If needed, you run multiple transfers at the same time. Our batch email example sends multiple emails in parallel. While one email uploads, another connection

may be at the login part and a third waits for server acknowledge the email is accepted. By using bandwidth for some transfers while others wait, we can increase the total used bandwidth and thus reduce the total time needed to upload multiple items.

Please try and let us know if you have questions.