Parallel network transfers with MBS Plugin

You may know how to download a file with either our CURL functions or Insert From URL in FileMaker. But have you ever tried to download multiple files in parallel?

Let us show you three variants. First using FileMaker's Insert from URL, then CURL sequential and finally CURL parallel. The FileMaker one may quite common, but you can speed up your processing with parallel requests and run a lot of them at the same time. Let's say you have a job to download 4 files together and you like to do it faster. This can of course also be changed to do multiple uploads and of course you could use index variables and loops to code it for a variable number of requests.

First variant: Insert From URL

Using Insert From URL may be familiar, but isn't as flexible as our CURL functions:

```
Set Error Capture [ On ]
#
# first one
Insert from URL [ Select ; With dialog: Off ; Target: CURL Test::HTML1 ; CURL
Test::URL1 ; cURL options: " --trace $$debug1 " ]
Set Variable [ $error1 ; Value: Get(LastError) ]
If [ $error1 ]
	Set Field [ CURL Test::debug ; $$debug1 ]
End If
#
# second download
Insert from URL [ Select ; With dialog: Off ; Target: CURL Test::HTML2 ; CURL
Test::URL2 ; cURL options: " --trace $$debug2 " ]
Set Variable [ $error2 ; Value: Get(LastError) ]
If [ $error2 ]
	Set Field [ CURL Test::debug ; $$debug2 ]
End If
#
# third download
Insert from URL [ Select ; With dialog: Off ; Target: CURL Test::HTML3 ; CURL
Test::URL3 ; cURL options: " --trace $$debug3 " ]
Set Variable [ $error3 ; Value: Get(LastError) ]
If [ $error3 ]
	Set Field [ CURL Test::debug ; $$debug3 ]
End If
#
# forth download
```

```
Insert from URL [ Select ; With dialog: Off ; Target: CURL Test::HTML4 ; CURL
Test::URL4 ; cURL options: " --trace $$debug4 " ]
Set Variable [ $error4 ; Value: Get(LastError) ]
If [ $error4 ]
	Set Field [ CURL Test::debug ; $$debug4 ]
End If
```

Second variant: CURL sequential

Next we do the same with MBS FileMaker Plugin and just download all files one after one. We can reuse the $curl, so we may save some time to create and destroy curl objects.

```
# Start new session
Set Variable [ $curl ; Value: MBS("CURL.New") ]
#
# first one
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; CURL Test::URL1) ]
Set Variable [ $result1 ; Value: MBS("CURL.Perform"; $curl) ]
If [ $result1 = "OK" ]
	Set Field [ CURL Test::HTML1 ; MBS("CURL.GetResultAsText"; $curl;
"UTF8") ]
Else
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl) ]
End If
#
# second download
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; CURL Test::URL2) ]
Set Variable [ $result2 ; Value: MBS("CURL.Perform"; $curl) ]
If [ $result2 = "OK" ]
	Set Field [ CURL Test::HTML2 ; MBS("CURL.GetResultAsText"; $curl;
"UTF8") ]
Else
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl) ]
End If
#
# third download
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; CURL Test::URL3) ]
Set Variable [ $result3 ; Value: MBS("CURL.Perform"; $curl) ]
If [ $result3 = "OK" ]
	Set Field [ CURL Test::HTML3 ; MBS("CURL.GetResultAsText"; $curl;
"UTF8") ]
Else
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl) ]
End If
```

```
#
# forth download
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; CURL Test::URL4) ]
Set Variable [ $result4 ; Value: MBS("CURL.Perform"; $curl) ]
If [ $result4 = "OK" ]
	Set Field [ CURL Test::HTML4 ; MBS("CURL.GetResultAsText"; $curl;
"UTF8") ]
Else
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl) ]
End If
#
# Cleanup
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl) ]
```

Third variant: CURL parallel

Finally we can show you running these four requests in parallel.

```
# Start new sessions
Set Variable [ $curl1 ; Value: MBS("CURL.New") ]
Set Variable [ $curl2 ; Value: MBS("CURL.New") ]
Set Variable [ $curl3 ; Value: MBS("CURL.New") ]
Set Variable [ $curl4 ; Value: MBS("CURL.New") ]
#
# set all the URLs
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl1; CURL
Test::URL1) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl2; CURL
Test::URL2) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl3; CURL
Test::URL3) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl4; CURL
Test::URL4) ]
#
# launch the transfers
Set Variable [ $r ; Value: MBS("CURL.PerformAsync"; $curl1) ]
Set Variable [ $r ; Value: MBS("CURL.PerformAsync"; $curl2) ]
Set Variable [ $r ; Value: MBS("CURL.PerformAsync"; $curl3) ]
Set Variable [ $r ; Value: MBS("CURL.PerformAsync"; $curl4) ]
#
# now wait for all to finish
Loop [ Flush: Always ]
	Pause/Resume Script [ Duration (seconds): ,01 ]
```

```
	Set Variable [ $Running ; Value: MBS("CURL.IsRunning"; $curl1) or
MBS("CURL.IsRunning"; $curl2) or MBS("CURL.IsRunning"; $curl3) or
MBS("CURL.IsRunning"; $curl4) ]
	Exit Loop If [ not $Running ]
End Loop
#
# check results
Set Variable [ $result1 ; Value: MBS("CURL.ErrorCode"; $curl1) ]
Set Variable [ $result2 ; Value: MBS("CURL.ErrorCode"; $curl2) ]
Set Variable [ $result3 ; Value: MBS("CURL.ErrorCode"; $curl3) ]
Set Variable [ $result4 ; Value: MBS("CURL.ErrorCode"; $curl4) ]
#
# get the downloaded html
Set Field [ CURL Test::HTML1 ; MBS("CURL.GetResultAsText"; $curl1; "UTF8") ]
Set Field [ CURL Test::HTML2 ; MBS("CURL.GetResultAsText"; $curl2; "UTF8") ]
Set Field [ CURL Test::HTML3 ; MBS("CURL.GetResultAsText"; $curl3; "UTF8") ]
Set Field [ CURL Test::HTML4 ; MBS("CURL.GetResultAsText"; $curl4; "UTF8") ]
#
# if one of them goes wrong, show log in a field
If [ $result1 ≠ 0 ]
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl1) ]
End If
If [ $result2 ≠ 0 ]
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl2) ]
End If
If [ $result3 ≠ 0 ]
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl3) ]
End If
If [ $result4 ≠ 0 ]
	Set Field [ CURL Test::debug ; MBS("CURL.GetDebugMessages"; $curl4) ]
End If
#
# Cleanup
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl1) ]
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl2) ]
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl3) ]
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl4) ]
```

If you run all three variants, you would see, that the last variant is faster. Your milage may vary depending on how quick servers react, the network connection and how big files are. There is time needed to connect, to initialize transport encryption, to send the request and wait for the answer to arrive. While this all happens for one request, you can also do the same for the next request. Bandwidth is shared, but for a lot of time of the script running, there is no packet transferred as one waits for the other.

Our CURL.PerformAsync function is great to perform things in background while your script runs with lots of parallel requests. The CURL.PerformInBackground function can run transfers on separate threads, so they won't stop even when FileMaker shows a dialog. And you can use it server side or in your FileMaker iOS SDK application, too.

Please try and let us know if you have questions.

See also

- Send Audit Log via CURL in background
- How long do you wait for Insert From URL to finish?
- Batch sending Emails in FileMaker via MBS Plugin
- CURL Tutorial for Filemaker with MBS Plugin