

Trigger Scripts via WebHook

Today we like to show you how to use our [WebHook](#) functions in [MBS FileMaker Plugin](#) to listen for incoming requests and send them from another place within the reachable network. Since a VPN and tunnelling can extend the range, you may even trigger from a FMP in one country to a FMP in another country. You may have various uses of triggering a script within the network.

For example you may have a need to trigger a script on another FileMaker Pro to do something for you. That may be to remotely start a batch process, which takes a while. You may not have our plugin in FileMaker Go, but you can trigger a script on a FileMaker Pro to do something like cropping a picture. Or the FileMaker Go may enter data for an invoice. Then it could send a request to a FileMaker Pro to print that invoice after it is finished.

Let us start with creating a web hook locally in one FileMaker Pro. We request to keep connections open and disable auto answer. Then we bind to a new random port here. We set the script to trigger and finally query the port number we got. Then we write this in a record, so others can read this configuration value to send something. Since the port is random, it changes each time our FileMaker Pro launches. Here is the script:

```
Set Variable [ $$WebHooks ; Value: MBS("WebHook.Create") ]
# we need to keep connection open
Set Variable [ $r ; Value: MBS("WebHook.SetMode"; $$WebHooks; 1) ]
Set Variable [ $r ; Value: MBS("WebHook.SetAutoAnswer"; $$WebHooks; "" ) ]
# listen on random port
Set Variable [ $r ; Value: MBS("WebHook.Listen"; $$WebHooks; 0) ]
# set script to run
Set Variable [ $r ; Value: MBS("WebHook.SetScript"; $$WebHooks;
Get(FileName); "WebHookReceived") ]
#
# now show IP and Port
Set Field [ Listener::Port ; MBS( "WebHook.GetPort"; $$WebHooks ) ]
Set Field [ Listener::My IP ; Get(SystemIPAddress) ]
Commit Records/Requests [ With dialog: Off ]
```

Now let us check the triggered script. This one gets the request number and then we query the URL components. We parse the URL and query the script and file name. The parameter is picked from the body of the request with proper text encoding (the same as we send later). The script is then performed via Perform Script with passing the name. Older FileMaker versions can use our [FM.RunScript](#) function instead. Next we send out the answer as HTTP Response. For that we need to make sure we have the correct length with encoding it as hex and then divide length by 2 to get the byte count in UTF-8. When you send out the headers

with CRLF line endings. Finally we can release the web request to cleanup memory. The full script is here:

```
# Which request triggered this?
Set Variable [ $WebRequest ; Value: Get(ScriptParameter) ]
# Parse parameters from the URL
Set Variable [ $URLComponents ; Value: MBS( "WebRequest.URLComponents";
$WebRequest ) ]
Set Variable [ $Parameters ; Value: JSONGetElement ( $URLComponents ;
"Parameters" ) ]
#
Set Variable [ $ScriptName ; Value: JSONGetElement ( $Parameters ;
"ScriptName" ) ]
Set Variable [ $FileName ; Value: JSONGetElement ( $Parameters ;
"FileName" ) ]
#
# we can move several Megabytes of Data in this parameter
Set Variable [ $Parameter ; Value: MBS( "WebRequest.GetBody"; $WebRequest;
"UTF-8") ]
Set Variable [ $Result ; Value: "" ]
#
Perform Script [ Specified: By name ; $ScriptName ; Parameter: $Parameter ]
Set Variable [ $Result ; Value: Get(ScriptResult) ]

# send HTTP Response here
# get length of answer
Set Variable [ $body ; Value: MBS( "Text.EncodeToHex"; $Result; "UTF-8") ]
Set Variable [ $ContentLength ; Value: Length($body) / 2 ]
# build header
Set Variable [ $text ; Value: "HTTP/1.1 200 OK¶Server: MyServer 1.0¶Connection:
close¶Content-Type: plain/text¶Content-Length: " & $contentLength & "¶¶" ]
Set Variable [ $text ; Value: MBS( "Text.ReplaceNewline"; $Text; 3 ) ]
Set Variable [ $r ; Value: MBS("WebRequest.Send"; $WebRequest; $text &
$result; "UTF-8") ]
#
# give time to send answer
Pause/Resume Script [ Duration (seconds): ,5 ]
Set Variable [ $r ; Value: MBS("WebRequest.Release"; $WebRequest) ]
Exit Script [ Text Result:  ]
```

Time to trigger the script. This can for example happen via [CURL](#) functions in [MBS FileMaker Plugin](#). We encode script and file name for inclusion in the RUL. The URL is then build with the protocol http, the IP or domain name from a field and a port number from another field. Separated with a question mark we add the file name and script names. We start a CURL session with the URL. The timeout is

set to short time for our example to avoid long waiting times. The parameter is sent as body in UTF-8 encoding, so we can send multiple megabytes if needed. And that can be JSON or XML to structure it. When the transfer run through, we can store result and debug messages to check later.

```
# Send request via CURL functions in MBS plugin and wait for result
Set Variable [ $ScriptName ; Value: GetAsURLEncoded ( Trigger script with
Webhook::ScriptName ) ]
Set Variable [ $FileName ; Value: GetAsURLEncoded ( Trigger script with
Webhook::FileName ) ]
Set Variable [ $Parameter ; Value: Trigger script with Webhook::Parameter ]
# build the URL
Set Variable [ $URL ; Value: "http://" & Trigger script with Webhook::Target IP & ":"
& Trigger script with Webhook::Target Port & "?FileName=" & $FileName &
"&ScriptName=" & $ScriptName ]
#
# Start new session
Set Variable [ $curl ; Value: MBS("CURL.New") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; $URL) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionConnectionTimeout"; $curl; 5) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionTimeout"; $curl; 30) ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionPostFields"; $curl; Trigger script
with Webhook::Parameter; "UTF-8") ]
# RUN now
Set Variable [ $r ; Value: MBS("CURL.Perform"; $curl) ]
# Check result
Set Variable [ $result ; Value: MBS("CURL.GetResultAsText"; $curl; "UTF8") ]
Set Variable [ $debug ; Value: MBS("CURL.GetDebugAsText"; $curl; "UTF8") ]
# store logs to debug
Set Field [ Trigger script with Webhook::Result ; $result ]
Set Field [ Trigger script with Webhook::Debug ; $debug ]
Commit Records/Requests [ With dialog: Off ]
# Cleanup
Set Variable [ $result ; Value: MBS("CURL.Release"; $curl) ]
Exit Script [ Text Result: $result ]
```

The same can be done on FileMaker Go with Insert From URL and the similar script. The URL is the same, but we pass the CURL options as an options string. There we encode the timeouts and where to pick from the parameter value and where to store the debug messages.

```

# Running via Insert From URL allows this to run on FileMaker Go
Set Variable [ $ScriptName ; Value: GetAsURLEncoded ( Trigger script with
Webhook::ScriptName ) ]
Set Variable [ $FileName ; Value: GetAsURLEncoded ( Trigger script with
Webhook::FileName ) ]
Set Variable [ $Parameter ; Value: Trigger script with Webhook::Parameter ]
#
Set Variable [ $URL ; Value: "http://" & Trigger script with Webhook::Target IP & ":"
& Trigger script with Webhook::Target Port & "?FileName=" & $FileName &
"&ScriptName=" & $ScriptName ]
#
Set Variable [ $result ; Value: "" ]
Set Variable [ $debug ; Value: "" ]
Set Variable [ $options ; Value: "--FM-text-encoding utf-8 --connect-timeout 5 --
timeout 30 --data @$Parameter --trace $debug " ]
Insert from URL [ Select ; With dialog: Off ; Target: $result ; $URL ; cURL options:
$options ; Do not automatically encode URL ]
#
Set Field [ Trigger script with Webhook::Result ; $result ]
Set Field [ Trigger script with Webhook::Debug ; $debug ]
Commit Records/Requests [ With dialog: Off ]

```

If you like, you can add some security. A good idea may be to generate a random UUID, which is then written with the port number to a record. And then include the UUID as header in the request and check it on the server. You may also do SSL to encrypt the transfers.

The example is included with MBS Plugin 12.6pr2 and newer.

See also [WebHook Introduction](#) and [Using MonkeyBread for FileMaker Webhooks - Day 1](#) and [Day 2](#).