MQTT in FileMaker

Recently a client asked about whether we can do MQTT inside FileMaker to subscribe to a topic and receive messages or even send messages. Let's summarize what MQTT is for you:

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for devices with limited resources and for low-bandwidth, high-latency, or unreliable networks. It operates on a publish/subscribe model which allows for efficient distribution of information to many receivers.

We discovered that we have MQTT capabilities built-in to the CURL library. Just use an URL with "mqtt://" followed by IP or domain name of the server. In the path of the URL goes the subject to subscribe or post to. If you like, set a user name and password. Run the transfer and check the debug messages for success. Let us show you how to this in a script below.

Post Message

To post you put the data into the post fields and run the transfer. For example here we make a transfer to send the test message to the server:

```
Set Variable [ $curl ; Value: MBS("CURL.New") ]
# connect to MQTT server and send
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; "mqtt://
yourserver.test:1883/test/test") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionPostFields"; $curl; "Hello World";
"UTF-8") ]
#
# send request now
Set Variable [ $result ; Value: MBS("CURL.Perform"; $curl) ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Failed to send." ; $result ]
End If
#
# for checking error details
Set Variable [ $$debugLog ; Value: MBS("CURL.GetDebugMessages"; $curl;
"UTF-8") ]
Set Variable [ $$output ; Value: MBS("CURL.GetResultAsText"; $curl; "UTF-8") ]
#
Set Variable [ $r ; Value: MBS("CURL.Release"; $curl) ]
```

On success the CURL.Perform function returns OK, otherwise an error message. In case of a failure, please check the debug log for details.

Subscribe

To subscribe you connect and then react to incoming data with our script trigger. But first you need to run a transfer asynchronously in the background to do the setup. The connection stays open as long as you want to listen to incoming messages. We use CURL.SetMQTTScript function to tell the plugin which script to run when new data is received. Here is the sample code to setup the connection:

```
Set Variable [ $curl ; Value: MBS("CURL.New") ]
#
# connect to MQTT server
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; "mqtt://
yourserver.test:1883/test/test") ]
Set Variable [ $r ; Value: MBS("CURL.SetFinishedScript"; $curl; Get(FileName);
"Finished") ]
Set Variable [ $r ; Value: MBS("CURL.SetMQTTScript"; $curl; Get(FileName);
"Received") ]
#
# run in background
Set Variable [ $result ; Value: MBS("CURL.PerformAsync"; $curl) ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Failed to send." ; $result ]
End If
```

As you see we keep a reference to the curl connection open and don't call CURL.Release function right now. We also setup a finished script to be called when the connection is dropped and we can do cleanup. We use CURL.PerformAsync function too tell MBS FileMaker Plugin to run the transfer in the background and trigger scripts as needed. Alternatively you could of course run a loop and check status every few seconds to react within a script.

Receive

When data is received, the MQTT script trigger firer and triggers our script. We run the CURL.GetMQTTMessages function to ask for the received data. The plugin prepares a JSON array for us with objects containing the topic and message data. And in the script we can loop over the entries and process them.

```
Set Variable [ $curl ; Value: Get(ScriptParameter) ]
Set Variable [ $json ; Value: MBS("CURL.GetMQTTMessages"; $curl) ]
#
Set Variable [ $count ; Value: MBS("JSON.GetArraySize"; $json) ]
Set Variable [ $index ; Value: 0 ]
If [ $index < $count ]
	Loop [ Flush: Always ]
		Set Variable [ $entry ; Value: MBS("JSON.GetArrayItem"; $json;
$index) ]
		New Record/Request
		Set Field [ MQTT::Channel ; MBS("JSON.GetPathItem"; $entry;
"topic"; 3) ]
		Set Field [ MQTT::Message ; MBS("JSON.GetPathItem"; $entry;
"content"; 3) ]
		Commit Records/Requests [ With dialog: Off ]
		#
		# next
		Set Variable [ $index ; Value: $index + 1 ]
		Exit Loop If [ $index ≥ $count ]
	End Loop
End If
```

The plugin manages a queue and every time you call CURL.GetMQTTMessages, the queue is emptied. If you use a script to loop over messages, you can simply just call it regularly to see if something new arrived.

Please try and let us know whether it works for you. Sending and receiving MQTT messages in FileMaker solutions may help you to automate a few things in your company.

If you need more we could of course add MQTT library in the future. Or you just use the new Python functions in our plugin to use a MQTT module for Python to connect. Or you use a web service interface to the same MQTT server or a proxy to connect. Like connecting with CURL or Insert From URL to a PHP script, which does the connection then.

Please try with 14.3pr versions of our plugins and let us know when you have questions.