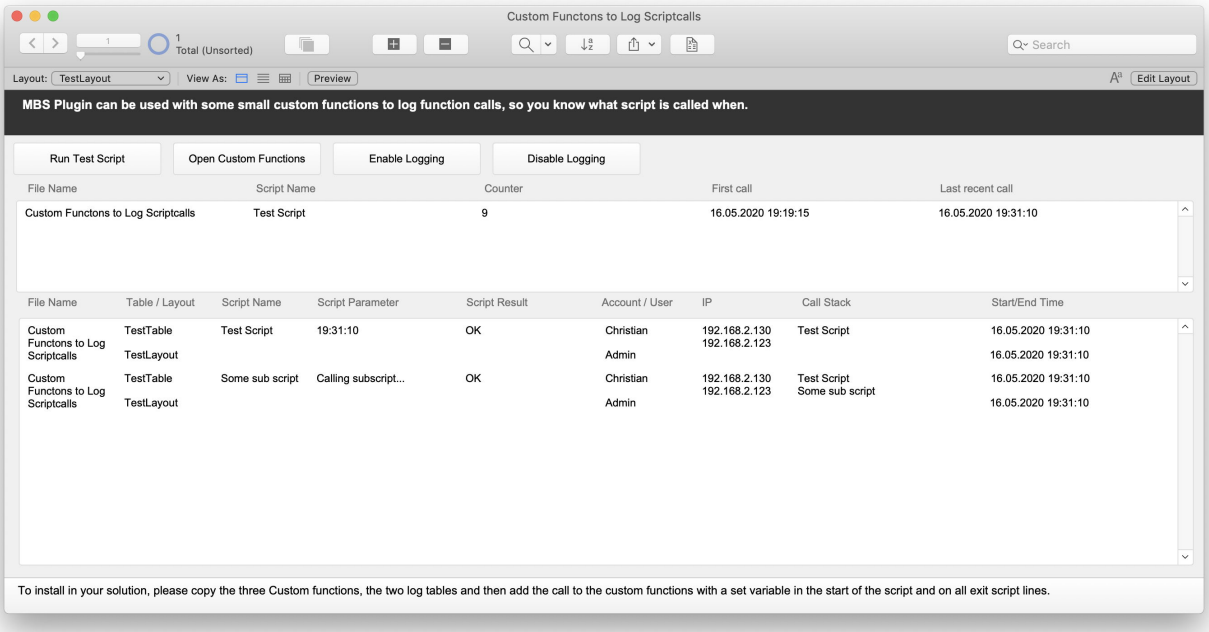
Custom Functions to Log Script Calls and maintain call stack

We have three handy custom functions for you to log function calls with MBS FileMaker Plugin:

1. CountScriptCall
2. LogFunctionCall
3. LogFunctionResult

And they can be used together to log function calls like on the screenshot here:



CountScriptCall

Now let's take a look on each function. Start of the script may just call the first two functions together in one Set variable script step:
Set Variable [$r ; Value: CountScriptCall & LogFunctionCall]
The CountScriptCall function is to check which scripts get called and how often. We can call it in each script on the start to log that this script is still in use. We use a global variable $$LogScriptCalls to turn this on/off as needed. In the custom function we first try to insert a record via FM.InsertRecord in the log table. If that fails as the record with same unique key is already there, we use the FM.ExecuteFileSQL to run an SQL statement to update the counter and increase it. FileMaker also updates the modification time stamp field. This way we know the first time the function was called via creation date and the

last time called via modification date. Take a look on our CountScriptCall function here:

```
If($$LogScriptCalls;
Let ( [
FileName = Get(FileName);
ScriptName = Get(ScriptName);

// try to insert new record
r = MBS( "FM.InsertRecord"; FileName; "ScriptCallCounter"; "Counter";
1; "ScriptName"; ScriptName; "FileName"; FileName );

// r shows OK or "[MBS] ERROR:  (504): Field failed unique value
validation test" in case of duplicate

// check for error: 504
p = Position ( r ; "(504)" ; 1 ; 1 );

// update existing call on error
r = If( p > 0; MBS( "FM.ExecuteFileSQL"; FileName; "UPDATE
\"ScriptCallCounter\" SET \"Counter\" = \"Counter\" + 1 WHERE
\"ScriptName\" = ? AND \"FileName\" = ?"; 9; 13; ScriptName;
FileName ); r)

]; r ); "")
```

As you see we pass FileName to the two MBS functions for the file name of the log table. But that is not required as you can move the logs to another file. This way logs appear in a different file as they can become quite big.

LogFunctionCall

This function should log a function call. Again enabled via $ $LogScriptCalls global variable, we query a few information pieces for the context like user name, current table and layout. We then generate an unique ID and store it in a local variable to be able to later find the record again. We add our script to the $$CallStack variable, we always have the current call stack in a variable and we can know which script calls which other script. To log we use FM.InsertRecord again to pass all the field names and values.

Here is the calculation for the LogFunctionCall function:

```
If($$LogScriptCalls; Let ( [

// query some values about current context
FileName = Get(FileName);
ScriptName = Get(ScriptName);
TableName = Get(LayoutTableName);
AccountName = Get(AccountName);
LayoutName = Get(LayoutName);
UserName = Get(UserName);
IP = Get(SystemIPAddress);
ScriptParameter = Get(ScriptParameter);

// make an unique key for later
$ScriptCallID = Get(UUID);

// and build call stack
$$CallStack = $$CallStack & ScriptName & ¶;

// now log it.
r = MBS( "FM.InsertRecord"; FileName; "ScriptCallLog";
"ScriptCallID"; $ScriptCallID;
"ScriptName"; ScriptName;
"FileName"; FileName;
"TableName"; TableName;
"LayoutName"; LayoutName;
"AccountName"; AccountName;
"UserName"; UserName;
"IP"; IP;
"ScriptParameter"; GetAsText(ScriptParameter);
"CallStack"; $$CallStack;
"StartTime"; Get ( CurrentTimestamp ))

]; r ); "")
```

Every script must have such a line on the end and you call the function for each way the script can be exited. That is some work, but only if the function is called, we can cleanup the $$CallState variable. We check $ScriptCallID variable for whether we have an ID set earlier. If so, we can use it to update the log and add the end time and the script result to the table. As start and end time of each script is set, you can calculate later how long those scripts take.

Here the calculation for the LogFunctionResult function with parameter Result:

```
If(not IsEmpty($ScriptCallID); Let ( [

// remove us from call stack
$$CallStack = LeftValues ( $$CallStack ; ValueCount ( $$CallStack ) - 1
);

// now updat entry in log to show result.
FileName = Get(FileName);

r = MBS( "FM.ExecuteFileSQL"; FileName; "UPDATE \"ScriptCallLog\"
SET \"ScriptResult\" = ?, \"EndTime\" = ? WHERE \"ScriptCallID\" = ?";
9; 13; GetAsText( Result ); Get(CurrentTimestamp); $ScriptCallID )

]; Result ); Result)
```

What do you think about those functions?
See problems? Flaws? Please let us know, so we can update our example.

The example database will be included with future plugin updates.
See also Looping over records in FileMaker with error checking for the CheckError custom function.