

How to use Trace

When you run into trouble and you contact us via support@ email address, we may frequently question you to provide:

- The text of your script.
But, please no screenshot. You can copy the script text into an email or print to PDF.
- The log of plugin calls from [Trace](#) function, so we see what you call and what parameters you actually pass.
- If [CURL](#) functions are involved, the CURL debug messages via [CURL.GetDebugMessages](#) function.

We frequently ask to reproduce the problem in a new database or with a short script. It's hard to dig through 1000 lines of script to come to the 10 lines where the problem shows and make sure none of the lines above causes the trouble.

Let us show you how to use [Trace](#) function in various situations:

FileMaker Server

Add a line to a script like this:

```
Set Variable [$r; Value: MBS("Trace")]
```

After this call, all plugin calls are logged to a log file in the Logs folder inside the FileMaker Server folder, where you find the normal FileMaker log files. We use the following log files:

- StdErrServerScriptingPlugins.log for Server Scripting and PSoS
- StdErrWebPlugins.log for Web Direct
- StdErrDataAPIPlugins.log for Data API process

If the process crashes, you find the old file being renamed to .old and a new .log file started.

You can of course include a native file path for the Trace call and log to a file, which you could also later read via script and send to yourself for debugging or store in a text field.

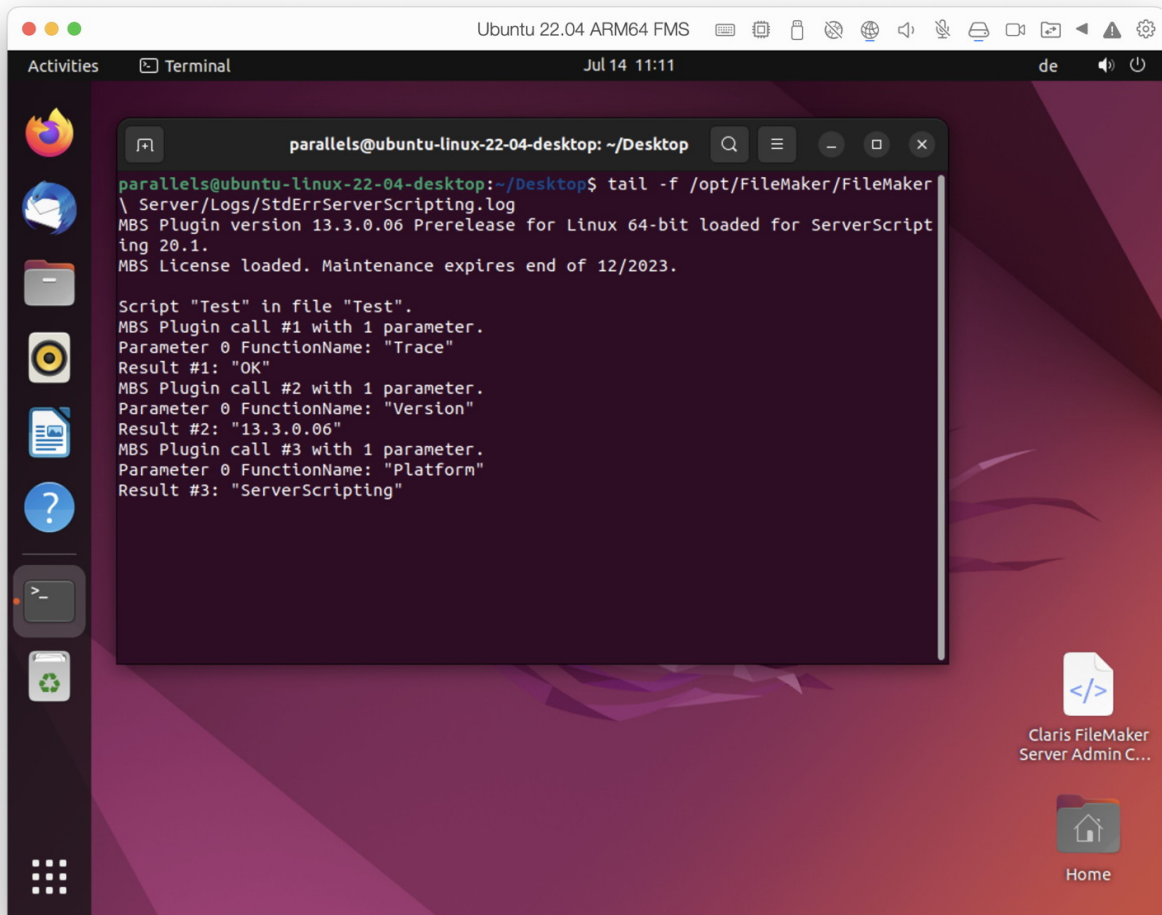
Watching Live on Server

On Windows, you can also watch live using [DebugView](#) application. You need to run it on the server as administrator. Then you need to enable "Capture Global Win32" in the Capture menu.

On Linux (and macOS), you can use a terminal window (or ssh connection) to run the tail command and see the logs live:

```
tail -f /opt/FileMaker/FileMaker\ Server/Logs/StdErrServerScriptingPlugins.log
```

The tail command will run until you exit with Control-C key combination and print out messages as they appear in the log file.



```
parallels@ubuntu-linux-22-04-desktop: ~/Desktop
parallels@ubuntu-linux-22-04-desktop:~/Desktop$ tail -f /opt/FileMaker/FileMaker\
\ Server/Logs/StdErrServerScripting.log
MBS Plugin version 13.3.0.06 Prerelease for Linux 64-bit loaded for ServerScripting
20.1.
MBS License loaded. Maintenance expires end of 12/2023.

Script "Test" in file "Test".
MBS Plugin call #1 with 1 parameter.
Parameter 0 FunctionName: "Trace"
Result #1: "OK"
MBS Plugin call #2 with 1 parameter.
Parameter 0 FunctionName: "Version"
Result #2: "13.3.0.06"
MBS Plugin call #3 with 1 parameter.
Parameter 0 FunctionName: "Platform"
Result #3: "ServerScripting"
```

FileMaker Pro

Add a line to a script like this:

```
Set Variable [$r; Value: MBS("Trace"; $file)]
```

Where \$file would point to a file you can access like "/Users/cs/Desktop/Trace.txt". You can automate this like this:

```
Set Variable [$r; Value: MBS( "Trace";
MBS( "Path.AddPathComponent"; MBS( "Folders.UserDesktop" );
"trace.txt" ) ) ]
```

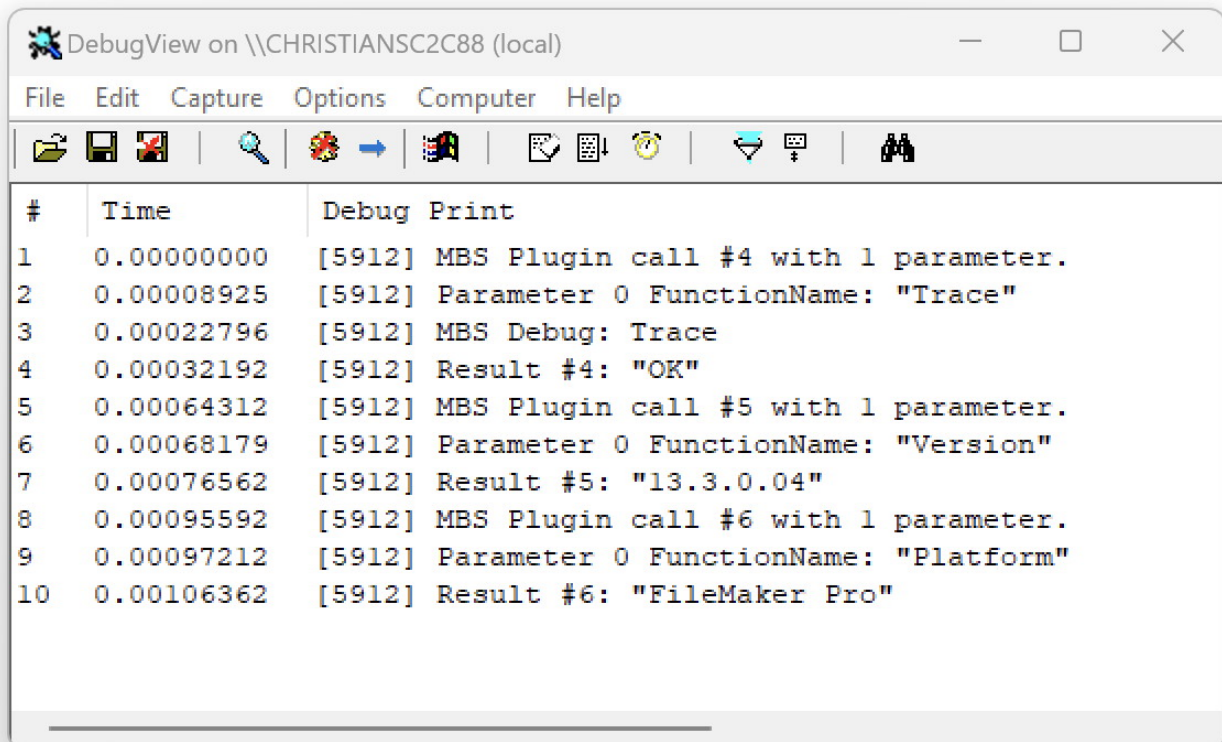
Now it writes to the desktop independent on where the desktop folder is. And [Path.AddPathComponent](#) makes sure we use \ or / depending on the platform.

Your scripts could even log to a temp file for a client, later read in that text file with [Text.ReadTextFile](#) function and put the text in a field or send it back to yourself.

Watching Live on Pro

If you skip the file path parameter in FileMaker Pro, you can watch live. For Windows use [DebugView](#) application from Microsoft. For macOS use Console.app and filter for MBS in the search field to hide all the other messages from other applications.

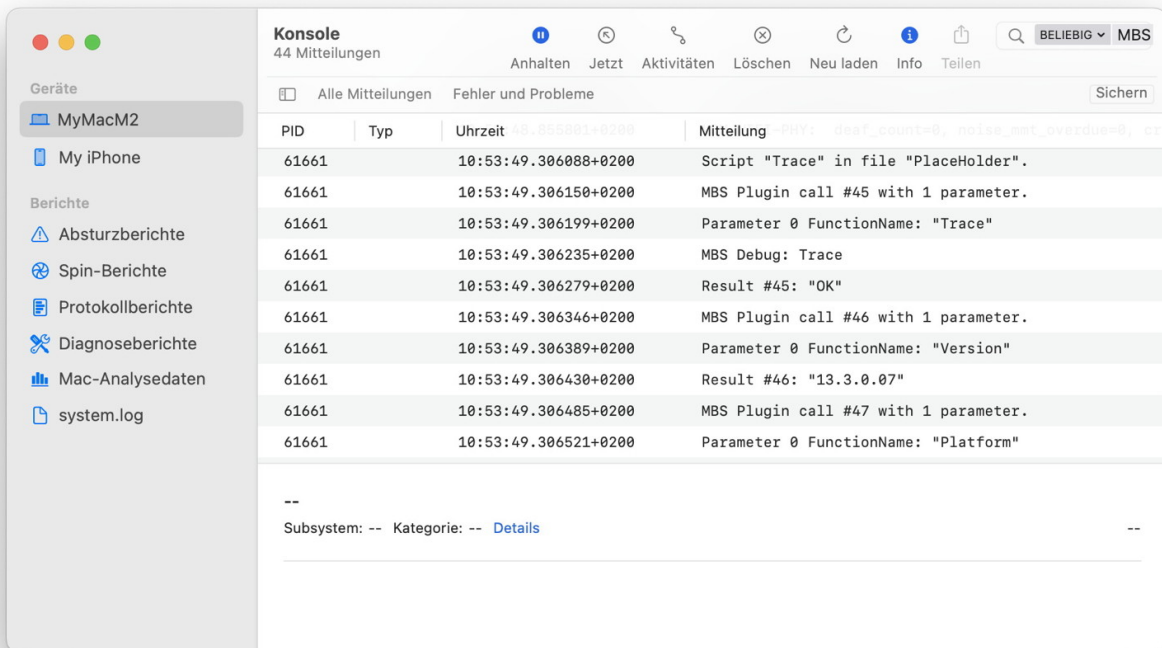
On Windows:



The screenshot shows the DebugView application window titled "DebugView on \\CHRISTIANS2C88 (local)". The window has a menu bar with "File", "Edit", "Capture", "Options", "Computer", and "Help". Below the menu bar is a toolbar with various icons. The main area displays a table of debug print messages:

#	Time	Debug Print
1	0.00000000	[5912] MBS Plugin call #4 with 1 parameter.
2	0.00008925	[5912] Parameter 0 FunctionName: "Trace"
3	0.00022796	[5912] MBS Debug: Trace
4	0.00032192	[5912] Result #4: "OK"
5	0.00064312	[5912] MBS Plugin call #5 with 1 parameter.
6	0.00068179	[5912] Parameter 0 FunctionName: "Version"
7	0.00076562	[5912] Result #5: "13.3.0.04"
8	0.00095592	[5912] MBS Plugin call #6 with 1 parameter.
9	0.00097212	[5912] Parameter 0 FunctionName: "Platform"
10	0.00106362	[5912] Result #6: "FileMaker Pro"

On macOS:

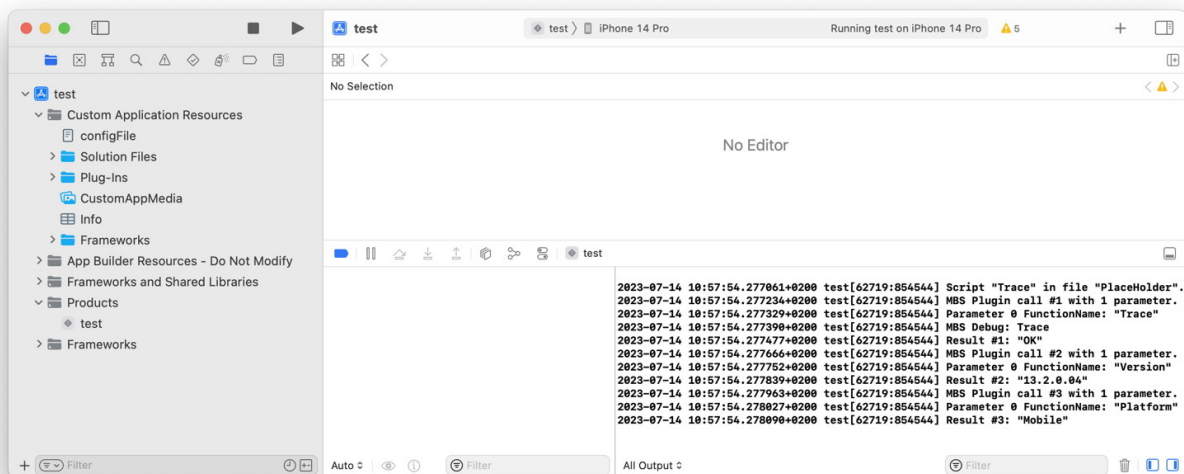


FileMaker iOS SDK

Add a line to a script like this:

Set Variable [{\$r; Value: MBS("Trace")}]

After this call, all plugin calls are logged to the Xcode console window when running from Xcode.



When running on the iPhone, you can open Console.app to see log messages from the console app.

You can of course include a native file path for the Trace call and log to a log file, which you could also later read via script and send to yourself for debugging or store in a text field.

Own logging

Check out our [Log](#) function to add your own entries.

```
Set Variable [$r; Value: MBS( "Log"; "Test: "; $test ) ]
```

And of course you can pass more parameters:

```
Set Variable [$r; Value:
    MBS( "Log";
        "Client: "; $client;
        ", Record: "; Get(RecordNumber);
        ", Request: "; $request)
]
```

After debugging

You can run `MBS("Trace.Off")` to turn the trace logging off when debugging ends. Otherwise you may leave FileMaker writing log entries for weeks until someone restarts it.

And you may want to comment out the Trace call in the scripts.

Please do not hesitate to contact us if you have questions.