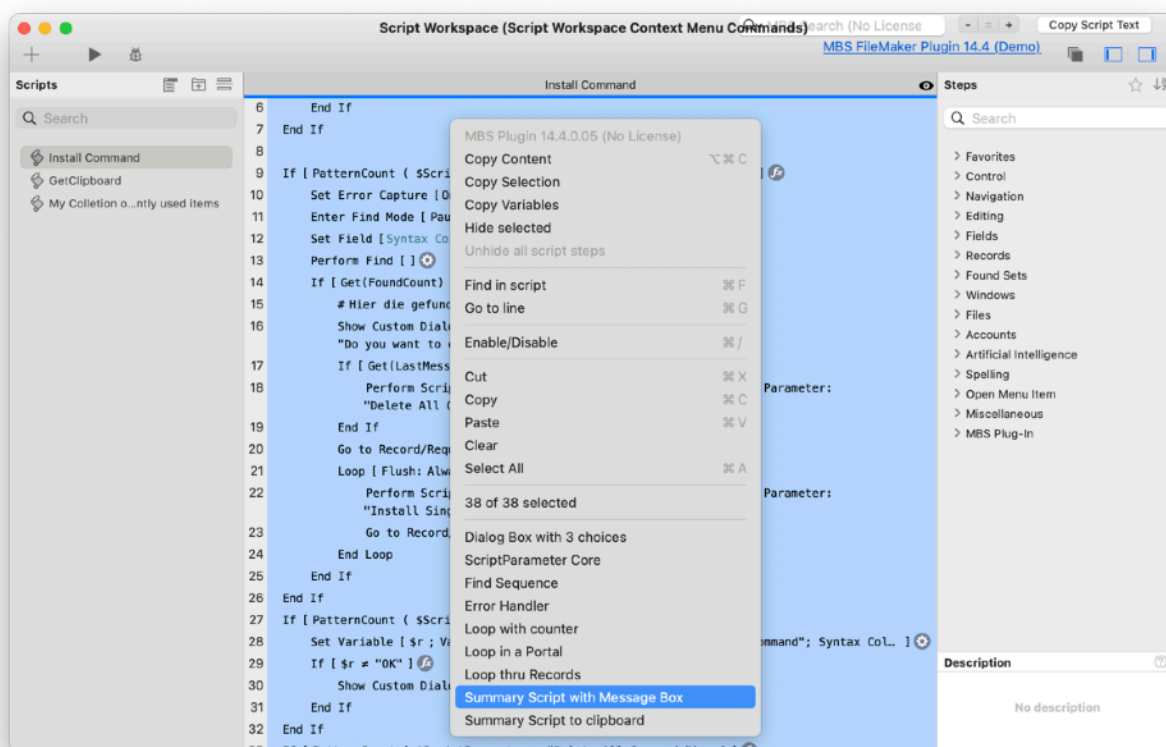


Adding ChatGPT to the ScriptWorkspace context menu

A few days ago we got a request on whether we can directly integrate ChatGPT to the ScriptWorkspace in FileMaker. Since we have already plenty of extensions to the ScriptWorkspace on macOS, we can just add a menu command for this. For Windows, the same thing could be done as a hotkey.

If you like to use this, you need to have your own API key and organization ID for ChatGPT and please insert it right into the sample code. Once you have the Let statement, you can pass it to the [SyntaxColoring.AddContextMenuCommand](#) function to add as a menu command.



To get the Let statement below, we started with a short script to use our [CURL](#) functions to query a completion via ChatGPT. We create the CURL session, set the URL and headers with the authentication details. Then we build the JSON with the request to have our prompt to summarize and the user text to process. Once we set the post content, we run the request and on success pick the output from the JSON and show it with a message box:

```

Set Variable [ $content ; Value: "Test text" ]
Set Variable [ $curl ; Value: MBS("CURL.New") ]
Set Variable [ $r ; Value: MBS("CURL.SetOptionURL"; $curl; "https://
api.openai.com/v1/chat/completions") ]
Set Variable [ $r ; Value: MBS( "CURL.SetOptionHTTPHeader"; $curl;
    "OpenAI-Organization: <<"organization>>;
    "Authorization: Bearer <<apikey>>;
    "Content-Type: application/json" ) ]
Set Variable [ $json ; Value: JSONSetElement ( "{" ;
    ["model" ; "gpt-4o-mini" ; JSONString];
    ["max_tokens" ; 100 ; JSONNumber];
    ["messages.[0].role"; "system"; JSONString];
    ["messages.[0].content"; "Please summarize what this FileMaker script does
in less than 100 words:"; JSONString];
    ["messages.[1].role"; "user"; JSONString];
    ["messages.[1].content"; $content; JSONString]) ]
Set Variable [ $r ; Value: MBS( "CURL.SetOptionPostFields"; $curl;
MBS( "CURL.SetOptionPostFields"; $curl; $json) ]
Set Variable [ $result ; Value: MBS("CURL.Perform"; $curl) ]
Set Variable [ $text ; Value: MBS("CURL.GetResultAsText"; $curl; "utf-8") ]
Set Variable [ $$debug ; Value: MBS("CURL.GetDebugMessages"; $curl) ]
Set Variable [ $r ; Value: MBS("Msgbox"; JSONGetElement ( $text ; "choices.
[0].message.content" )) ]
Set Variable [ $r ; Value: MBS("CURL.Release"; $curl) ]

```

We translate this line by line to a Let statement. This is usually a bit tricky and difficult to debug, so try in chunks. On the end we added code to run Copy command (id 57634) and get the script xml from the clipboard. That is our input for ChatGPT and we can run the same request.

Here is the big Let statement to do all:

```

Let(
    [
        r = MBS( "Menubar.RunMenuCommand"; 57634 );
        content = MBS( "Clipboard.GetFileMakerData";
"ScriptStep" );
        curl = MBS( "CURL.New" );
        r = MBS(
            "CURL.SetOptionURL";
            curl;
            "https://api.openai.com/v1/chat/completions"
        );
        r = MBS(
            "CURL.SetOptionHTTPHeader";
            curl;

```

```

        "OpenAI-Organization: <<organization>>";
        "Authorization: Bearer <<apikey>>";
        "Content-Type: application/json"
    );
    json = JSONSetElement(
        "{}";
        [
            "model";
            "gpt-4o-mini";
            JSONString
        ];
        [
            "max_tokens";
            100;
            JSONNumber
        ];
        [
            "messages.[0].role";
            "system";
            JSONString
        ];
        [
            "messages.[0].content";
            "Please summarize what this FileMaker
script does in less than 100 words:";
            JSONString
        ];
        [
            "messages.[1].role";
            "user";
            JSONString
        ];
        [
            "messages.[1].content";
            content;
            JSONString
        ]
    );
    r = MBS( "CURL.SetOptionPostFields"; curl; json );
    result = MBS( "CURL.Perform"; curl );
    text = MBS( "CURL.GetResultAsText"; curl;
"utf-8" );
    $$debug = MBS( "CURL.GetDebugMessages"; curl );
    r = MBS( "Msgbox";
        JSONGetElement( text;
            "choices.[0].message.content" ) );
    r = MBS( "CURL.Release"; curl )
];

```

```
) result
```

If you like to try this, you can of course write something to call the [SyntaxColoring.AddContextMenuCommand](#) function directly with this. Or you use our sample project “Script Workspace Context Menu Commands.fmp12”. For 14.4pr6 we include the new file with the plugin and you can just go there to install the command after adding your API key and organization ID.

Now let's go a step further and let ChatGPT write an analysis of the script with a bigger prompt. Philipp Puls from [72solutions.eu](#) got this nice prompt to create a summarization of the script and use [Clipboard.SetText](#) instead of [Msgbox](#) in the calculation to put it on the clipboard, so he can paste this as a comment into the script itself.

"Please summarize what this FileMaker script does. Structure your response so that it includes an overall summary of the purpose of the script. Follow this up with the defined variables and the customFunctions used in this script. Next build a bulleted point list, describing groups of scriptlines. These groups of lines are either delimited by the comment line starting with a '#' or by logical script-line groups like 'loop'-'end loop' or 'if'-'end if'. please reference the corresponding script lines in brackets to note what lines are referred to. Comments (starting with #) and empty lines are counted as script lines. In your interpretation consider german comments and english script lines being mixed. give the output of this analysis in german."

As you see this creates different sections, translates comments and script lines to German, adds line numbers and lists variables and custom functions. I changed output to English and run it on the Install Command script and here is a sample output:

Overview:

The provided FileMaker script is intended to manage context menu commands within a FileMaker application, leveraging the MBS plugin functions for syntax coloring and command management. The script operates based on the input provided through ` \$ScriptParameter ` , and it executes different branches to either install single commands, install active commands, delete specific commands, or delete all commands.

Defined Variables:

1. ` \$ScriptParameter ` - Contains the parameters passed to the script, used to control the script flow [Line 1].
2. ` \$r ` - Stores the result of the MBS plugin function calls, used for error handling and responses [Lines 5, 34, 52, 66].

Custom Functions:

- `MBS("SyntaxColoring.AddContextMenuCommand"...` - Adds a context menu command [Line 5].
- `MBS("SyntaxColoring.RemoveContextMenuCommand"...` - Removes a specific context menu command [Line 34].
- `MBS("SyntaxColoring.ClearContextMenuCommands"...` - Clears all context menu commands [Line 52].

Script Description:

1. ****Check for 'Install Single Command' and Process**** [Lines 2-8]:

- ****If Condition****: Checks if script parameter includes "Install Single Command".
- ****Set Variable `r` and Error Handling****: Attempts to add a single command and shows a dialog if an error occurs.

2. ****Install All Active Commands**** [Lines 12-44]:

- ****If Condition****: Looks for "Install Active Commands" in `\$ScriptParameter`.
- ****Finding Active Commands****:
 - Sets error capture, enters find mode, and performs a search for active commands [Lines 16-20].
 - ****If Condition****: Checks if any records found [Line 22].
 - ****User Confirmation and Records Loop****:
 - Displays a dialog asking whether to clear all commands first, processes based on user choice, and loops through all found records to install each [Lines 22-41].
 - Calls script "Install Single Command" for each record and navigates to the next record [Lines 26, 27, 29, 33].

3. ****Delete a Specific Command**** [Lines 46-54]:

- ****If Condition****: Checks if script parameter includes "Delete Command".
- ****Attempt Deletion and Handle Failure****:
 - Tries to delete a specified command and shows a dialog if unsuccessful [Lines 48-54].

4. ****Delete All Commands**** [Lines 56-64]:

- ****If Condition****: Identifies if the parameter includes "Delete All Commands".
- ****Attempt Clearing All Commands and Handle Errors****:
 - Executes the clear all commands function and shows an error dialog if it fails [Lines 58-64].

Each section of the script is well structured to check the command parameter, execute the specified action, and handle any errors using a combination of conditional checks and dialogue prompts. This makes the script robust for managing syntax coloring commands as part of a FileMaker application's interface customization.

Feel free to experiment yourself on this. There are a couple of things ChatGPT could do on the scripts.