



.com Solutions Inc.TM

FmPro Clipboard Explorer Manual

FmPro Clipboard Explorer Manual

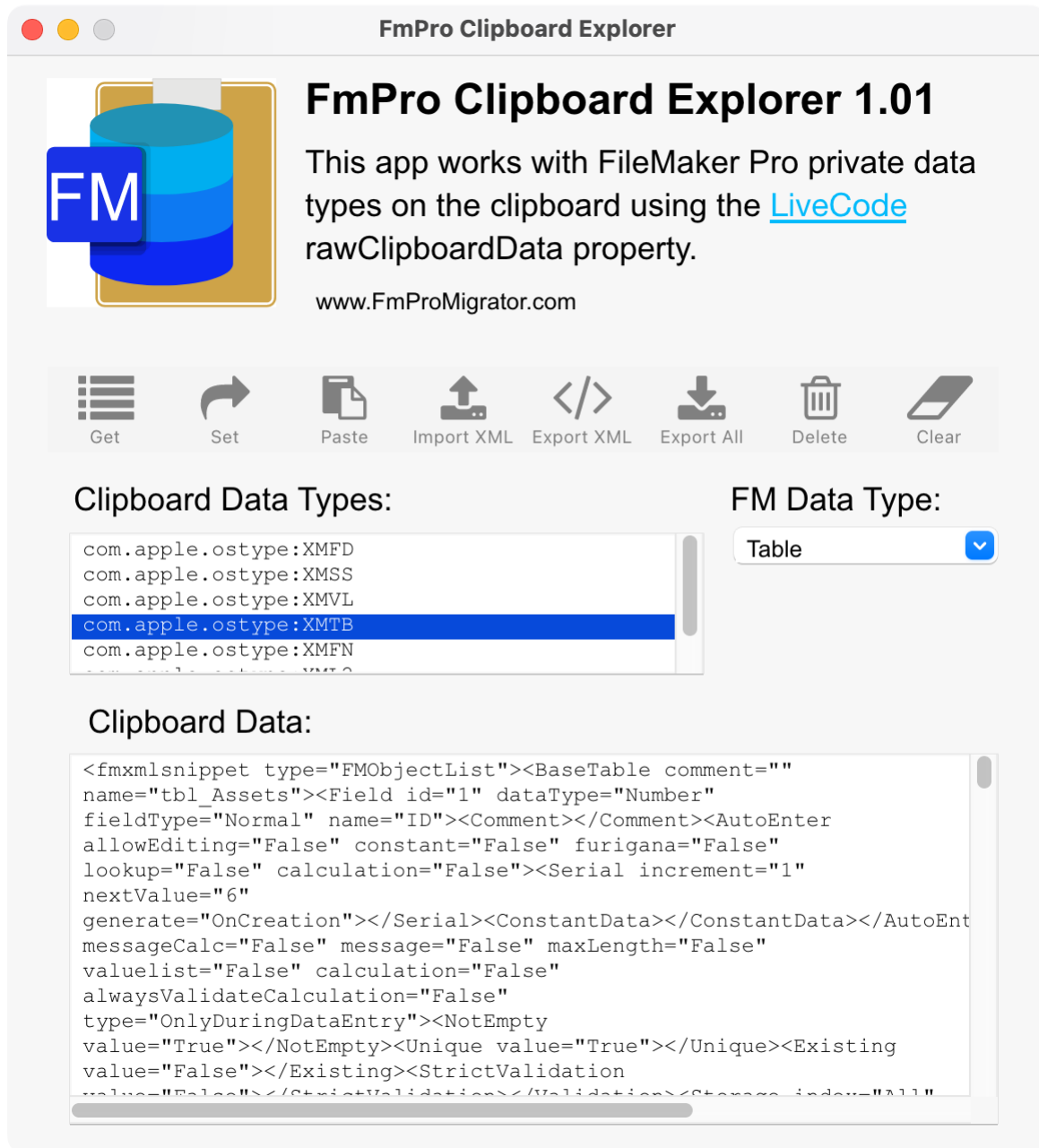
1	Overview	
1.1	Overview	4
2	Exploring the Clipboard	
2.1	Explanation of the GUI	7
3	Working with LLM Generated Code	
3.1	Pasting Code from LLMs	13
4	Source Files & Development Process	
4.1	Included Files...	16
4.2	Installing the Button Bar Widget (FCE)	18
4.3	Summary of Stack Handlers	21
4.4	Development Process	24

Overview

Overview

Initial Release for FmPro Clipboard Explorer 1.01
8/30/2025

What is the FmPro Clipboard Explorer?



FmPro Clipboard Explorer makes it easy to copy LLM generated scripts and paste them directly into the Script Workspace window on macOS and Windows.

This tool enables FileMaker developers to work with the normally hidden private data types

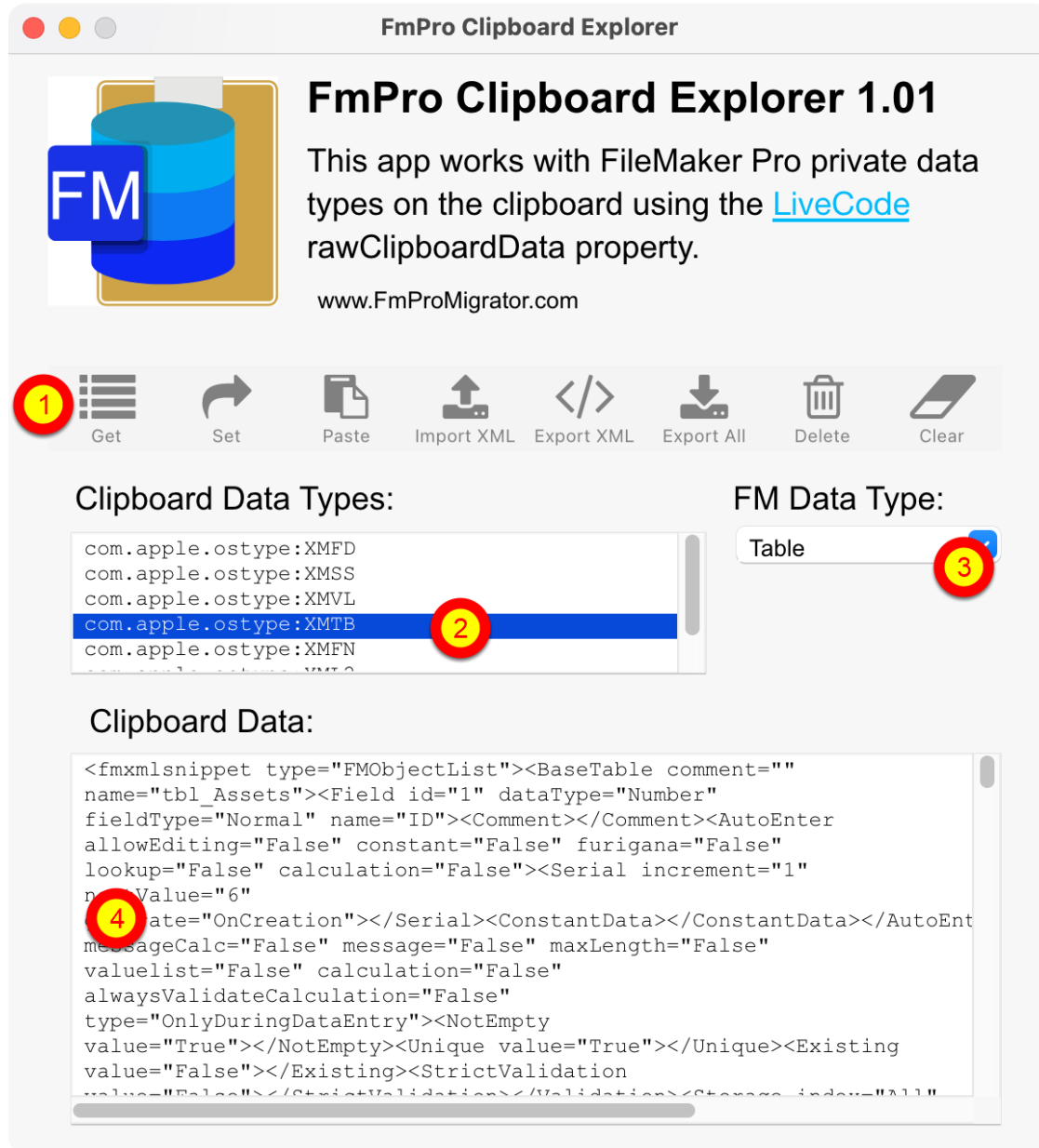
FileMaker Pro puts onto the clipboard when copying objects like layouts, value lists, custom functions, scripts/script steps and tables/fields.

This free and open source tool is able read, save, load from disk and put these private data types back onto the clipboard in a format which is usable by the FileMaker Pro database.

Exploring the Clipboard

Explanation of the GUI

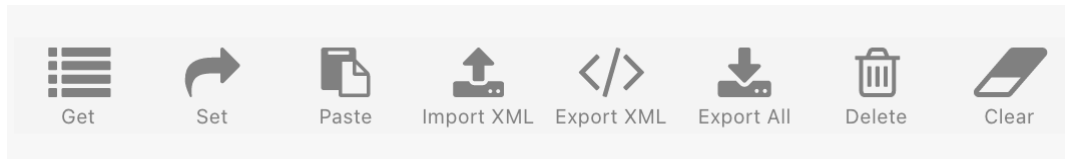
GUI Overview



The GUI consists of 4 main elements:

(1) The button bar of actions you can perform, (2) the list of private data types imported from the clipboard, (3) the FM Data Type menu, (4) the raw data from the clipboard for the data type line item currently clicked in the Clipboard Data Types field.

Button Bar Buttons



The Button Bar has the following buttons:

Get - Clicking this button replaces the current stored copy of stored clipboard data and replaces it with the current clipboard contents. The clipboard data doesn't have to be from FileMaker, though the features of this app are optimized for use with FileMaker clipboard data. On Windows, the 4 binary length bytes are removed from the XML, so that only the raw XML data is stored in memory or exported to disk.

Set - Puts data onto the clipboard using the currently stored copy of clipboard data shown in the Clipboard Data Types field. On Windows, all XML data containing the <fmxmlsnippet text gets prefixed with 4 binary bytes representing the length of the XML contents.

Paste - Imports unicode text from the clipboard placed there by another app like a web browser - using the data type selected in the FM Data Type menu. This is how to get XML data into the app from web sites, LLM chat websites and apps. No validation is being performed on the data to make sure that it is the actual type selected in the FM Data Type menu.

The clipboard data types being looked for by the Paste button include:

macOS => [public.utf8-plain-text](#)

Windows => [CF_UNICODETEXT](#)

Import XML - Clicking this button prompts for the selection of an XML file on disk, then imports the file with the data type selected in the FM Data Type menu.

Export XML - Loops thru the list of Clipboard Data Types and prompts to export each of the XML object types which contain the <fmxmlsnippet text within the XML.

Export All - Loops thru the list of Clipboard Data Types and exports them to disk. This feature is most helpful for exporting the binary screenshots FileMaker puts onto the clipboard with layouts.

Delete - Deletes the selected Clipboard Data Type.

Clear - Deletes all of the stored clipboard data which has been saved into memory within the app.

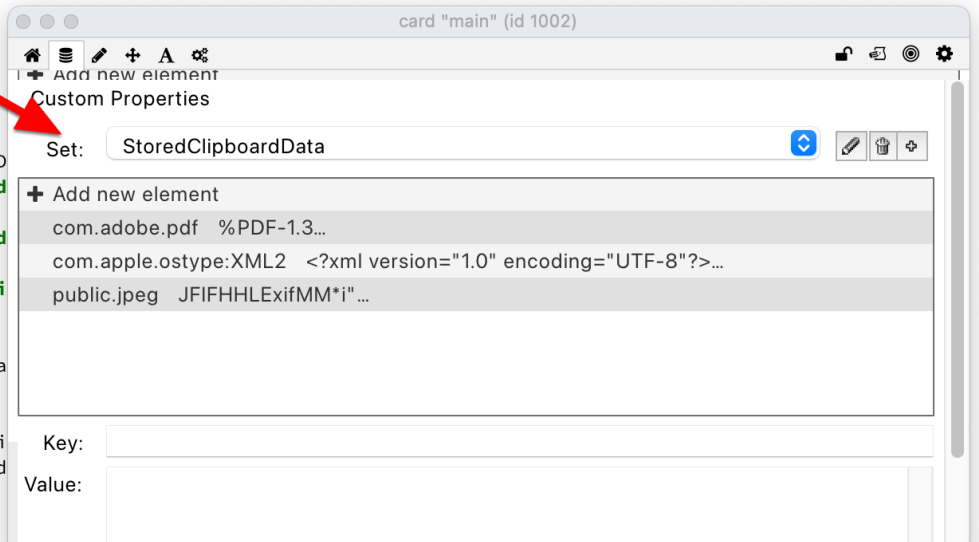
Storage of Clipboard Data

```
set the StoredClipboardData[tClipboardDataType] of card "main" to tRawClipboardData
catch tError
end try
end repeat
unlock the clipboard
end getClipboardDataTypes

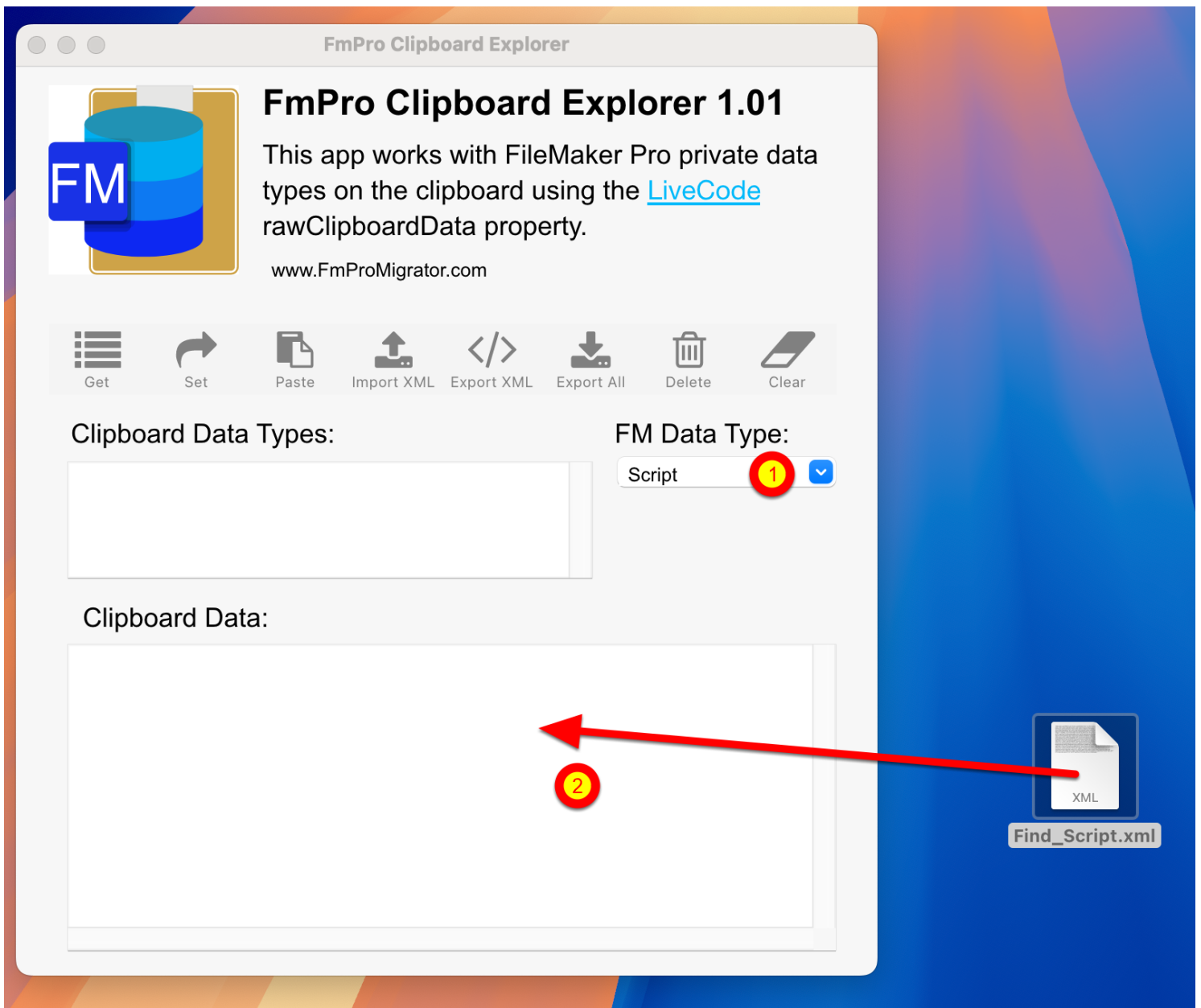
function removeLengthChars pRawClipboardData
-- on Windows only - remove the 4 leading length characters
-- Parameters:
-- pRawClipboardData - raw data passed
-- Returns:
-- tRawClipboardData - data with leading length characters removed

local tRawClipboardData
put pRawClipboardData into tRawClipboardData

-- no need to remove 4 leading length characters
if the platform is "Win32" and pRawClipboardData is a list
delete char 1 to 4 of tRawClipboardData
end if
return tRawClipboardData
end removeLengthChars
```



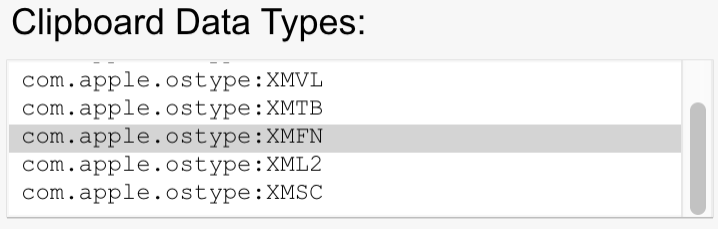
FmPro Clipboard Explorer stores its copy of clipboard data into a custom property on the card named "main" within the app. During development, this data can be seen by looking at the custom property named `StoredClipboardData` using the property inspector. Each data item is referenced via an array key which is the data type shown in the Clipboard Data Types field.



It is easy to import XML files via drag & drop by (1) first selecting the FM Data Type menu, then (2) dragging the XML file onto the Clipboard Data field. The data is then imported as the requested data type.

Why Store the Clipboard Data?

Clipboard Data Types:



```
com.apple.ostype:XMVL  
com.apple.ostype:XMTB  
com.apple.ostype:XMFN  
com.apple.ostype:XML2  
com.apple.ostype:XMSC
```

It is important for the app to store the data internally in order to make it possible to manipulate the data in the following ways:

- 1) It is possible to perform multiple XML imports and "accumulate" different data types within the tool at the same time. With the different data types shown here, just pasting into the appropriate FileMaker dialog/window will paste the correct data type for that feature of FileMaker.
- 2) Cross platform data handling. On Windows FileMaker adds a 4 byte size prefix to the XML data types. This app transparently removes that prefix and stores the plain text version of the XML internally. It also saves the plain text version of the XML to disk when exporting files.
- 3) On Windows, when placing data onto the clipboard the 4 byte size prefix is placed onto the clipboard automatically for XML data types so that FileMaker will accept the data.

Cross Platform Features

There are separate versions of the app built for macOS and Windows, with the same functionality.

On Windows, the <fmxmlsnippet text in the XML is preceded by 4 characters of size information. FmPro Clipboard Explorer transparently removes these 4 bytes when it reads the data from the clipboard, and restores the info when putting the info back onto the clipboard. This is done in order to make it easier to work with the XML files with other tools after they have been saved to disk.

Working with LLM Generated Code

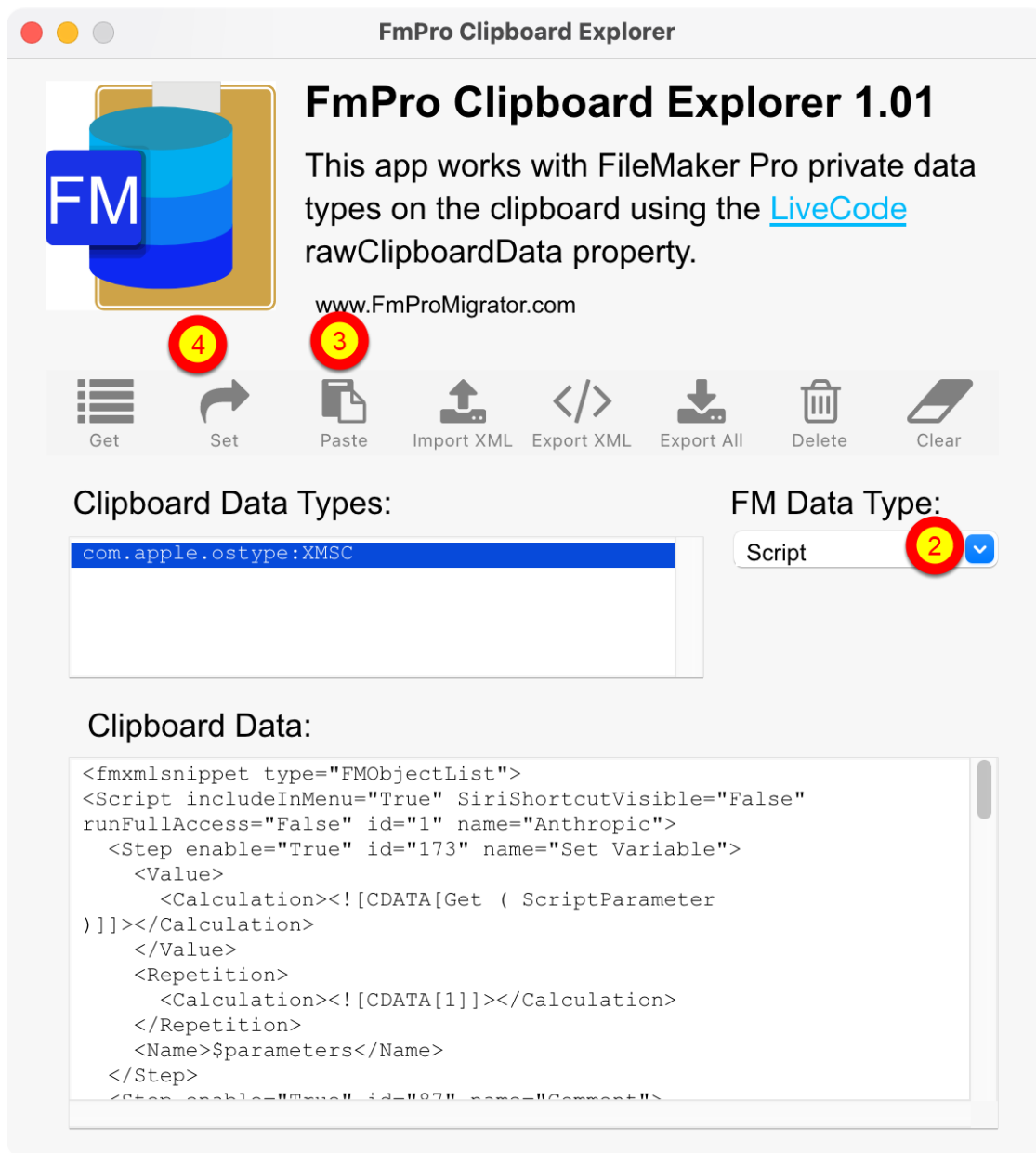
Pasting Code from LLMs

Using Anthropic Generated Scripts - Copy the Script

The screenshot shows the Claude AI web interface. On the left, a chat window titled "FileMaker Pro Script to Add Numbers" contains a prompt: "create a FileMaker Pro script which adds 2 numbers together. return filmmaker script xml". The response from Claude states: "I'll create a FileMaker Pro script that adds two numbers together. Here's the FileMaker script XML:". Below the text is a code block titled "FileMaker Add Two Numbers Script Code" which contains XML code. To the right of the chat window, the generated XML code is displayed in a larger font. The code is an FMXML snippet with two steps: a "Set Variable" step that calculates the sum of two script parameters and a "Comment" step that describes the calculation. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<fmxmlsnippet type="FMObjectList">
  <Step enable="True" id="173" name="Set Variable">
    <Value>
      <Calculation><![CDATA[Get ( ScriptParameter )]]></Calculation>
    </Value>
    <Repetition>
      <Calculation><![CDATA[1]]></Calculation>
    </Repetition>
    <Name>$parameters</Name>
  </Step>
  <Step enable="True" id="87" name="Comment">
    <Value>Extract the two numbers from the script parameter
  </Step>
  <Step enable="True" id="173" name="Set Variable">
    <Value>
      <Calculation><![CDATA[JSONGetElement ( $parameters ; '
    </Value>
    <Repetition>
      <Calculation><![CDATA[1]]></Calculation>
    </Repetition>
    <Name>$number1</Name>
  </Step>
  <Step enable="True" id="173" name="Set Variable">
    <Value>
      <Calculation><![CDATA[JSONGetElement ( $parameters ; '
    </Value>
    <Repetition>
      <Calculation><![CDATA[1]]></Calculation>
    </Repetition>
    <Name>$number2</Name>
  </Step>
  <Step enable="True" id="173" name="Set Variable">
    <Value>
      <Calculation><![CDATA[Get ( ScriptParameter )]]></Calculation>
    </Value>
    <Repetition>
      <Calculation><![CDATA[1]]></Calculation>
    </Repetition>
    <Name>$parameters</Name>
  </Step>
</fmxmlsnippet>
```

Once you generate a script via a website or app, (1) copy the script to the clipboard.



- (2) Select the proper data type from the FM Data Type menu.
- (3) Click the Paste button or use the Command+V / Control+V keys via the keyboard to paste the XML directly into the FmPro Clipboard Explorer.
- (4) Click the Set button or use the Command+C / Control+C keys via the keyboard to put the script onto the clipboard - ready for pasting into FileMaker.

Source Files & Development Process

Included Files...

FCE Files - macOS				
FCE_macOS_Folder				
Folder shared with File Sharing				
Name	^	Date Modified	Size	Kind
com.dotcomsolutionsinc.widget.buttonbar.zip		Yesterday at 11:01 AM	119 KB	ZIP archive
FmPro Clipboard Explorer 1.01.dmg		Yesterday at 8:11 PM	16.5 MB	Disk Image
FmPro Clipboard Explorer Manual.pdf		Yesterday at 8:50 PM	4.3 MB	PDF Document
FmPro_Clipboard_Explorer_101fc1.livecode		Yesterday at 8:08 PM	58 KB	LiveCode Stack
icon		Yesterday at 11:44 AM	--	Folder
FCE_icon_1024_1024_01.afdesign		May 7, 2025 at 1:09 PM	29 KB	Affinity Openable
FCE_icon_1024_1024_01.icns		Jul 1, 2025 at 6:55 AM	99 KB	Apple icon image
FCE_icon_1024_1024_white_bkgd_01.afdesign		May 9, 2025 at 1:10 PM	30 KB	Affinity Openable
FCE_icon_1024_1024_white_bkgd_01.png		May 7, 2025 at 1:31 PM	33 KB	PNG image
FCE_icon_1024_1024_white_bkgd_01.svg		May 7, 2025 at 1:15 PM	4 KB	SVG

The files provided with the macOS package include:

com.dotcomsolutionsinc.widget.buttonbar.zip - This is the open source button bar widget used for the main toolbar.

FmPro Clipboard Explorer 1.01.dmg - The digitally signed/notarized macOS app.

FmPro Clipboard Explorer Manual.pdf - The PDF manual.

FmPro_Clipboard_Explorer_101fc1.livecode - This is the self-contained stack containing the entire application, built with the LiveCode development environment. LiveCode website:

www.LiveCode.com

icon folder:

FCE_icon_1024_1024_01.afdesign - The Affinity Designer file used to create the application icon.











FCE_icon_1024_1024_01.icns - This macOS icon uses the version of the icon file without the white background.

FCE_icon_1024_1024_white_bkgd_01.afdesign - The Affinity Designer file used to create the SVG icon used within the app.

FCE_icon_1024_1024_white_bkgd_01.png - This PNG isn't used within the app.

FCE_icon_1024_1024_white_bkgd_01.svg - The SVG icon used and sized down to 128 x 128 for the main screen of the app.

FCE Files - Windows

FCE_Windows_Folder				
Folder shared with File Sharing				
Name		Date Modified	Size	Kind
 com.dotcomsolutionsinc.widget.buttonbar.zip		Yesterday at 11:01 AM	119 KB	ZIP archive
 FmPro Clipboard Explorer 1.01.zip		Today at 11:42 AM	10.1 MB	ZIP archive
 FmPro Clipboard Explorer Manual.pdf		Yesterday at 8:50 PM	4.3 MB	PDF Document
 FmPro_Clipboard_Explorer_101fc1.livecode		Today at 11:34 AM	58 KB	LiveCode Stack
▼  icon		Yesterday at 11:44 AM	--	Folder
 FCE_icon_1024_1024_01.afdesign		May 7, 2025 at 1:09 PM	29 KB	Affinity...enable
 FCE_icon_1024_1024_01.ico		Jul 1, 2025 at 6:55 AM	41 KB	Windo...n image
 FCE_icon_1024_1024_white_bkgd_01.afdesign		May 9, 2025 at 1:10 PM	30 KB	Affinity...enable
 FCE_icon_1024_1024_white_bkgd_01.png		May 7, 2025 at 1:31 PM	33 KB	PNG image
 FCE_icon_1024_1024_white_bkgd_01.svg		May 7, 2025 at 1:15 PM	4 KB	SVG

The files provided with the Windows package include:

com.dotcomsolutionsinc.widget.buttonbar.zip - This is the open source button bar widget used for the main toolbar.

FmPro Clipboard Explorer 1.01.zip - The digitally signed 64-bit Windows app.

FmPro Clipboard Explorer Manual.pdf - The PDF manual.

FmPro_Clipboard_Explorer_101fc1.livecode - This is the self-contained stack containing the entire application, built with the LiveCode development environment. LiveCode website: www.LiveCode.com

icon folder:

FCE_icon_1024_1024_01.afdesign - The Affinity Designer file used to create the application icon.

FCE_icon_1024_1024_01.ico - This Windows icon uses the version of the icon file without the white background.

FCE_icon_1024_1024_white_bkgd_01.afdesign - The Affinity Designer file used to create the SVG icon used within the app.

FCE_icon_1024_1024_white_bkgd_01.png - This PNG isn't used within the app.

FCE_icon_1024_1024_white_bkgd_01.svg - The SVG icon used and sized down to 128 x 128 for the main screen of the app.

Installing the Button Bar Widget (FCE)

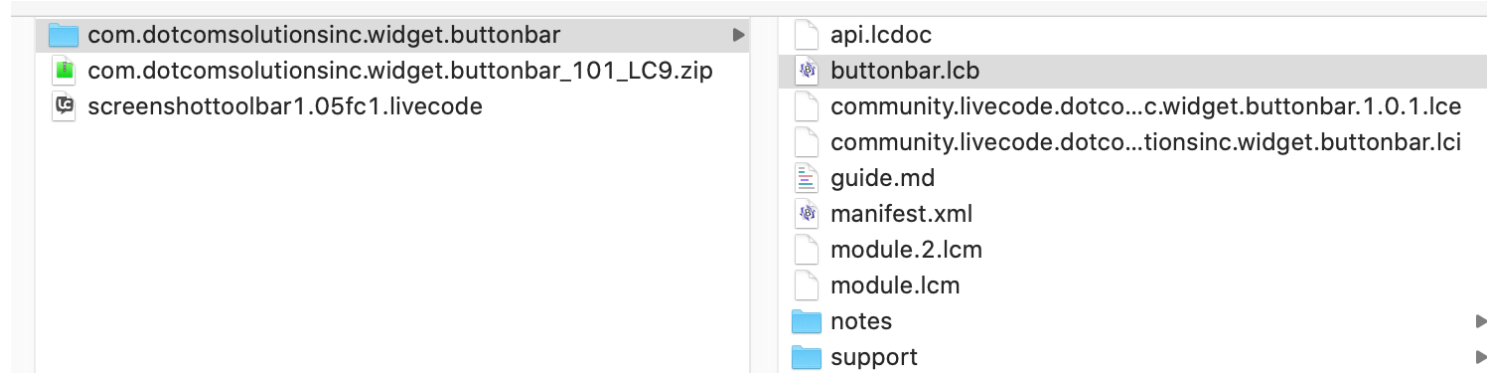
The .com Solutions Inc. developed Button Bar widget is used for the main toolbar of the app. But this widget needs added to the LiveCode IDE before it will be displayed within the stack from within the LiveCode IDE. Follow these instructions to install the Button Bar widget.

Missing Button Bar Widget



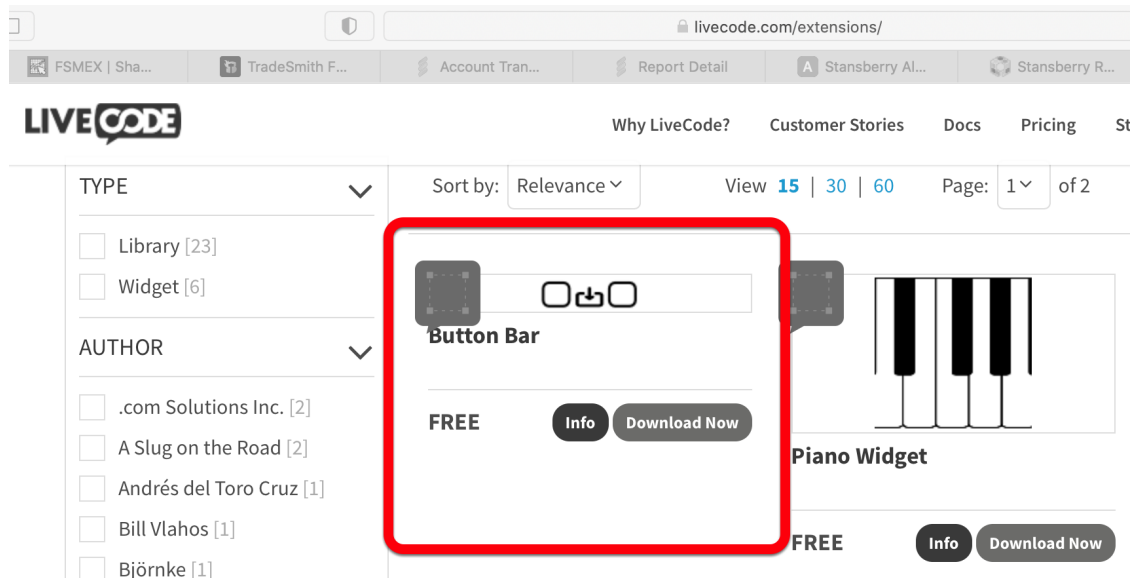
If the Button Bar widget isn't installed before opening the FmPro Clipboard Explorer LiveCode stack, the buttons and text labels below them shown within the highlighted area of this image will be missing - because they are created by the widget.

Install Button Bar Widget - Downloaded Folder



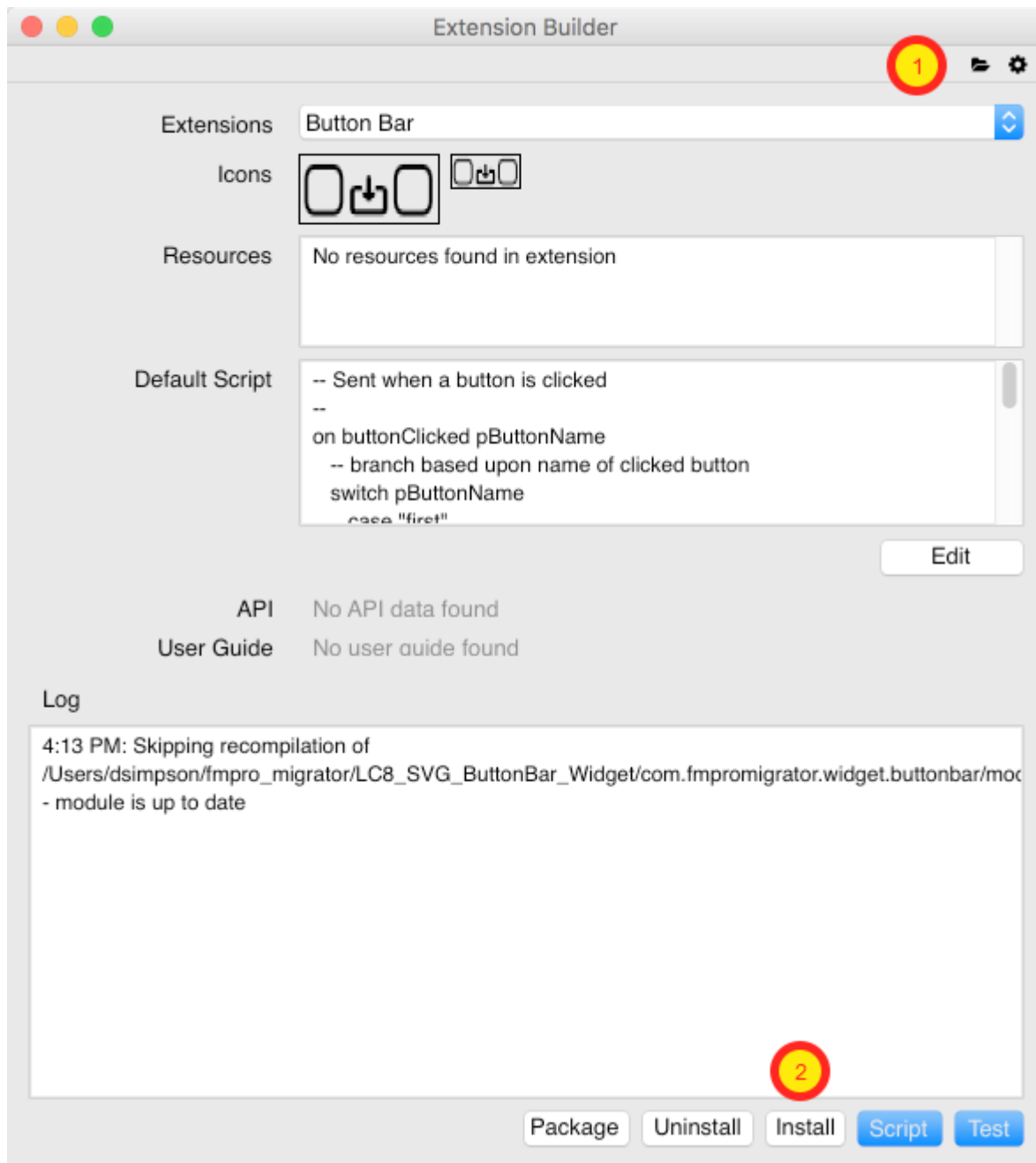
The Button Bar widget is included in the zip folder containing the FmPro Clipboard Explorer stack file. Unzip the widget so that the buttonbar.lcb file can be selected when installing with the Extension Builder.

Alternative Option: Download Button Bar Widget - LiveCode Store Web Site



The latest version of the Button Bar widget can also be downloaded from the LiveCode store at <https://livecode.com/extensions/>
Download and unzip the Button Bar widget folder.

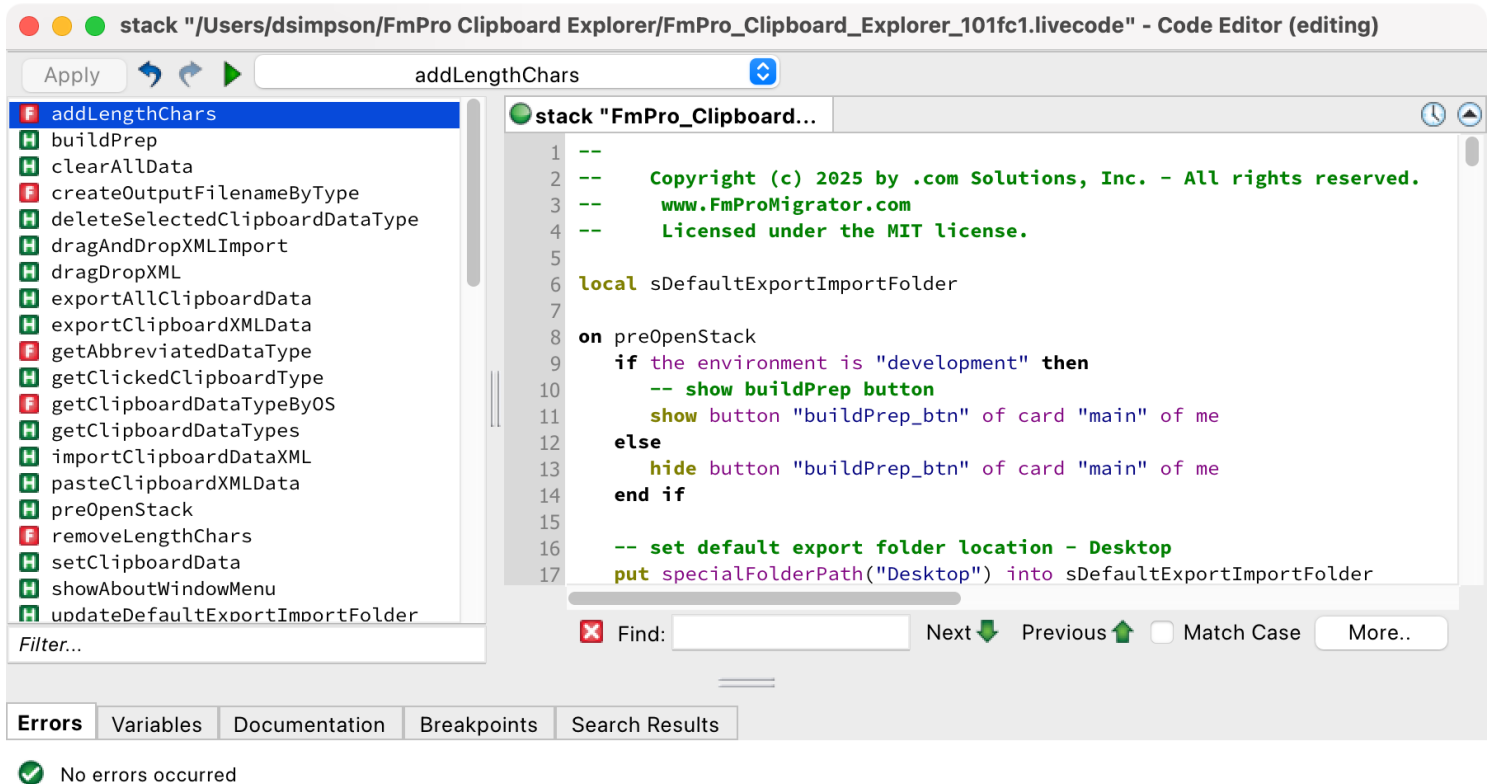
Extension Builder Window



From within the LiveCode IDE, select the Tools -> Extension Builder menu item. (1) Click the folder icon in the upper right corner of the Extension Builder window, select the buttonbar.lcb file from the Button Bar widget folder. (2) Click the Install button. Quit and re-open the LiveCode IDE so that the dictionary entries are displayed for the Button Bar widget.

Summary of Stack Handlers

Stack Script Handler Summary



preOpenStack - Initializes the application on startup by setting the default import/export folder to the Desktop and managing the visibility of a build preparation button based on the environment.

buildPrep - Prepares the application for distribution by setting standalone application settings for both macOS and Windows, such as the app name and version number which is read from the text of the title_lbl field at the top of the app window. It also clears data fields to their default state.

dragDropXML - Accepts a single XML file via drag and drop and passes its path to the dragAndDropXMLImport handler for import processing. This handler is triggered by doing a drag & drop of an XML file anywhere on the main window of the app.

getClipboardDataTypes - Reads all data types from the system clipboard, lists them in a field, and saves the raw data for each type into a the [StoredClipboardData](#) custom property for later use.

removeLengthChars - A function that removes the four leading length characters from raw

clipboard data if the platform is Windows and the data is XML.

addLengthChars - A function that adds four leading length characters as a prefix to raw XML data when the platform is Windows.

getClickedClipboardType - Displays the raw data in the Raw Clipboard Data field after a user clicks on a specific clipboard data type from the list.

setClipboardData - Clears the system clipboard and repopulates it with all the data types and their corresponding data currently stored in the [StoredClipboardData](#) custom property.

updateSelectedClipboardCprop - Updates the [StoredClipboardData](#) custom property data for the selected clipboard type with new data that has been edited in the data field.

getClipboardDataTypeByOS - A function that returns the correct FileMaker-specific clipboard data type name based on the operating system (macOS or Windows).

getAbbreviatedDataType - A function that converts a technical clipboard data type (e.g., "XMSC") into a simplified, human-readable name (e.g., "Script").

updateDefaultExportImportFolder - Sets the default folder for file dialogs to the path of the most recently used file, removing the filename itself.

exportClipboardXMLData - Finds the first available XML data type from the stored clipboard information, prompts the user to save it as a file, and then exports it. It assumes only one XML data type will be present, as is typical with FileMaker.

exportAllClipboardData - Prompts the user for a base filename and saves all stored clipboard data types to individual files, creating a unique filename for each file.

importClipboardDataXML - Prompts the user to select an XML file, reads its contents, and stores the data in the [StoredClipboardData](#) custom property under the currently selected data type.

dragAndDropXMLImport - Processes a dropped file, checks if it has an "xml" extension, and if so, imports its data into the application under the currently selected data type.

pasteClipboardXMLData - Pastes text data from the system clipboard and stores it in the application under the currently selected data type.

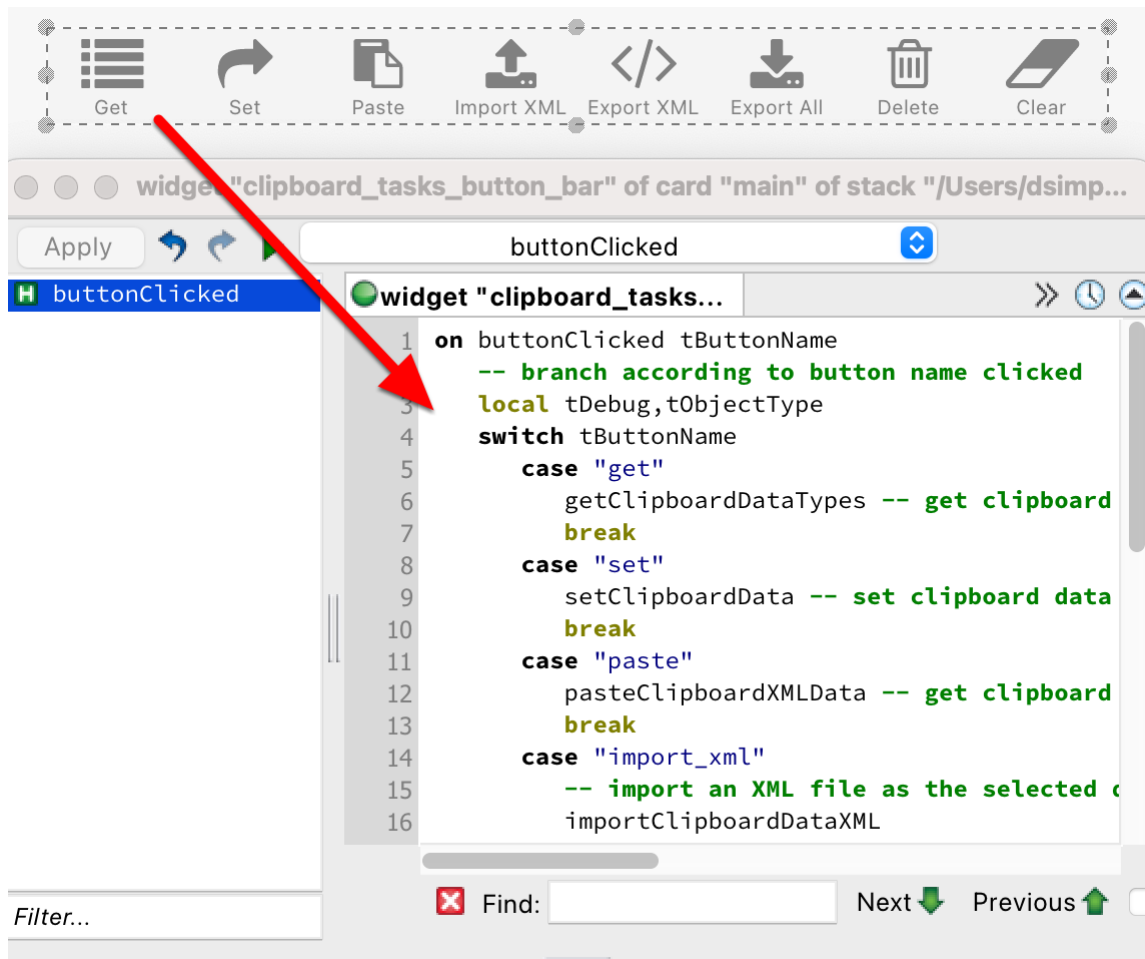
createOutputFilenameByType - A function that generates a filename for export by taking a base name, removing any existing file extension, and appending a new extension based on the clipboard data type.

deleteSelectedClipboardDataType - Deletes the currently highlighted data type from the the [StoredClipboardData](#) custom property and updates the display.

clearAllData - Clears the [StoredClipboardData](#) custom property data and field contents, with an option to skip the confirmation prompt for non-interactive use (for instance when the buildPrep script is run).

showAboutWindowMenu - Opens a separate stack to display the application's "About" window.

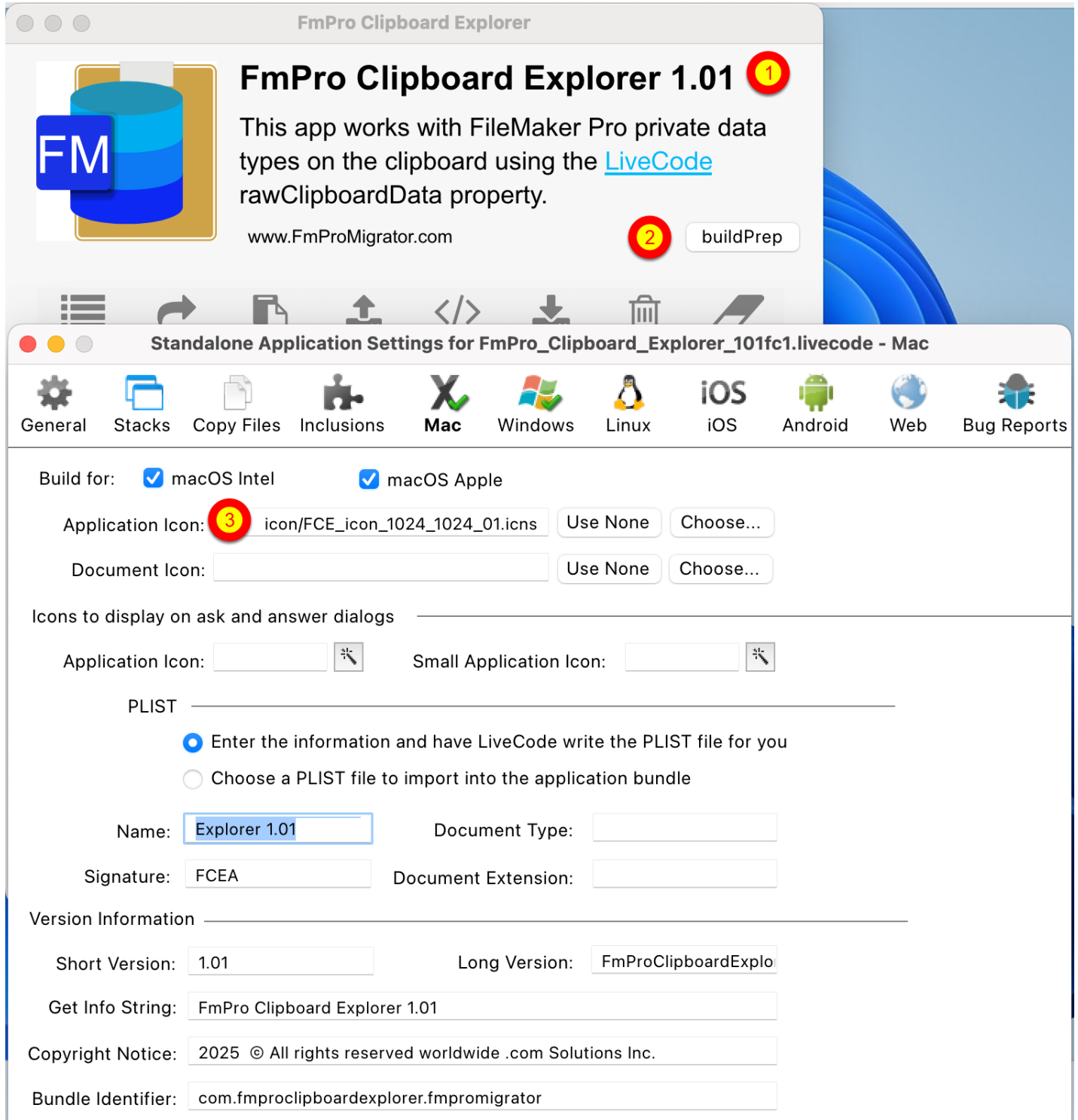
Button Bar Script



The buttonClicked handler of the button bar widget calls the stack level handlers shown above.

Development Process

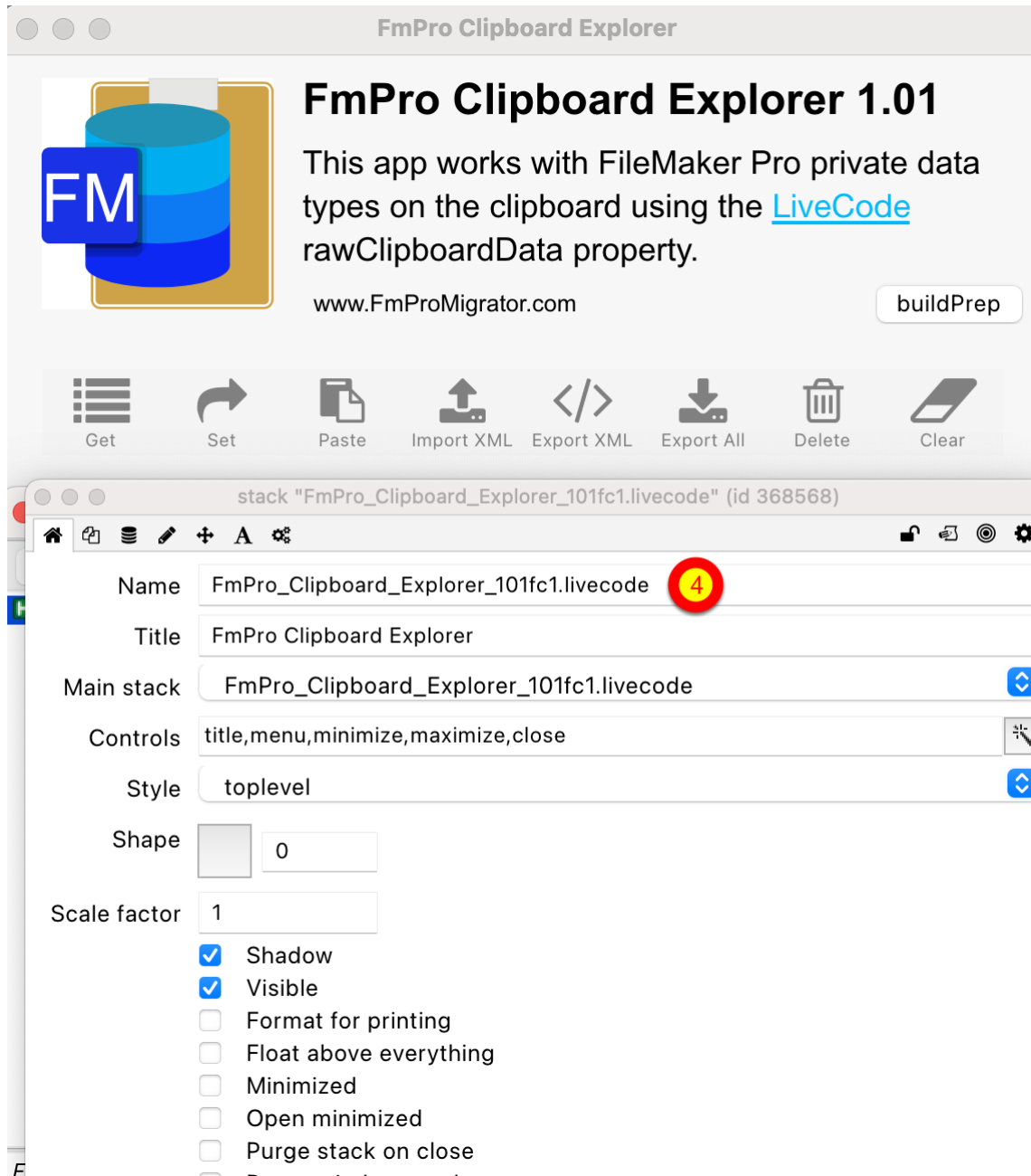
Change Version Info



When developing a new version of app, (1) change the version number in the title_lbl field at the top of the window, then (2) click the buildPrep button to update the version into in the standalone settings.

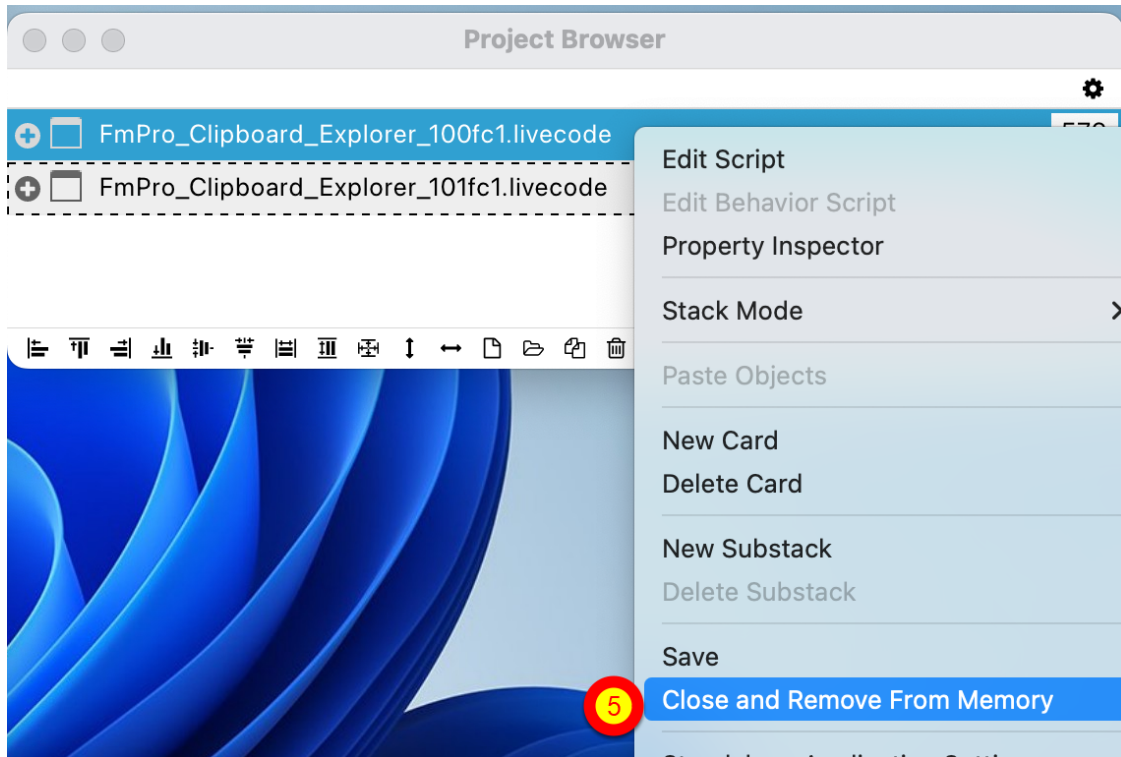
(3) Select the icon file for each platform for which you want to build standalone apps (macOS & Windows for instance).

Save Stack



Use Command/Contrl+K to open the stack properties, (4) rename and save the stack under a different name.

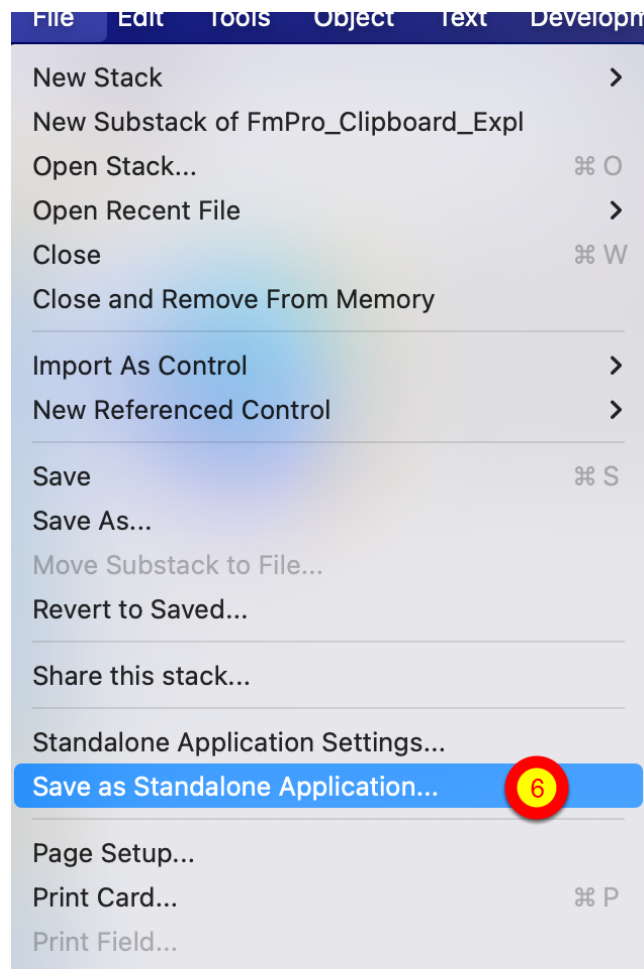
Close & Remove Old Stack - Using Project Browser



Open the Tools -> Project Browser, right click on the old version of the stack, (5) select Close and Remove From Memory.

This needs done in order to prevent you and LiveCode from being confused about which version of the app you are working with.

Build the App



Select the File -> Save as Standalone Application... menu to build the app.