.com Solutions Inc.

Layout Conversion Workbench Manual

Layout Conversion Workbench Manual

About

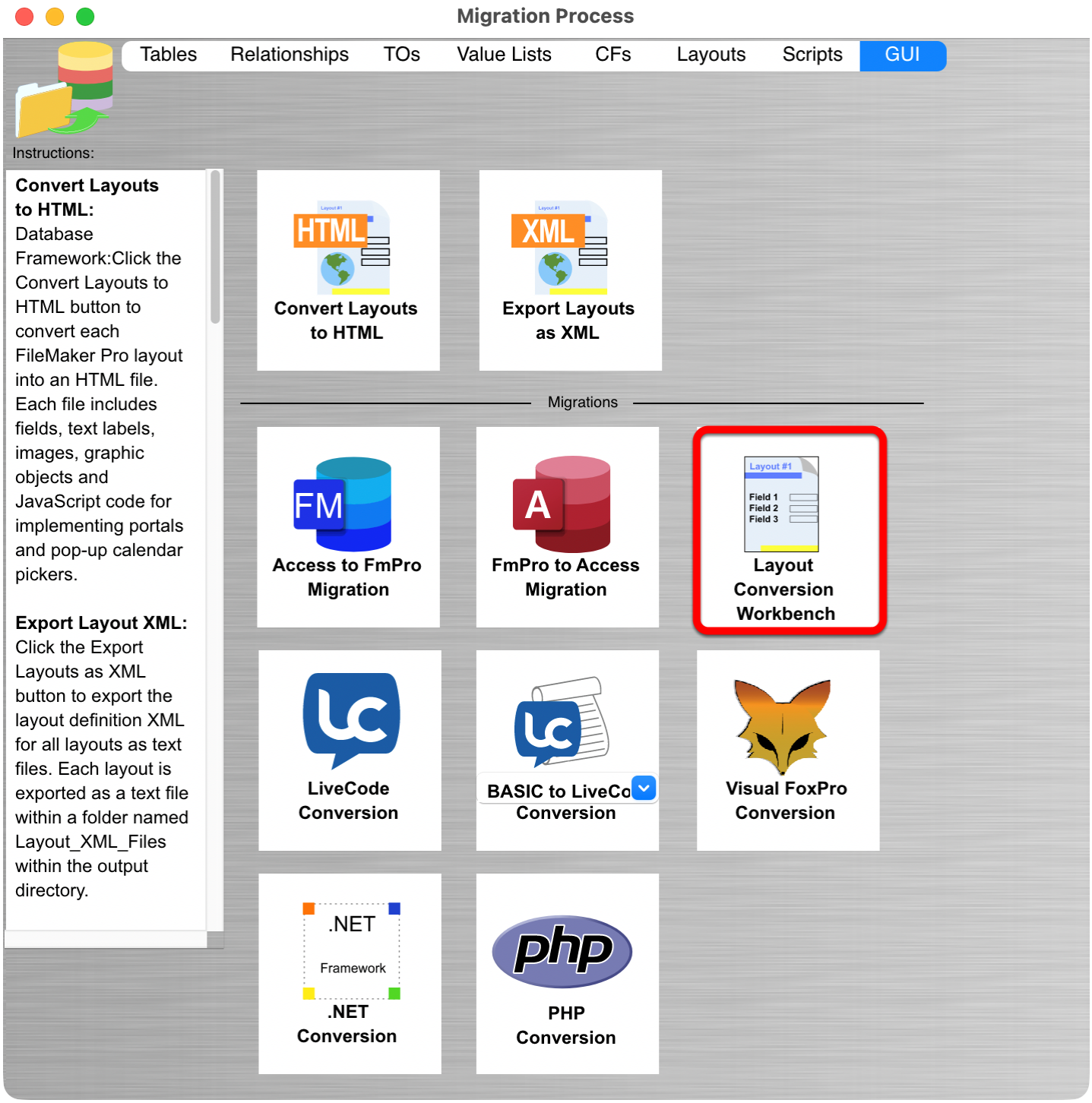
About the Layout Conversion Workbench Manual

FmPro Migrator is a cross-platform application designed for converting database data, scripts and GUI interface (Layouts/Forms/Reports) between database development environments. Screenshots in this manual will include macOS and Windows versions of the application as the Layout Conversion Workbench features work the same across both platforms.

Revision 1
FmPro Migrator 11.84
5/22/2026

Overview

Overview - Layout Conversion Workbench

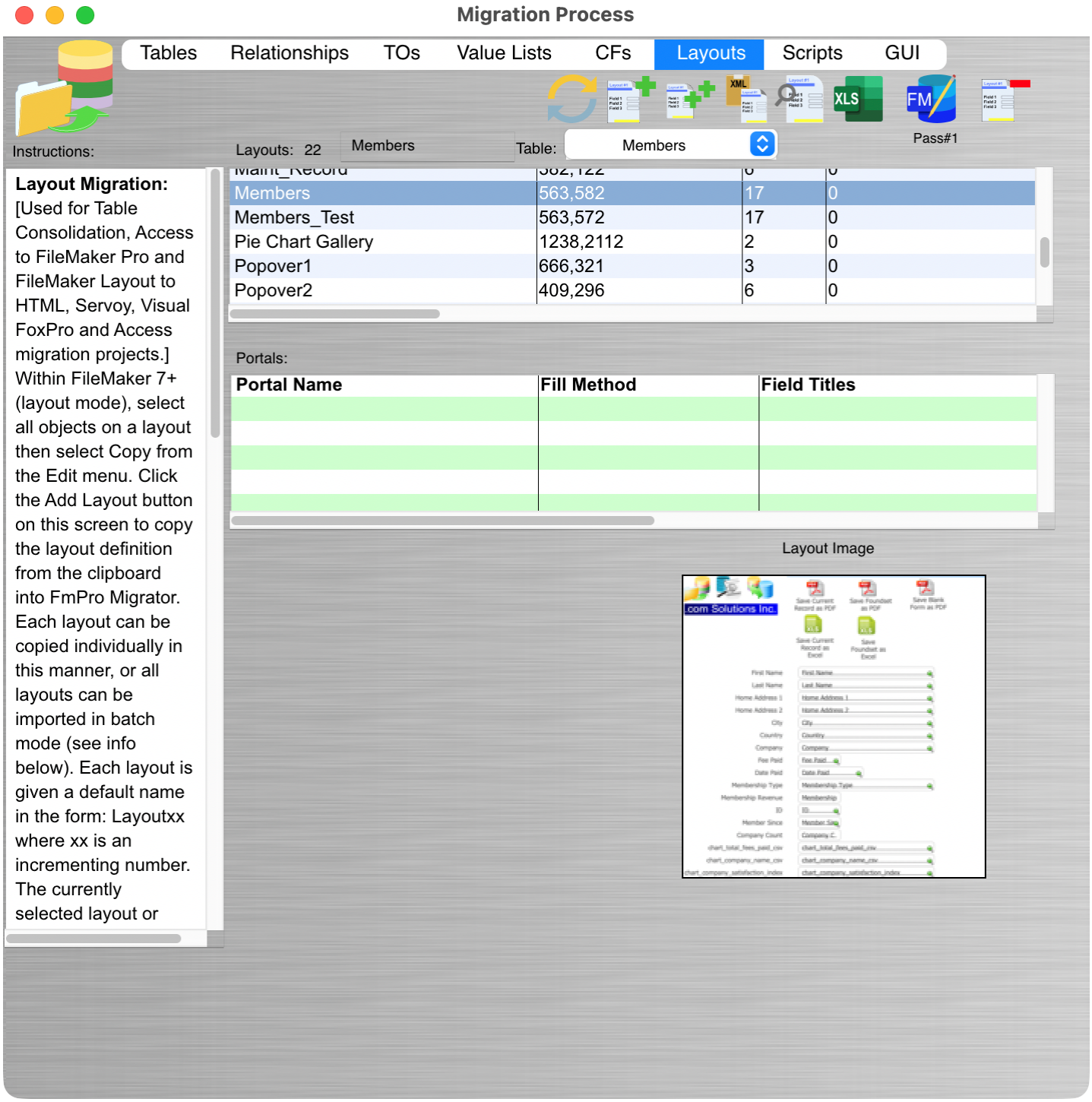


The Layout Conversion Workbench automates the time consuming task of manually rebuilding Layouts/Forms/Reports for a variety of programming languages and frameworks.
[Example: Convert Visual FoxPro Forms/Reports into .NET C# XAML for further development in Visual Studio]
This feature is opened by clicking the Layout Conversion Workbench button on the GUI tab of the

Migration Process window in FmPro Migrator.
The Layout Conversion Workbench represents a complete re-design of the conversion features implemented on the GUI tab. The implementation of GUI conversion features has typically required many months and hundreds of hours of complex development effort. Now, with the implementation of the Layout Conversion Workbench, this development effort can be reduced to typically a few weeks. Further advantages include making the process of updating to new framework versions a much more efficient task which can now be performed either by .com Solutions Inc. or by customers.

GUI

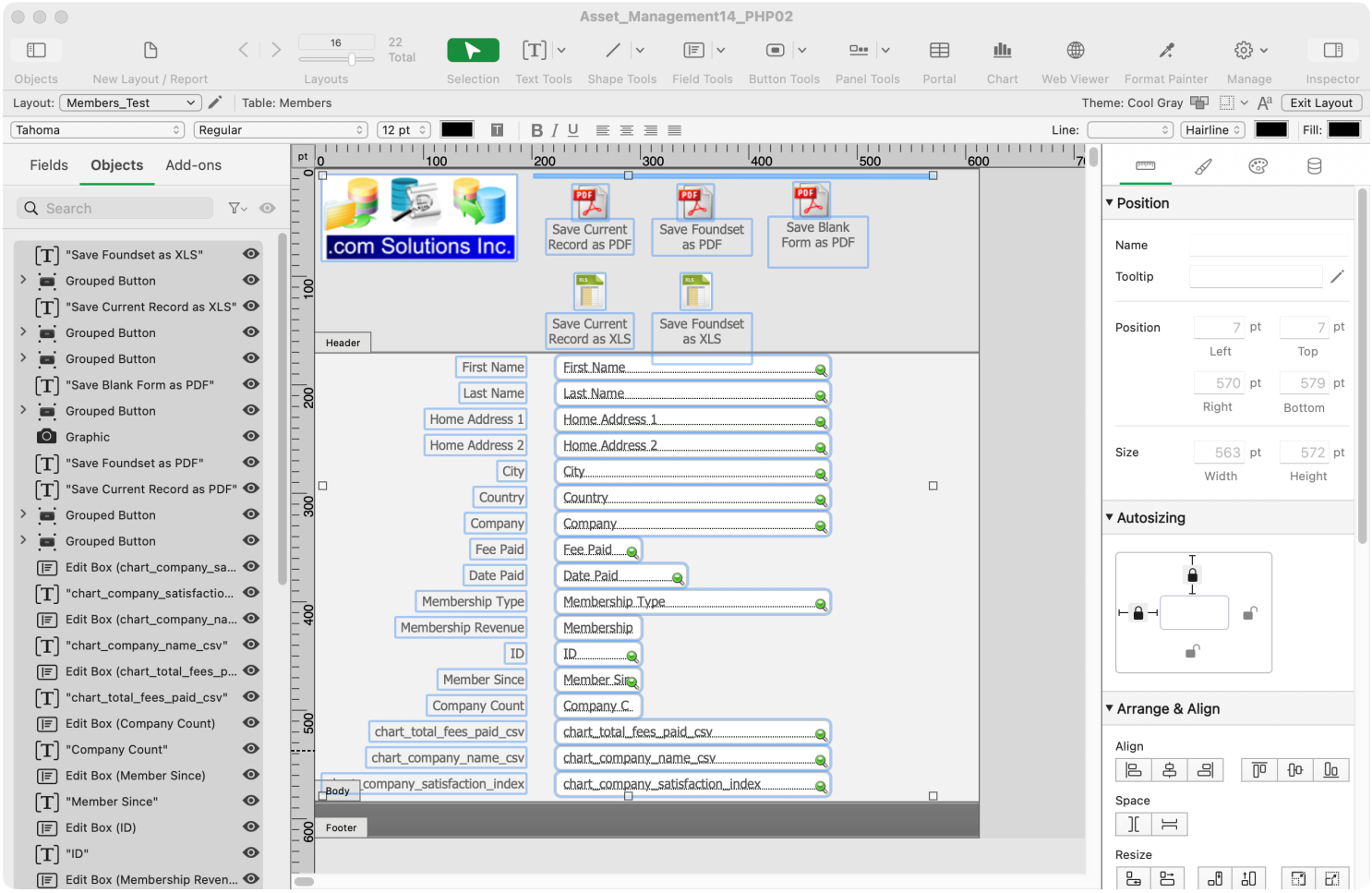
Layouts Tab - Migration Process Window



Alayout is the GUI representation of a FileMaker layout or a Form/Report from Visual FoxPro or Microsoft Access database. The underlying structure of a layout consists of an XML description of the objects being displayed (text labels, fields, grids, portals and embedded images).

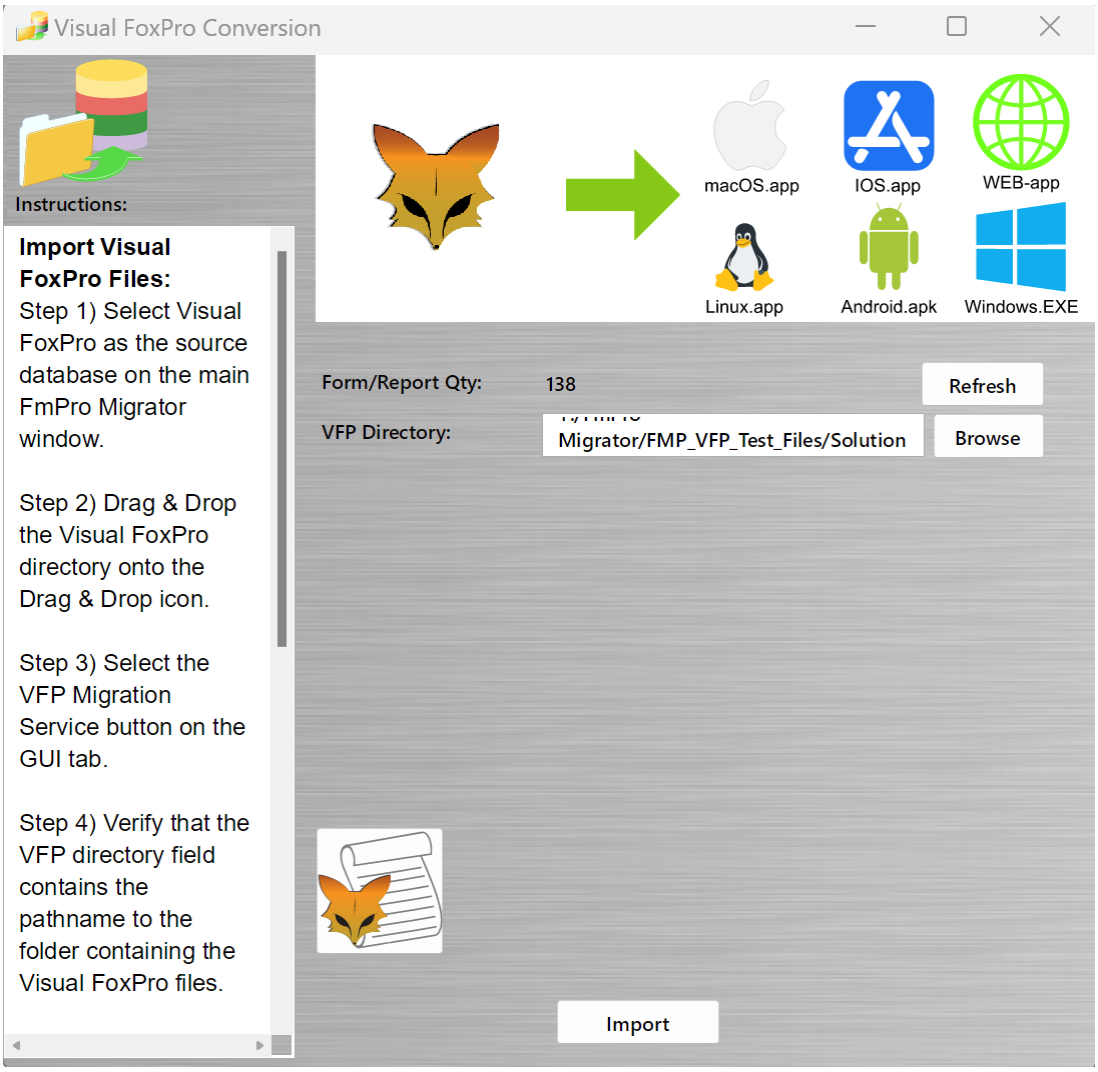
At the present time, layout thumbnail images are only created for FileMaker layouts.

FileMaker XML - Clipboard Importing



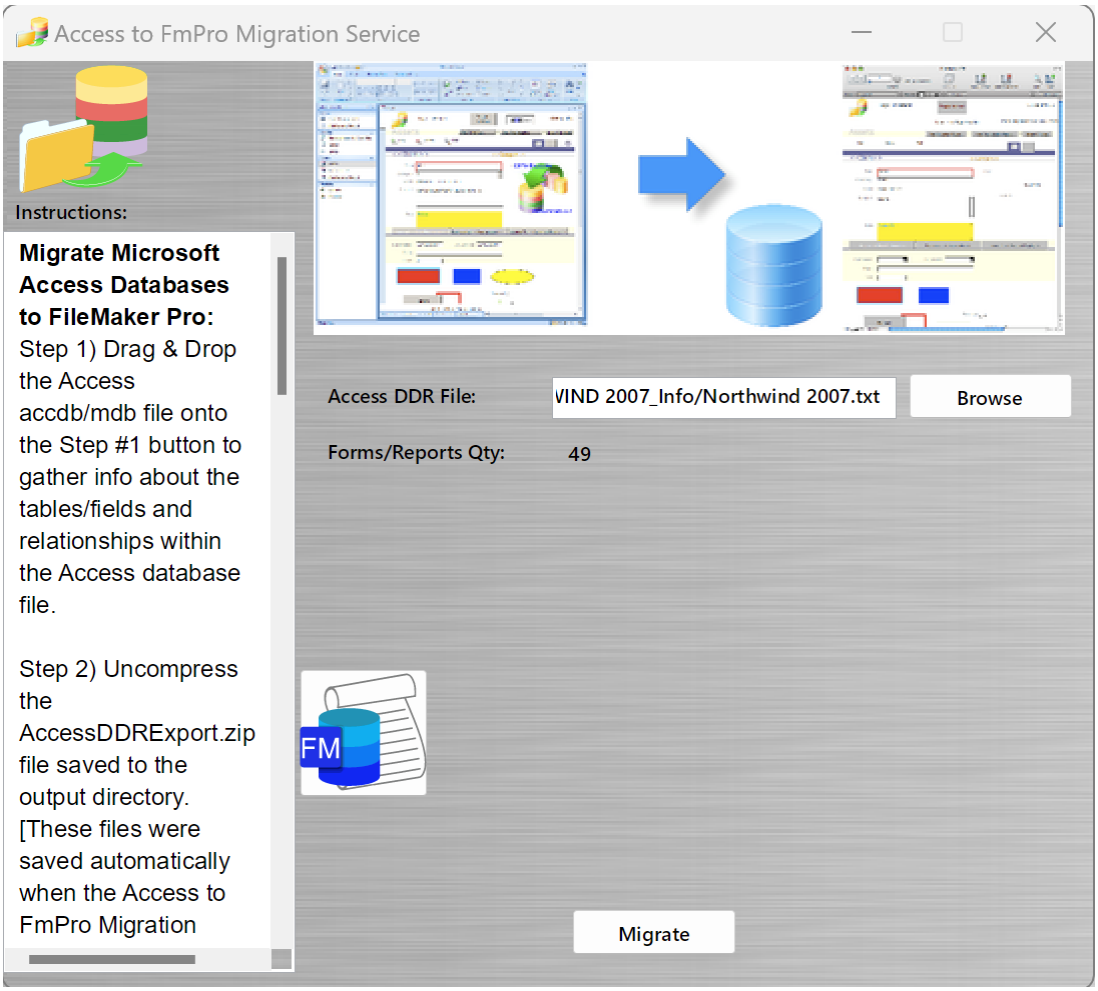
For FileMaker databases, the layout XML is imported via the clipboard via the Batch Import All Layouts button on the Layouts tab of the Migration Process window.
The free FmPro Clipboard Explorer tool can be used to explore the various FileMaker objects exchanged via the clipboard.

Visual FoxPro - Layout XML Creation During Import



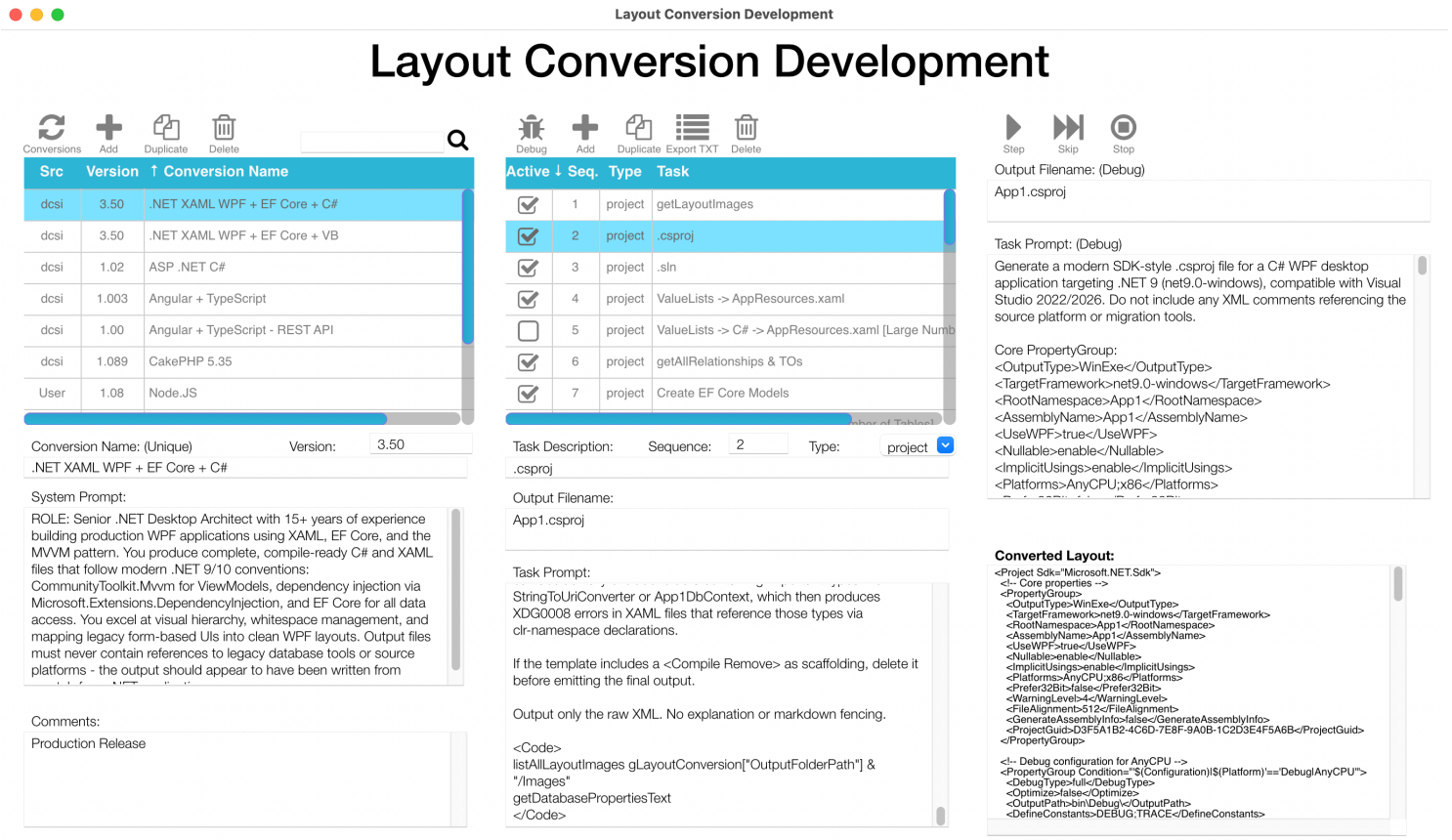
For Visual FoxPro projects, FmPro Migrator creates layout XML for each Form/Report when importing the metadata from the VFPExport.db3 file created by the included VFPExport.EXE utility.

Microsoft Access - Layout XML Creation During Import



For Microsoft Access databases, layout XML is created by FmPro Migrator when importing the metadata created by the included AccessDDRExport utility.

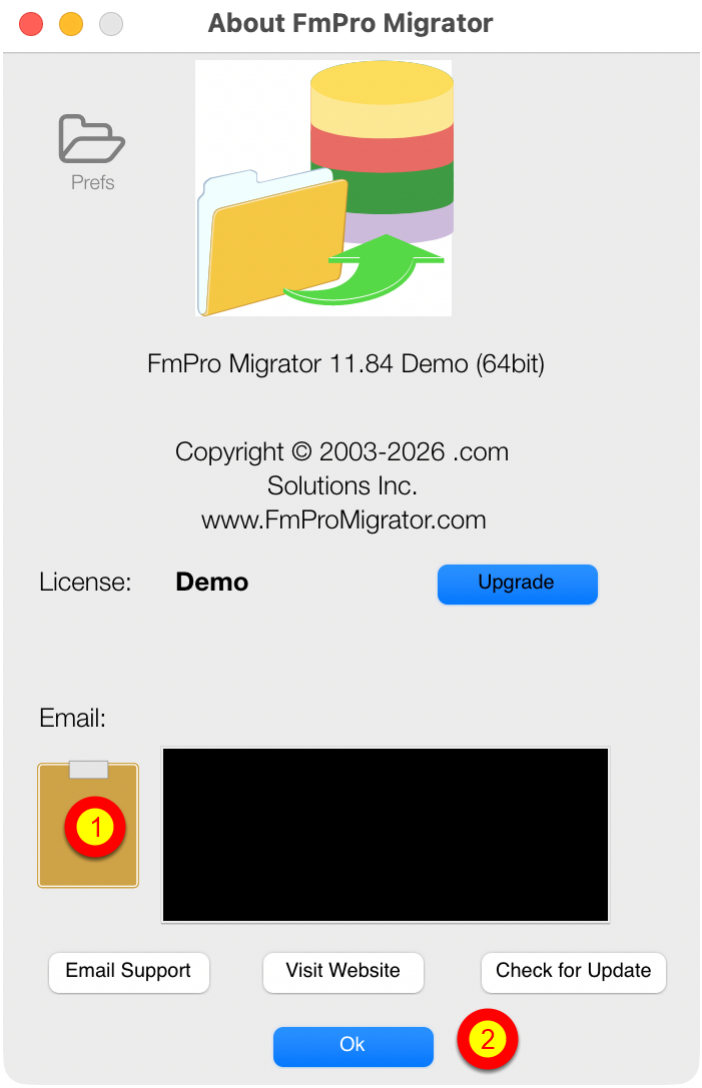
Layout Conversion Workbench - Development Window Screenshot



The top-left grid on the Development window shows the list of available layout conversions. The .NET XAML WPF + EF Core + C# conversion is currently selected in the grid. Each conversion includes one or more Tasks which actually perform the conversion by sending the prompt text to your choice of LLM vendor and model for processing. Depending upon your FmPro Migrator license, you can use either cloud based LLMs or local LLMs (when privacy & security are of paramount importance).

For the .NET XAML C# conversion shown here, there are 30 Tasks which are run in order to complete the entire conversion process.
Tasks are preformed for either the "project" or the "layout". Project tasks, run only once for the entire project. Layout tasks, run for each layout in the project.
The currently selected .csproj task creates the Visual Studio .csproj file based upon information and metadata sent in the prompt. The last part of the prompt includes a <Code> block which outputs a de-duplicated list of images used in the layouts of the project, followed by the database properties selected on the main screen of FmPro Migrator, and the list of database tables in the project. All of this information is put together by the LLM to create a file named App1.csproj - ready to be opened by Visual Studio.

Enter FmPro Migrator License key



Select the Help -> License window and (1) paste in the license key received in your purchase email into the License field or click the clipboard icon. (2) Click the Ok button.

Follow the appropriate database importing PDF manual as shown below:

FileMaker Pro - Import the FileMaker Database into FmPro Migrator

The How to Import FileMaker Pro Databases into FmPro Migrator PDF manual provides importing details.

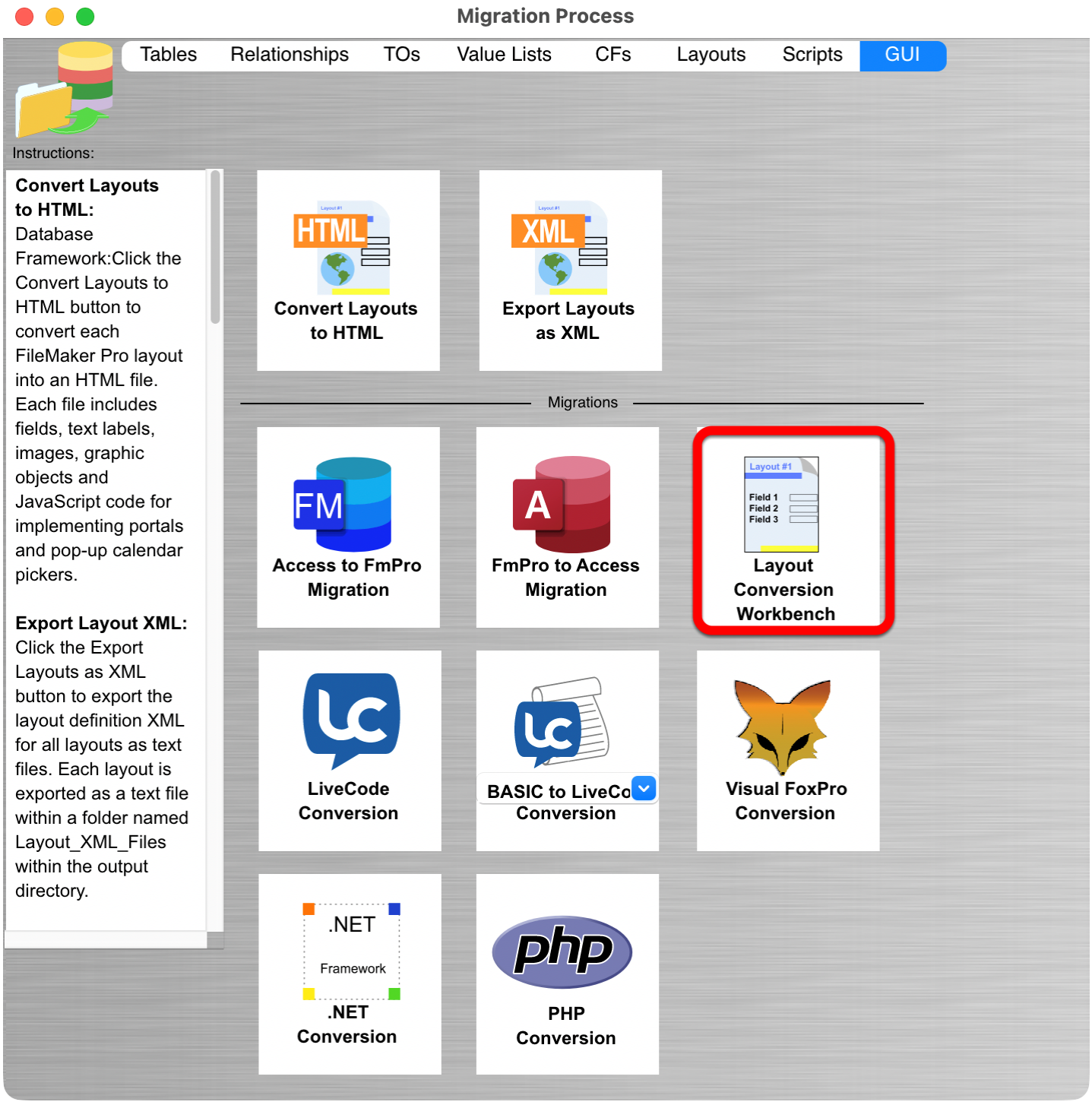
FileMaker Pro - Import the Microsoft Access Database into FmPro Migrator

The How to Import Microsoft Access Databases into FmPro Migrator PDF manual provides importing details.

FileMaker Pro - Import the Visual FoxPro Project into FmPro Migrator

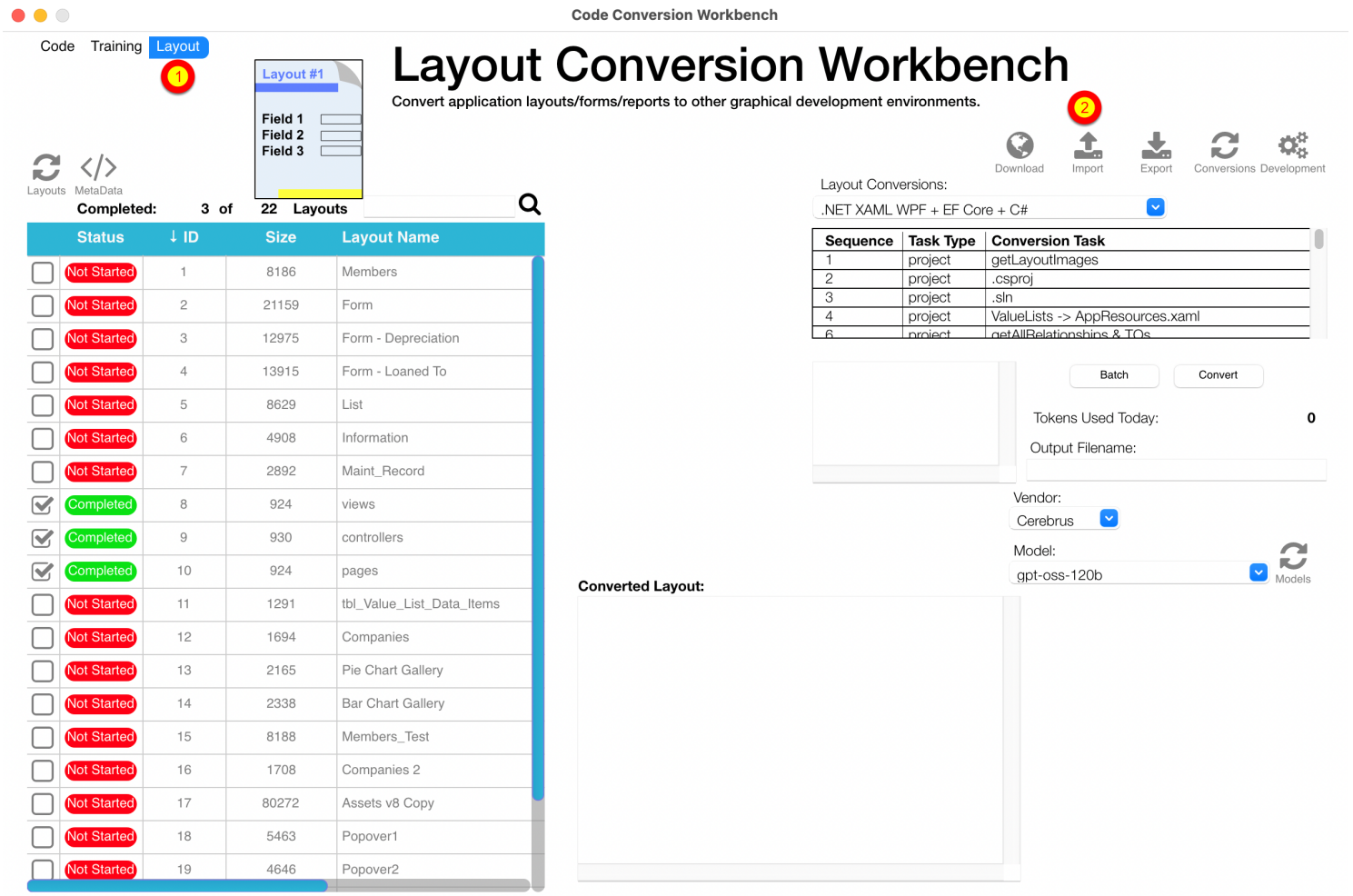
The How to Import Visual FoxPro Projects into FmPro Migrator PDF manual provides importing details.

Click the Layout Conversion Workbench Icon on the GUI Tab of the Migration Process Window



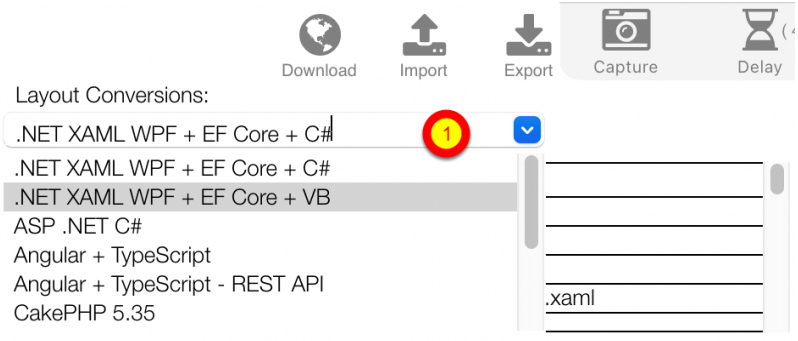
Clicking the Layout Conversion Workbench button opens the Code Conversion Workbench window with the Code tab opened. Click the Layout tab to perform layout conversions.

Importing LayoutConversion.db3 from Layout Tab



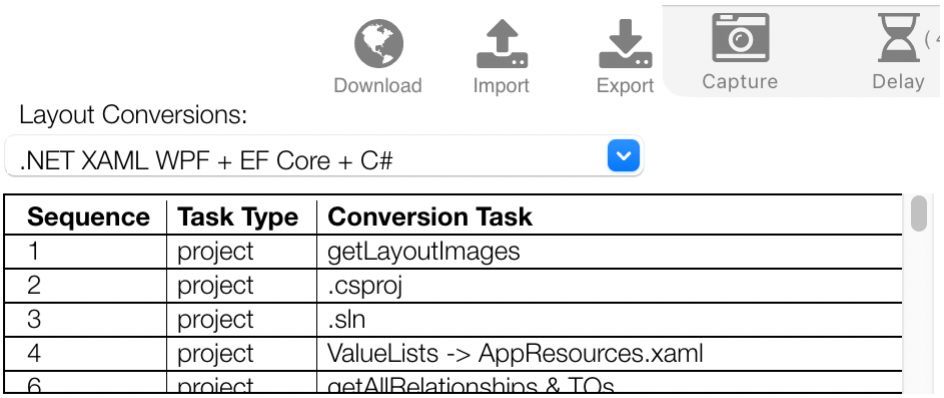
(1) Click the Layouts tab for conversion of layouts. When first opened, there won't be any conversions shown in the conversions menu in the upper right.
(2) Click the Import button to import the LayoutConversion.db3 linked in your License Key email.

Selecting Conversion Menu Options



(1) Select your desired conversion type from the Layout Conversions menu.

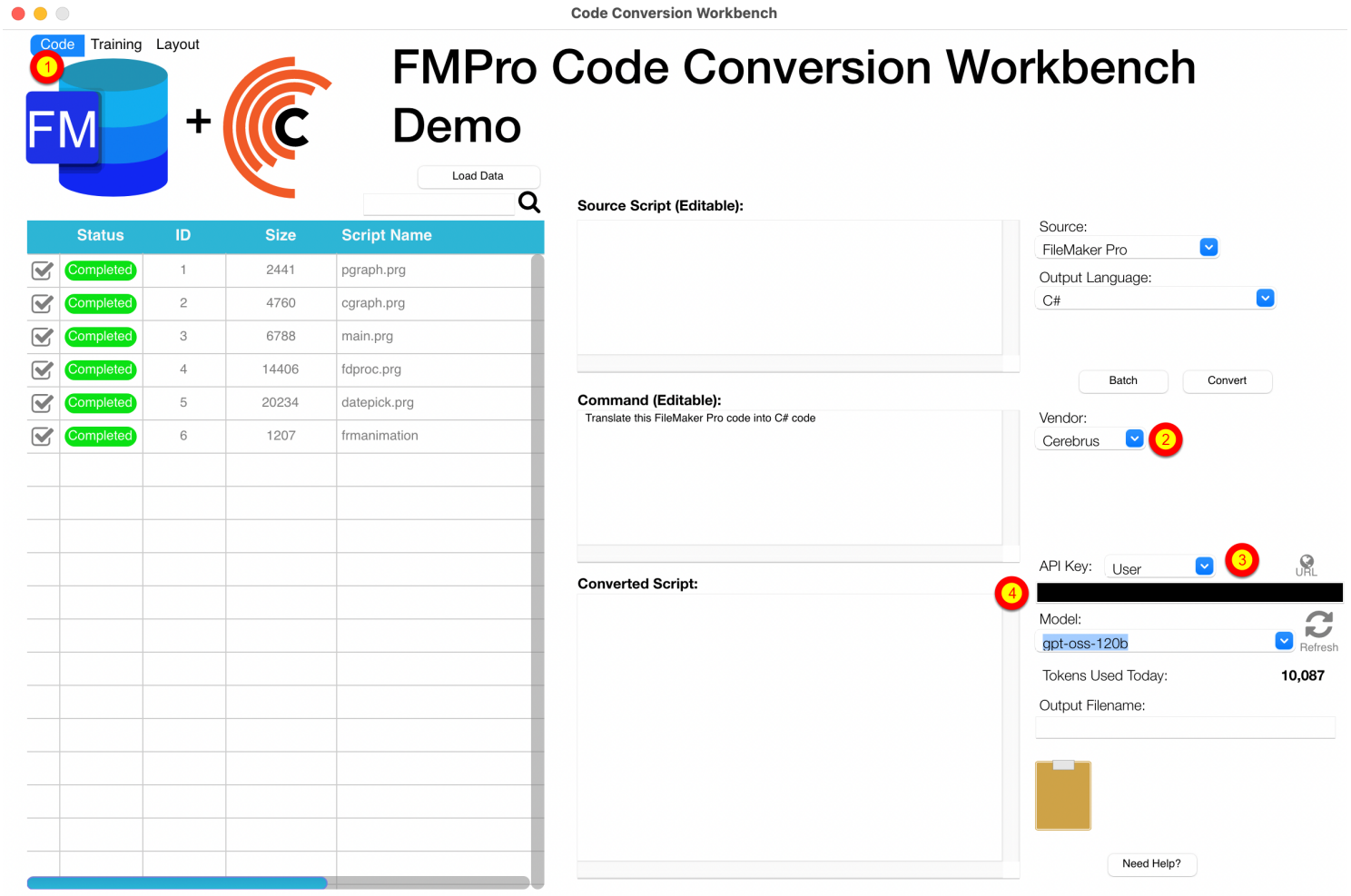
Conversion Tasks List - Layout Tab



Sequence	Task Type	Conversion Task
1	project	getLayoutImages
2	project	.csproj
3	project	.sln
4	project	ValueLists -> AppResources.xaml
6	project	getAllRelationships & TOs

Once a conversion type has been selected, the list of tasks will be displayed in the field below the menu. During conversion processing, the lines of this field will be highlighted with the currently running task. Once all of the project level tasks have been completed, processing will continue for the layout level tasks - if for instance you are doing batch processing of layouts.

Entering Vendor API Keys for Layout Conversions



Layout conversions are a token intensive task, therefore an API key needs entered for each of the LLM vendors you want to use.
Click on the (1) Code tab of the Code Conversion Workbench window to select vendors and enter API keys for any of them you may want to use.
(2) Select the vendor, (3) Select "User" as the API key type (instead of dcsi), (4) enter the API key for the vendor in the API key field [shown blacked out in the screenshot.] Clicking in the API key field will display its contents (if anything has been entered).

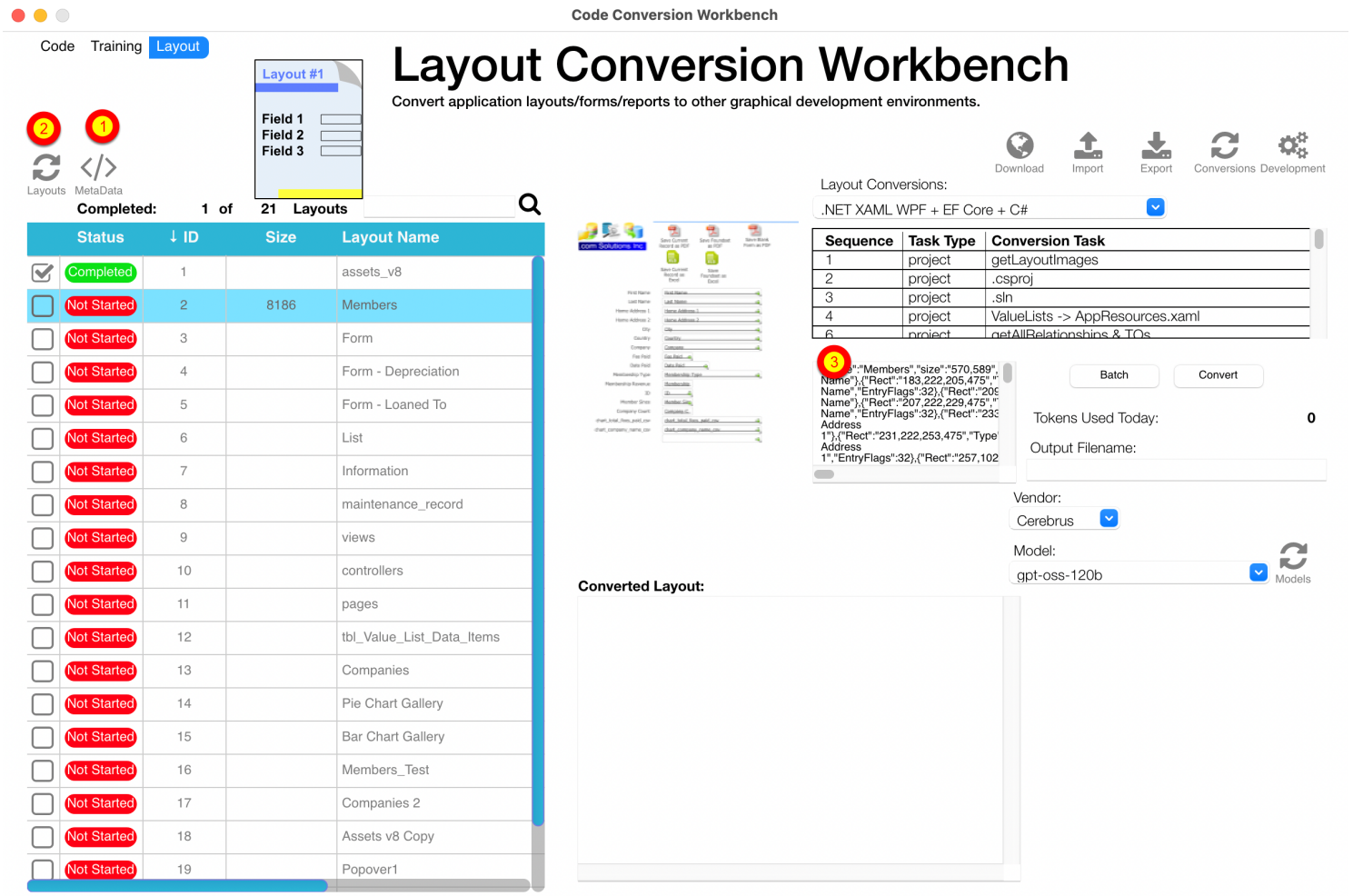
API keys are stored individually for each vendor you may want to use - you can enter API keys for as many vendors as you like, but you need to have at least one.

In this screenshot, you can see that Cerebrus is selected as the vendor, and gpt-oss-120b is selected as the model. This vendor and model combination has been used for a lot of the testing of the Layout Conversion Workbench features due to its high performance of roughly 1000 - 3000 tokens per second.

Note1: Using Ollama to run LLMs locally requires an FmPro Migrator Custom Dev Edition (Trial or Monthly License) or the FmPro Migrator Site License Edition License key.

Note2: Even in Demo mode, layout conversions can be performed on the 1st 5 layouts in the list of layouts - as long as you enter your API key.

Generating MetaData for Layouts



The underlying layout XML for each layout is often going to be too large for processing by LLMs. Therefore the Layout Conversion Workbench includes a metadata summary feature for converting the essential portions of the XML into JSON for token efficient processing by LLMs. This task only needs done once, after importing the layouts.

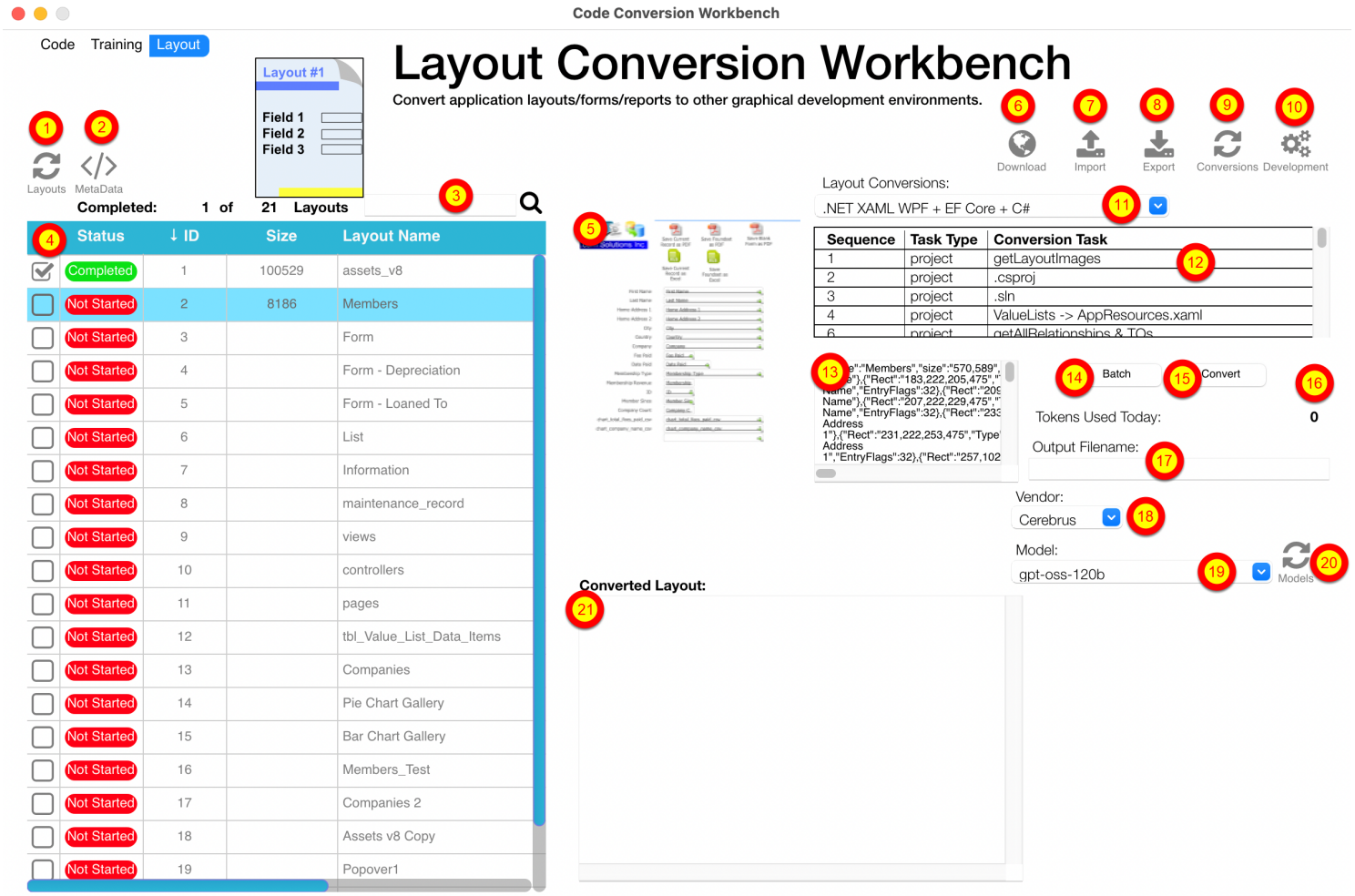
When layouts are initially imported, this processing has not yet been done, as evidenced by the Size column being empty.
Select all of the layouts, (1) click the Metadata button, (2) refresh the list of layouts, (3) then you will be able to observe the JSON metadata in the unlabeled field below the conversion tasks field.

Ready to Convert Layouts

Now that the pre-migration tasks have been completed, layout conversions can be performed.

When using FmPro Migrator (including the Layout Conversion Workbench feature), field contents and menu selections are saved automatically when exiting the field or selecting a menu item.

Overview of Features - Layout Conversion Workbench



The features of the Layout Conversion Workbench tab of the Code Conversion Workbench window are as follows:

(1) Refresh Layouts button - Click this button to refresh the list of layouts, in case layouts have been added or deleted via the Layouts tab of the Migration Process window.

(2) Metadata Calculation button - Selecting one or more layouts and then clicking this button calculates the JSON metadata summary from the XML representing each selected layout. After performing this calculation, the Size column of the layouts grid will be filled with the size of the JSON metadata. In this screenshot the only the 1st two layouts have had metadata calculations performed.

Note: The JSON metadata must be calculated for each layout which is going to be processed.

(3) Search field - Text can be entered in this field for searching the layout names by clicking the magnifying glass icon or just pressing the Tab key. Click the red circled X icon to exit the found set of records display.

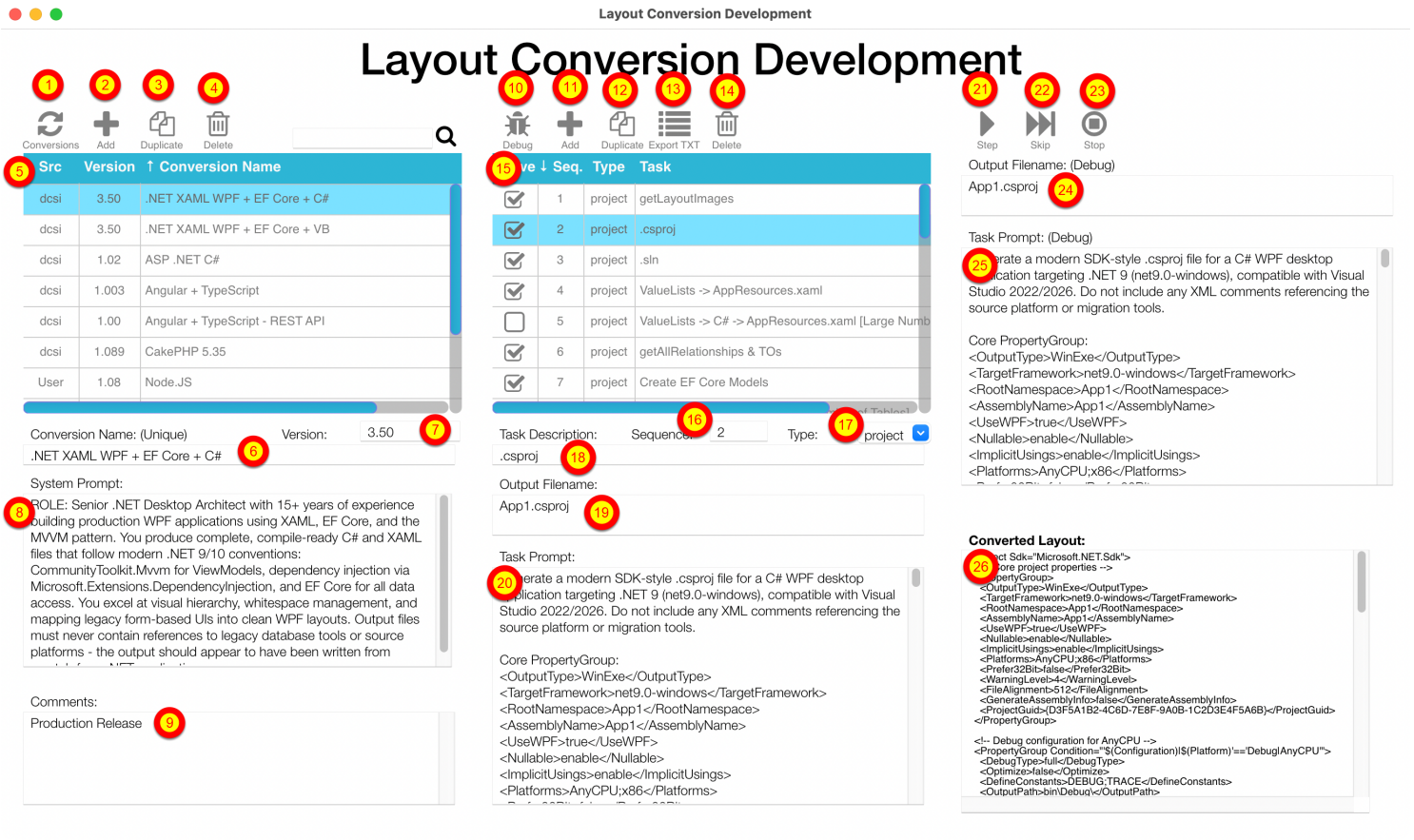
*? - Search - Adding an asterisk at the start of the search text, followed by some text with search for layouts ENDing with the specified text.

?* - Search - Entering some search text followed by an asterisk will find layout names STARTing

with the specified text.
EMPTY Search - Searching with nothing entered, returns all records, in addition to clicking the red circled X icon.
(4) The title row of the layouts grid can be clicked to perform an ascending or descending sort the grid by the clicked column.
(5) FileMaker layouts include a thumbnail imported from the clipboard, displayed in this image field for the currently selected layout. Future versions may include thumbnail images from other development environments like Visual FoxPro.
(6) Download link for the latest version of the LayoutConversion.db3 database of conversions [licensed customers only].
(7) Import the downloaded LayoutConversion.db3 file into the FmPro Migrator preferences location shown in the License window. Each conversion and task record includes a checksum used during importing. Existing records are updated during importing. New records are imported as new records.
(8) Export LayoutConversion.db3 for sharing among a workgroup of developers.
(9) Refresh Conversions - Refreshes the list of conversions shown in the conversions menu.
(10) Layout Conversion Workbench Development button - Opens the development window, enabling modifications to the conversion system prompt and task record contents.
(11) Layout Conversions Selection Menu - Select an available conversion from this menu. Switching conversions removes the ProjectFilesCreated.txt file from the Converted_Layouts folder. This file serves as a flag indicating that project level tasks have been created and don't need run again.
(12) Layout Conversion Tasks - The tasks which will be run for the selected layout conversion are shown in this scrollable field. During processing of the tasks, each row is hilited while the task is running.
(13) JSON MetaData - Once the layout XML metadata has been calculated, it will be displayed in this field when a layout is selected. The contents of this field can be copied and pasted into other apps. This can be helpful in order to see the number of JSON keys which exist in the data and perform other types of analysis.
(14) Batch Button - The Batch button enables the automated conversion of selected layouts, all layouts, not started layouts, completed layouts as shown by the left side status value column.
(15) Convert Button - The Convert button converts only the selected layout. This is a good way to get started, just converting one single layout, before performing Batch processing.
(16) Tokens Used Today - This text label is updated with the number of tokens processed so far today.
(17) Output Filename - Multiple files will generally get written for each layout and these are displayed in this field during task processing.
(18) LLM Vendor Menu - Select the vendor you want to use for processing layouts with this menu. Selecting Ollama for local LLM processing requires a Custom Development (Monthly or Trial) or an FmPro Migrator Site License Edition License.
(19) Model Name Menu - Available models for each vendor are shown in this menu.
(20) Refresh Models Button - LLM vendors frequently release new models after FmPro Migrator has shipped. Clicking this button sends a query to the vendor obtaining a list of the latest models.

The list of downloaded models are stored locally in the FmPro Migrator application preferences file. This is a filtered list of models which are known to be compatible with FmPro Migrator. Holding down the Shift key when clicking this button returns an unfiltered list of models. This button should also be used to refresh the list of models after installing local models with Ollama.
(21) Converted Layout Field - As layouts are being processed, the generated text is displayed in this field. Error messages from the LLM vendor are also displayed in this field.

Overview of Features - Layout Conversion Workbench Development



The Layout Conversion Development window consists of 3 main sections:

[1 - 9 Layout Conversions Grid & Fields]

(1) Refresh Conversions grid button.

(2) Add Layout Conversion button - Adds a new Layout Conversion.

(3) Duplicate Conversion button - Duplicates the selected Layout Conversion and appends the number of seconds to the name to insure that the name is unique.

(4) Deletes the selected Layout Conversion.

(5) The Layout Conversion grid is sortable by clicking the column header.

Note: This grid is also search able just like the Layouts grid.

(6) Conversion Name field - Enter a unique name in this field, since it is used as a unique key in the LayoutConversion.db3 SQLite database. Duplicate names will be flagged as errors during entry.

(7) Version field - The version number is manually entered as an indication of the revision level of the conversion and its included tasks, it is not sent to the LLM during task processing.

(8) System Prompt field - The system prompt provides the proper context to the LLM when processing each individual task.

(9) Comments - The comments field provides a place to make revision notes about the Layout Conversion, and is not sent to the LLM during task processing.

[10 - 20 Conversion Tasks Grid, Fields & Task Type Menu]
(10) Enable/Disable Debug Mode Button - To enable task debugging, make sure that a layout is selected on the Layout tab of the Code Conversion Workbench window. Clicking the button a 2nd time disables debug mode and hides the debugging fields.
(11) Add button - Click to add a new task to the currently selected layout conversion. New tasks appear at the bottom of the tasks grid.
(12) Duplicate Selected Task button - This button duplicates the currently selected task, and it gets listed at the bottom of the grid. For new tasks, the sequence number is entered automatically as the next highest number and tasks are automatically sorted by t he sequence number.
(13) Export Text - This export button exports the prompts for of all of the tasks for currently selected Layout Conversion, including the system prompt, layout conversion name, comments and version field. This feature makes it easier to search & modify the text or just submit the entire file to an LLM to create an entirely new conversion based upon an existing layout conversion process.
(14) Delete Selected Conversion Task button - Deletes the currently selected conversion task.
(15) Tasks grid - The grid of tasks can be sorted via the column headers. Tasks can also be disabled for processing by unchecking the Active checkbox. Some tasks might be used for some projects but not other projects, so this way additional tasks can be created for outlier situations.
(16) Task Sequence field - Tasks are executed in numerical sequence order, as defined by the number placed in this field. Integers are generally used, but decimal numbers can also be used - when inserting tasks between each other.
(17) Task Type Menu - Tasks can run at either the project level or layout level. Project level tasks run just once for an entire project and layout level tasks run for every layout. (i.e. For .NET XAML projects a .XAML file and code behind file are created for each layout - so there are individual tasks which create each of these files for every layout.)
(18) Task Description field - This is a human readable field generally with an output field type shown for instance.
(19) Output Fieldname - The output fields are named either statically or dynamically by the contents of this field. Unless specified otherwise, all files are written into the Converted_Layouts folder within the FmPro Migrator project directory containing the MigrationProcess.db3 file.
Astatic field name might be something like: App1.csproj as shown here for the project filename.
Adynamic field name example would be like the following: [["/Properties/" & AssemblyInfo.cs]]
For this example, a Properties folder has been created during a previous task, and the AssemblyInfo.cs file is being written into the Properties folder. The enclosing [[]] square brackets indicate that the contents will be evaluated at run time. The result of this rendering is displayed in the debugging version of this field.
Note: Tasks without a fieldname - are not sent to the LLM for processing. This provides the flexibility of just running internal code blocks to perform special tasks like exporting the binary images from all of the layouts.
(20) Task Prompt field - This is the main prompt field for the task to be processed by the LLM. Task prompts also usually include <Code></Code> blocks which are executed at run time. The code blocks are written in the LiveCode scripting language used to develop FmPro Migrator.
Here is an example code block:
<Code>

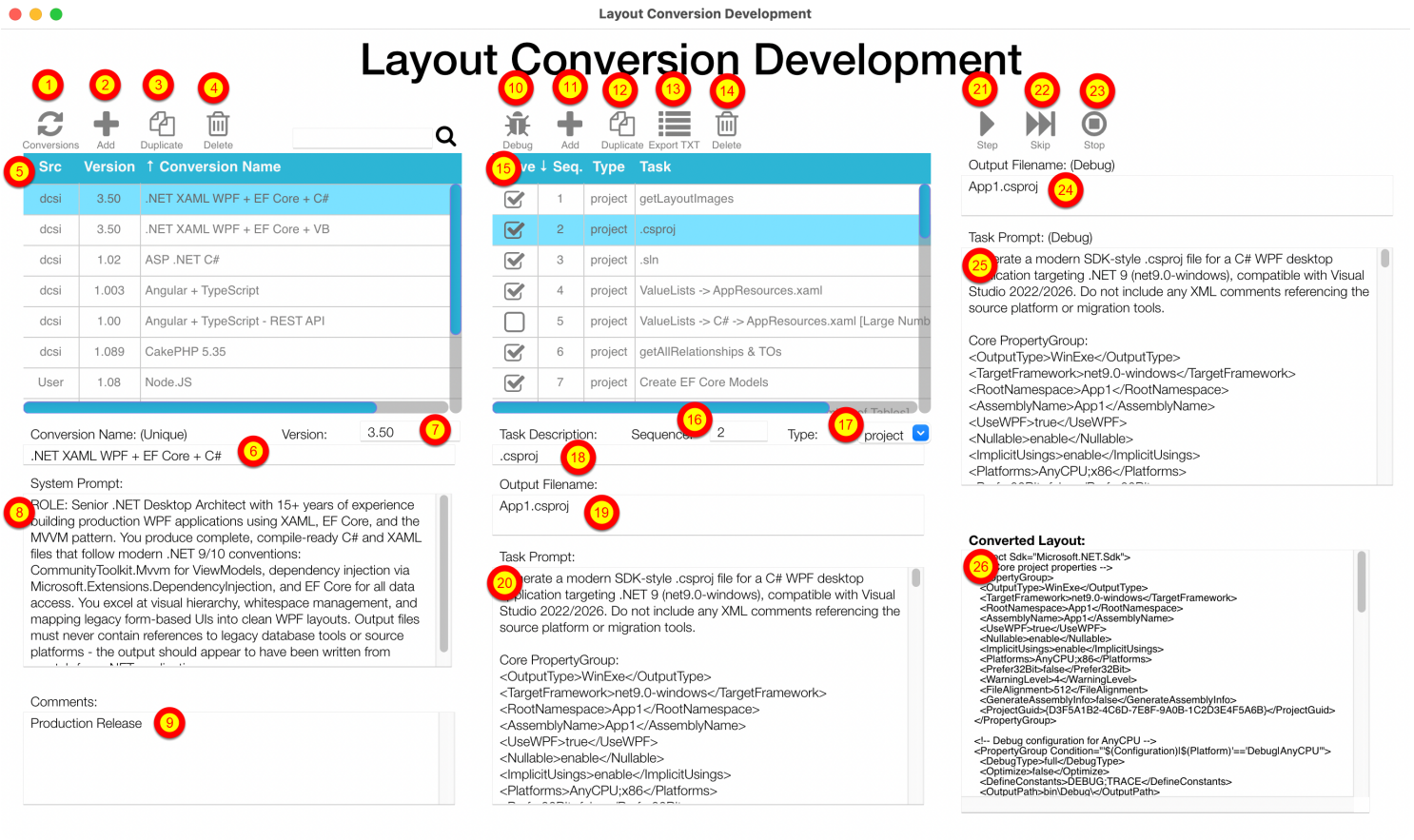
-- override sending layout metadata
put "N/A" into gLayoutConversion["PromptData"]
createFolder gLayoutConversion["OutputFolderPath"] & "/Properties"
</Code>
By default, the JSON metadata would be sent to the LLM in the ["PromptData"] key of the gLayoutConversion global variable. Since that JSON metadata isn't required for this task, the value "N/A" is getting sent in its place.
The createFolder handler then runs and creates a folder in the Converted_Layouts folder from the path which is passed into the handler.

[21 - 26 Conversion Task Debug Fields]

(21) Debug Step button - In debug mode, click this button to run the currently selected task. After running, the debug fields on the right side will be rendered with dynamically executed code and the results from the LLM will be displayed in the Converted Layout field.
(22) Debug Skip button - In debug mode, clicking this button skips to the next task and runs it.
(23) Debug Stop button - Exits debug mode.
(24) Debug Output filename field - This is the calculated version of the output filename.
(25) Debug Task Prompt field - This is the calculated version of the task prompt, which includes the results of the <Code> block.
(26) Converted Layout field - This field contains the results of the prompts returned by the LLM, with the exception of some error message text. Error message text which doesn't result in any output by the LLM gets written into the Converted Layout field on the Layout tab of the Code Conversion Workbench window.

Using the Layout Conversion Development & Debug Window

Clicking the Development button on the Layout Conversion Workbench window opens the Layout Conversion Workbench Development window shown below.
This window enables the development and debugging of Layout Conversions and Layout Conversion Tasks. The debug feature enables running individual tasks one at a time - showing the results of dynamically generated text like filename and <Code> blocks.



The Layout Conversion Development window consists of 3 main sections:

[1 - 9 Layout Conversions Grid & Fields]

(1) Refresh Conversions grid button.

(2) Add Layout Conversion button - Adds a new Layout Conversion.

(3) Duplicate Conversion button - Duplicates the selected Layout Conversion and appends the number of seconds to the name to insure that the name is unique.

(4) Deletes the selected Layout Conversion.

(5) The Layout Conversion grid is sortable by clicking the column header.

Note: This grid is also search able just like the Layouts grid.

(6) Conversion Name field - Enter a unique name in this field, since it is used as a unique key in the LayoutConversion.db3 SQLite database. Duplicate names will be flagged as errors during entry.

(7) Version field - The version number is manually entered as an indication of the revision level of the conversion and its included tasks, it is not sent to the LLM during task processing.

(8) System Prompt field - The system prompt provides the proper context to the LLM when processing each individual task.

(9) Comments - The comments field provides a place to make revision notes about the Layout Conversion, and is not sent to the LLM during task processing.

[10 - 20 Conversion Tasks Grid, Fields & Task Type Menu]

(10) Enable/Disable Debug Mode Button - To enable task debugging, make sure that a layout is selected on the Layout tab of the Code Conversion Workbench window. Clicking the button a 2nd time disables debug mode and hides the debugging fields.

(11) Add button - Click to add a new task to the currently selected layout conversion. New tasks appear at the bottom of the tasks grid.

(12) Duplicate Selected Task button - This button duplicates the currently selected task, and it gets listed at the bottom of the grid. For new tasks, the sequence number is entered automatically as the next highest number and tasks are automatically sorted by t he sequence number.

(13) Export Text - This export button exports the prompts for of all of the tasks for currently selected Layout Conversion, including the system prompt, layout conversion name, comments and version field. This feature makes it easier to search & modify the text or just submit the entire file to an LLM to create an entirely new conversion based upon an existing layout conversion process.

(14) Delete Selected Conversion Task button - Deletes the currently selected conversion task.

(15) Tasks grid - The grid of tasks can be sorted via the column headers. Tasks can also be disabled for processing by unchecking the Active checkbox. Some tasks might be used for some projects but not other projects, so this way additional tasks can be created for outlier situations.

(16) Task Sequence field - Tasks are executed in numerical sequence order, as defined by the number placed in this field. Integers are generally used, but decimal numbers can also be used - when inserting tasks between each other.

(17) Task Type Menu - Tasks can run at either the project level or layout level. Project level tasks run just once for an entire project and layout level tasks run for every layout. (i.e. For .NET XAML projects a .XAML file and code behind file are created for each layout - so there are individual tasks which create each of these files for every layout.)

(18) Task Description field - This is a human readable field generally with an output field type shown for instance.

(19) Output Fieldname - The output fields are named either statically or dynamically by the contents of this field. Unless specified otherwise, all files are written into the Converted_Layouts folder within the FmPro Migrator project directory containing the MigrationProcess.db3 file.
Astatic field name might be something like: App1.csproj as shown here for the project filename.
Adynamic field name example would be like the following: [["/Properties/" & AssemblyInfo.cs]]
For this example, a Properties folder has been created during a previous task, and the AssemblyInfo.cs file is being written into the Properties folder. The enclosing [[]] square brackets indicate that the contents will be evaluated at run time. The result of this rendering is displayed in the debugging version of this field.

Note: Tasks without a fieldname - are not sent to the LLM for processing. This provides the flexibility of just running internal code blocks to perform special tasks like exporting the binary images from all of the layouts.

(20) Task Prompt field - This is the main prompt field for the task to be processed by the LLM. Task prompts also usually include <Code></Code> blocks which are executed at run time. The code blocks are written in the LiveCode scripting language used to develop FmPro Migrator.
Here is an example code block:

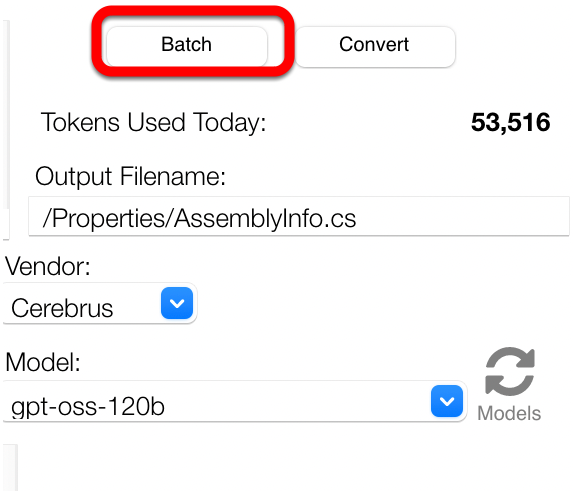
<Code>

-- override sending layout metadata
put "N/A" into gLayoutConversion["PromptData"]
createFolder gLayoutConversion["OutputFolderPath"] & "/Properties"
</Code>
By default, the JSON metadata would be sent to the LLM in the ["PromptData"] key of the gLayoutConversion global variable. Since that JSON metadata isn't required for this task, the value "N/A" is getting sent in its place.
The createFolder handler then runs and creates a folder in the Converted_Layouts folder from the path which is passed into the handler.

[21 - 26 Conversion Task Debug Fields]

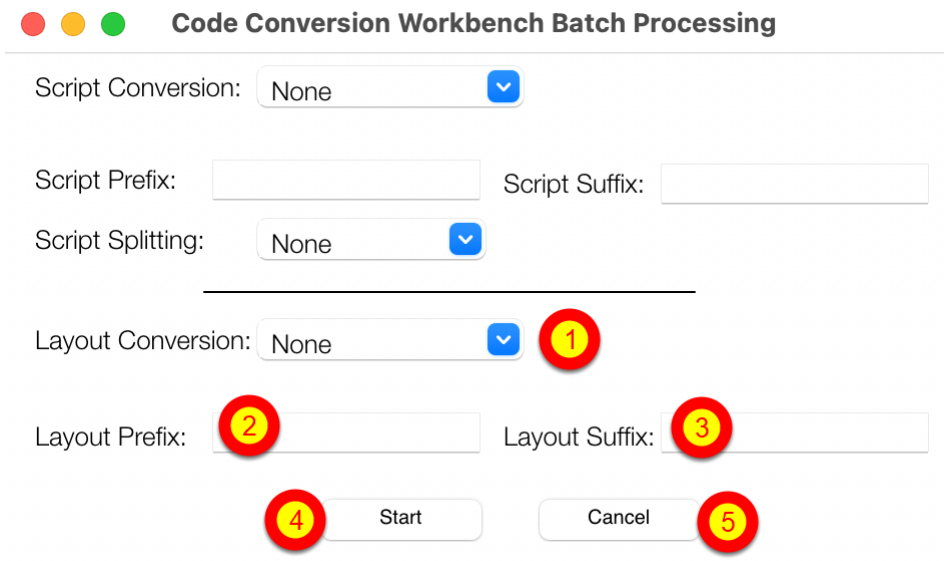
(21) Debug Step button - In debug mode, click this button to run the currently selected task. After running, the debug fields on the right side will be rendered with dynamically executed code and the results from the LLM will be displayed in the Converted Layout field.
(22) Debug Skip button - In debug mode, clicking this button skips to the next task and runs it.
(23) Debug Stop button - Exits debug mode.
(24) Debug Output filename field - This is the calculated version of the output filename.
(25) Debug Task Prompt field - This is the calculated version of the task prompt, which includes the results of the <Code> block.
(26) Converted Layout field - This field contains the results of the prompts returned by the LLM, with the exception of some error message text. Error message text which doesn't result in any output by the LLM gets written into the Converted Layout field on the Layout tab of the Code Conversion Workbench window.

Batch Processing Button - Layout Tab



Click the Batch button next to the Convert button on the Layout tab of the Code Conversion Workbench window to open the Code Conversion Workbench Batch Processing window.
Note: The Batch button on the Code tab works identically, and opens the same Batch Processing window.

Code Conversion Workbench Batch Processing Window

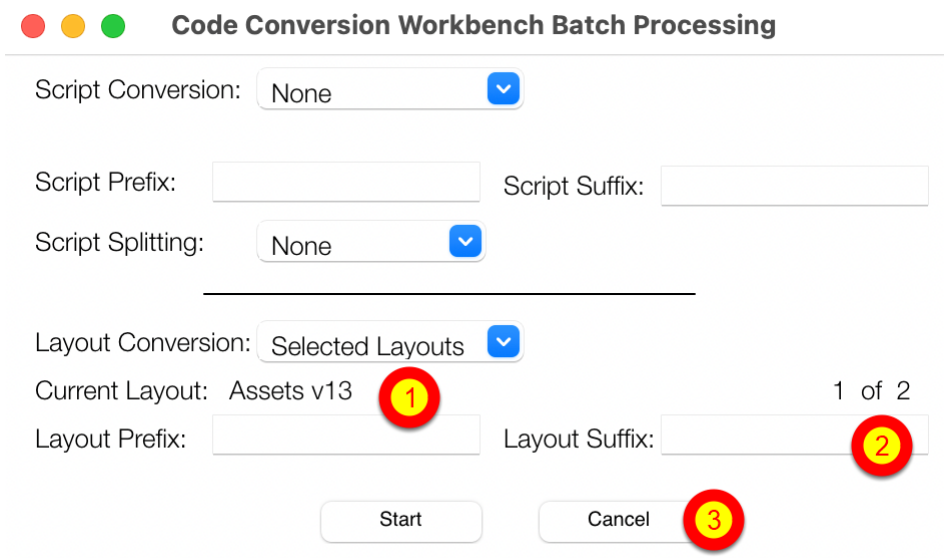


The Batch Processing window controls batch processing for both script and layout conversions. Batch processing can be done on only scripts, only layouts or both scripts and layouts.This manual will focus on the Layout Conversion processing.

Layout Conversion batch processing options include:
(1) Selection of layouts include:
All - All layouts will be processed.
Selected Layouts - Only selected layouts will be processed.
None - No layouts will be processed.
Completed - Only layouts with status = Completed will be processed.
Not Started - Only layouts with status = Not Started will be processed.

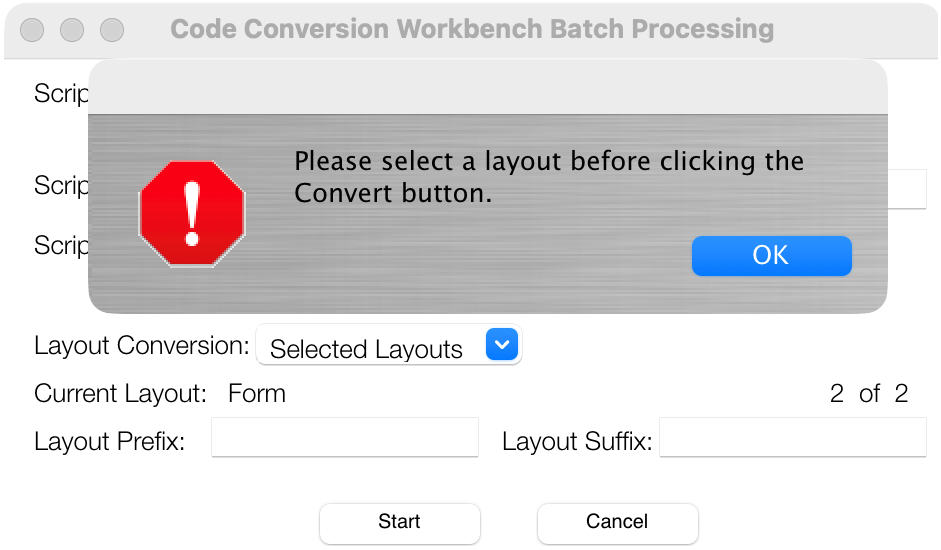
(2) Layout Prefix field - Text entered in this field will be added as a prefix to the name of the converted layout file.
(3) Layout Suffix field - Text entered in this field will be added as a suffix to the name of the converted layout file.
(4) Start button - Starts the batch processing.
(5) Cancel button - Cancel batch processing and close the dialog.

Batch Processing - Status Info



During batch processing, the (1) current layout name will be displayed, and the (2) countdown of the number of remaining layouts will be displayed.
(3) The Cancel button can be used to cancel processing during batch processing.

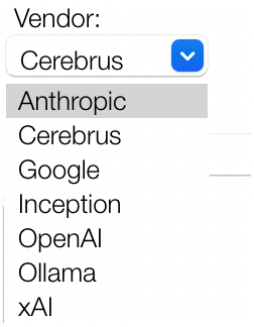
Troubleshooting Batch Processing - Please Select Layout Error



This error will occur and stop the batch processing if any of the layouts don't contain JSON metadata.

Available Vendors & Models

List of Vendors



The current list of available vendors includes:

Anthropic - The Anthropic Opus model has been primarily used for the development of the prompts used for layout conversions. API pricing is generally higher than other vendors.

Cerebrus - Cerebrus offers the fastest processing of any of the vendors with speeds reaching 1000 - 3000 tokens per second. Cerebrus has been used extensively during development in order to primarily use the gpt-oss-120b model for fast iteration of the Layout Conversion Workbench. Getting started with Cerebrus costs only $10 for API access.

Google - The Google Gemini Pro models have been used economically in situations where many hundreds of objects exist on a layout. Large layouts may take several minutes to process, but do a better job than the flash models for this task.

Inception - Inception is the 2nd fastest LLM tested for layout conversions, and presently offers 10M free tokens for testing purposes. The Mercury2 model is one of the first text diffusion models, offering faster performance and fewer hallucinations compared with traditional models.

OpenAI - OpenAI Pro models are an additional option for large layout processing.

Ollama - Ollama models can be used locally when privacy and security needs are of paramount importance. The gpt-oss-120b model is a good choice for converting most layouts, except for the very largest layouts.

xAI - xAI is another option with models having a 2M context size for processing large layouts.

Summary of Code Handlers

Task prompts also usually include <Code></Code> blocks which are executed at run time. The code blocks are written in the LiveCode scripting language used to develop FmPro Migrator.
Here is an example code block:

```
<Code>
-- override sending layout metadata
put "N/A" into gLayoutConversion["PromptData"]
createFolder gLayoutConversion["OutputFolderPath"] & "/Properties"
</Code>
```

By default, the JSON metadata would be sent to the LLM in the ["PromptData"] key of the gLayoutConversion global variable. Since that JSON metadata isn't required for this task, the value "N/A" is getting sent in its place.
The createFolder handler then runs and creates a folder in the Converted_Layouts folder from the path which is passed into the handler.

A<Code> block needs to start with the <Code> keyword at the beginning of the line and it ends with the closing </Code> tag at the beginning of the last line of the code block.
Only one Code block is implemented.

Prompt text may occur before or after the <Code> block.

The following is a summary of LiveCode handlers which can be useful to run within <Code> blocks.

createFolder
Type: Handler
Parameters pFolderPath — full path to the folder to create
Returns: None (appends result message to gLayoutConversion["PromptCodeResult"])
Description: Uses a three-way switch: if the path is empty, reports an error; if the folder already exists, reports that it was skipped; otherwise calls LiveCode's create folder command in a try/catch and reports success or the caught error.

getDatabasePropertiesText
Type: Handler
Parameters: None
Returns: None (appends database info to gLayoutConversion["PromptCodeResult"])
Description: Queries Migration_Tables for source table names and destination DB type, builds a block listing the database type, name/filename, hostname, and port, followed by a formatted list of converted table names, and appends the result to PromptCodeResult.

getAllValueLists
Type: Handler

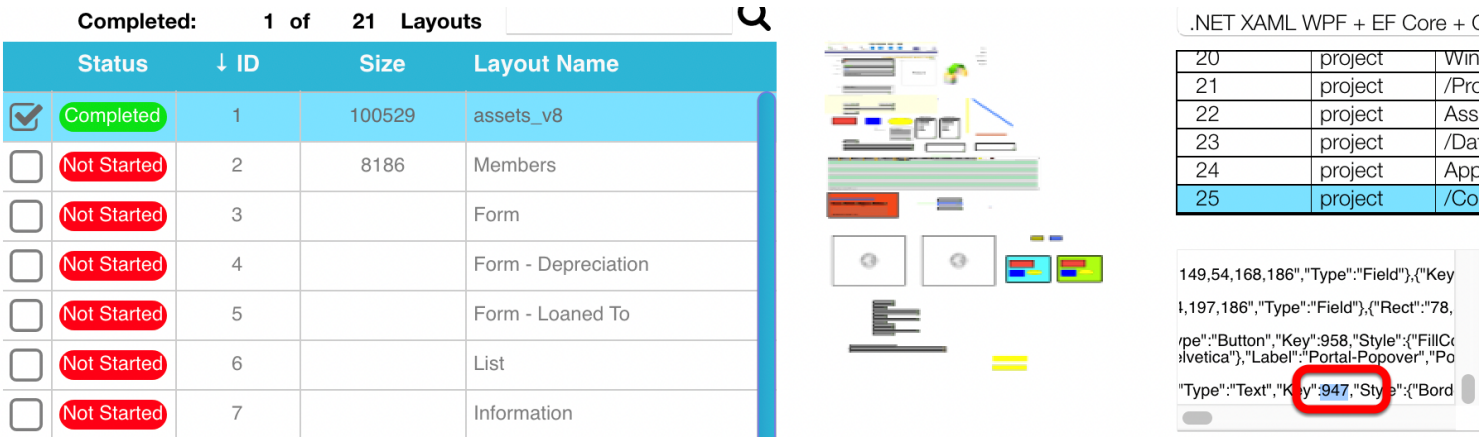
Parameters: pOutputType — "File" | anything else (Variable); pOutputDataLocation — file path when File
Returns: None (populates gLayoutConversion["PromptData"] or writes UTF-8 file)
Description: Reads all Value_Lists records from the MigrationProcess DB. Custom and External/Custom value lists are assembled into full FM-compatible XML; other types use their stored XML directly. Wraps all entries in an fmxmlsnippet envelope. Outputs to file or to gLayoutConversion["PromptData"].

listAllLayoutImages
Type Handler
Parameters: pLayoutImagesFolder — folder path to scan
Returns: None (appends sorted image filenames to gLayoutConversion["PromptCodeResult"] and copies the list into gLayoutConversion["PromptData"])
Description: Lists all non-hidden, non-.txt files in pLayoutImagesFolder, extracts numeric suffixes from filenames (e.g. Image3 → 3), sorts the list numerically, and outputs the sorted filenames.

getLayoutImagesList
Type: Handler
Parameters: pLayoutImageListFilePath — full path to an images manifest .txt file
Returns: None (appends file contents to gLayoutConversion["PromptCodeResult"])
Description: Checks for the manifest file's existence, opens it as UTF-8, reads its contents, and appends them (plus a return) to the PromptCodeResult global key.

Troubleshooting

Incomplete Layout Conversion

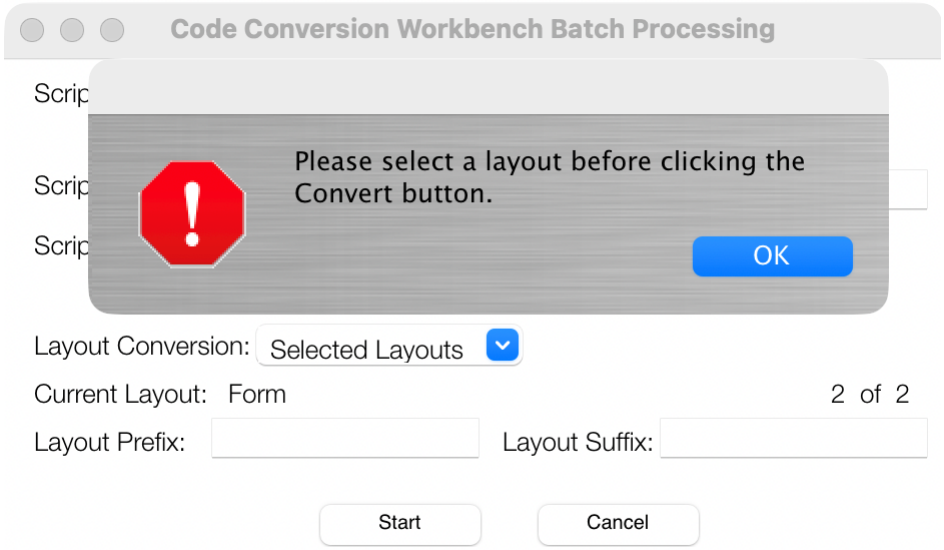


Looking at the JSON metadata for this layout, you can see that there are over 900 keys in the JSON array. Some of the LLMs have trouble processing this many objects accurately (i.e. gpt-oss-120b for instance). When this occurs the layout does get converted into .NET XAML for instance, but there may be a significant number of missing objects rendered into the XAML file.

Therefore for large layouts like this one, there are a couple of options available:
1) For projects where cloud-based models can be used, the Google Gemini Pro 3.x models generally seem to process all of the 900+ objects on the layout.
2) For projects requiring local processing via Ollama, there are a number of larger context sized models which could be tested on the 512GB Unified memory server provided with FmPro Migrator Site License Edition:

mistral-medium-3.5 128b - 256k Context
deepseek-v4-flash 158b - 1M Context
minimax-m2.7 229b - 200K Context
qwen3.6 35b-mlx - 256K Context
nemotron-3-super 120b - 256K Context
gemma4:31b-mlx 20b - 256K Context

Troubleshooting Batch Processing - Please Select Layout Error



This error will occur and stop the batch processing if any of the layouts don't contain JSON metadata.