.com Solutions Inc.

Code Conversion Workbench Manual

Code Conversion Workbench Manual

Overview

Overview

The Code Conversion Workbench feature within the AI Accelerated version of FmPro Migrator Platinum Edition is used to convert code from FileMaker Pro, FoxPro 2.6, Microsoft Access, Visual FoxPro, COBOL and other programming languages. The Code Conversion Workbench converts code into over 50 different programming languages by using multiple machine learning models. As the code is converted, the files are written into the output directory, and can also be copied via the clipboard and pasted directly into FileMaker Pro or other development environments.

Revision 6
FmPro Migrator 11.73
1/26/2026

Videos showing the conversion process are available on You Tube at: @FmProMigrator

[Revision Notes: Added info about the new script search and sorting features and replacement of the ScriptStatus.JSON file with columns added to the Scripts table of the MigrationProcess.db3 SQLite database.]

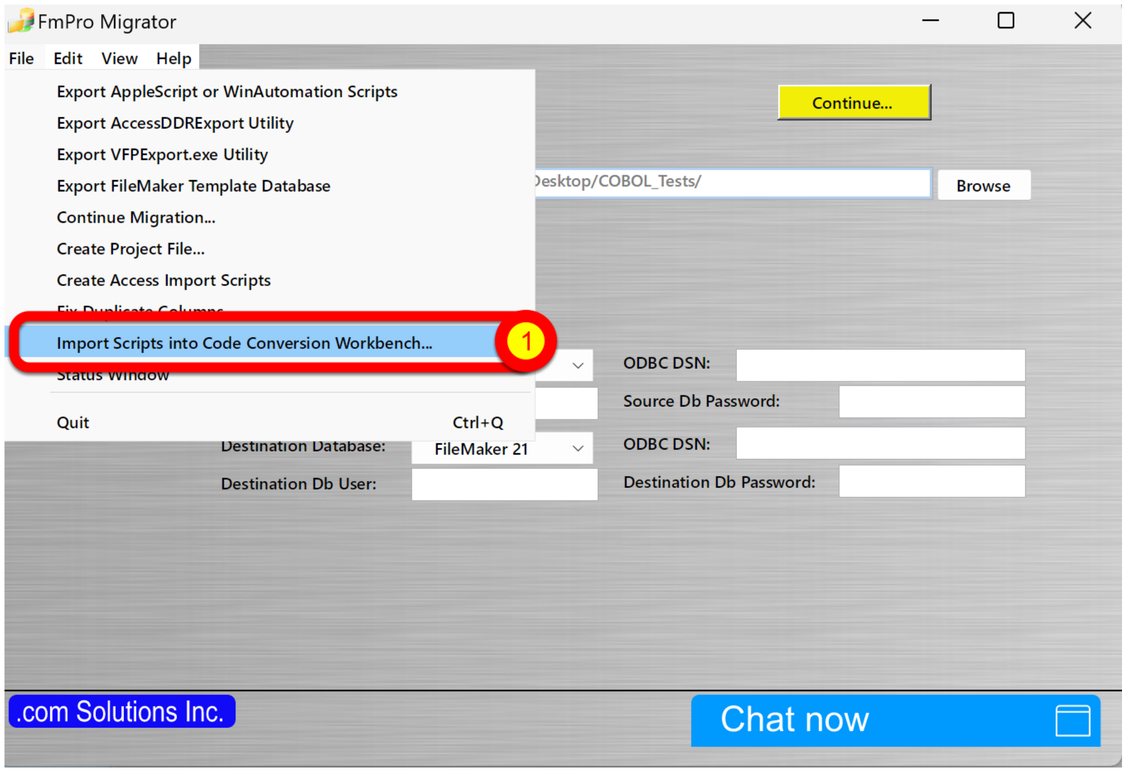
GUI

Importing Scripts with the Quick Start Feature

The Quick Start feature makes it easy to import scripts as text files from any programming language directly into the Code Conversion Workbench. This feature is especially helpful for quickly building Proof of Concept projects during testing.

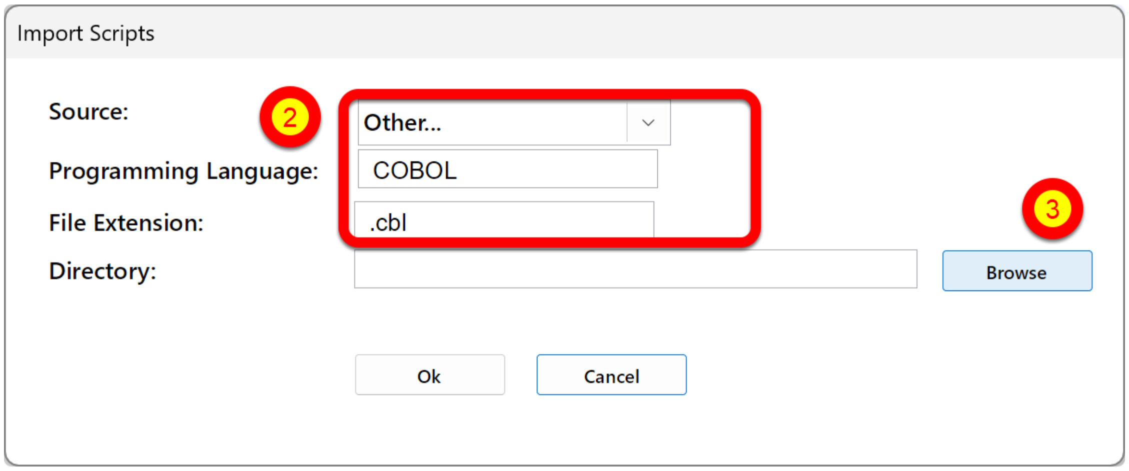
Note: However this feature is not intended to import database tables, layouts/forms/reports or relationships or Script Workspace scripts from FileMaker Pro. If you end up following the full import process when converting a Visual FoxPro project after using the Quick Start feature, the scripts of the project will get imported twice.

Import Scripts Menu



1) Select the File -> Import Scripts into Code Conversion Workbench... menu in the FmPro Migrator main application window.

Select Programming Language in Import Scripts Dialog



Source: Other...
Programming Language: COBOL
File Extension: .cbl
Directory:

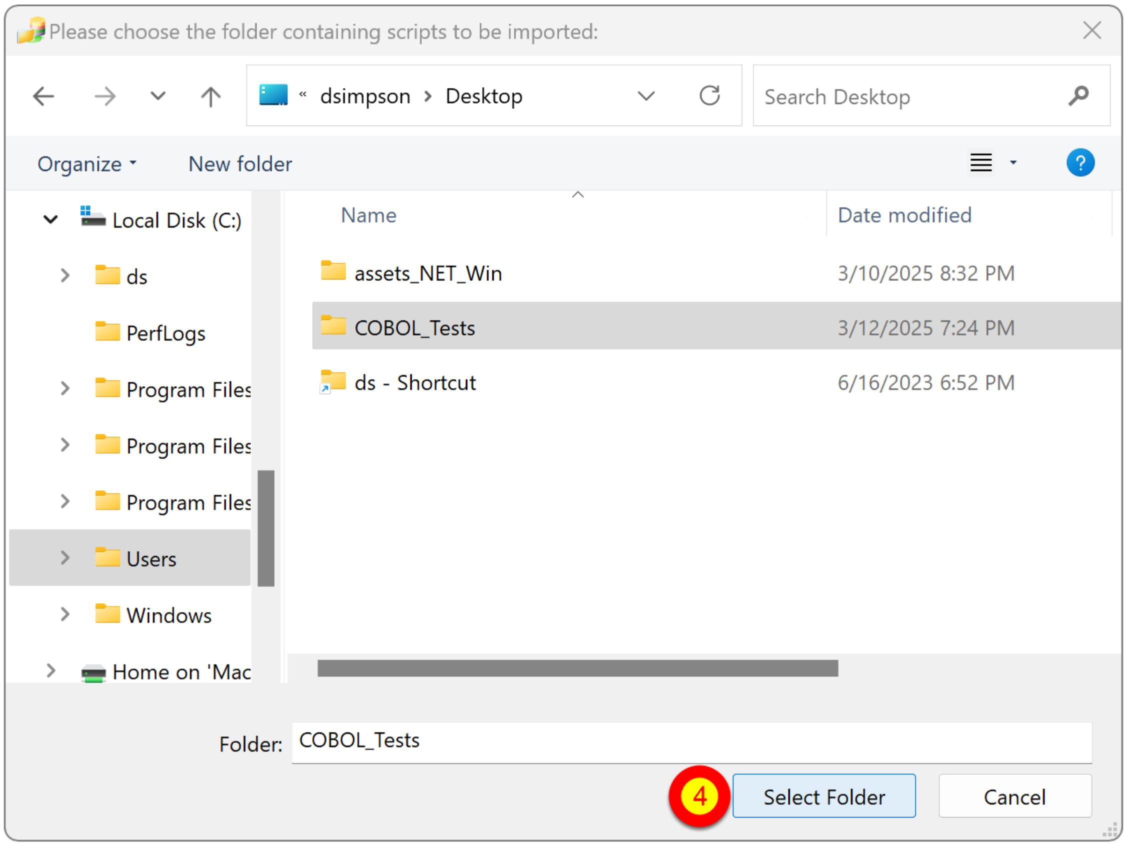
The Programming Language menu includes commonly used items including:
FoxPro 2.6
Visual FoxPro
Other...

2) When selecting FoxPro 2.6 or Visual FoxPro, the Programming Language and file Extension fields will be filled in automatically.

When selecting Other... for the Programming Language, manually enter the name and file extension for the scripts to be imported. The example shown here is showing the import of COBOL/.cbl for the Programming Language and file extension. This is just an example, since any text file format can be imported for automated conversion. The Programming Language name will be used directly in the prompt sent to the LLMs for processing.

3) Click the Browse button to select the folder of scripts which will be imported.

Select Import Folder

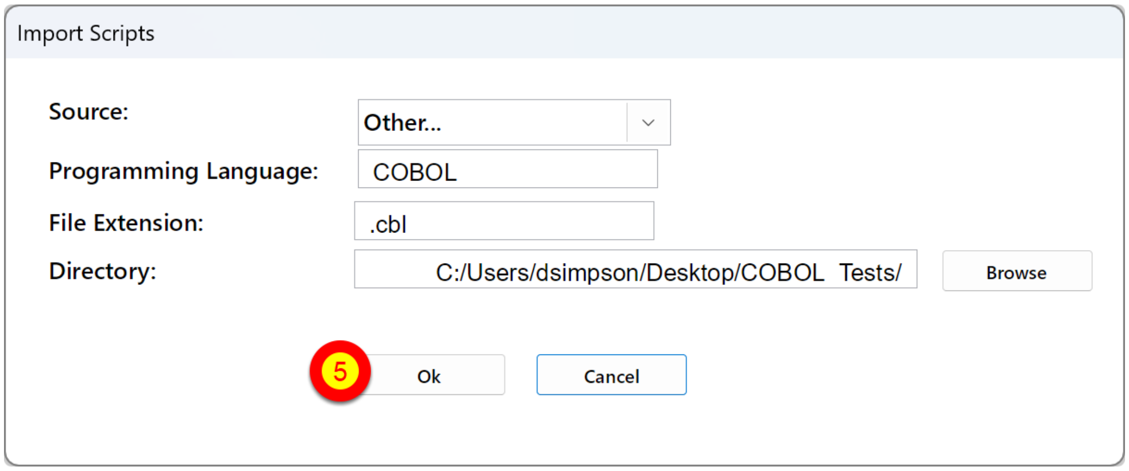


Name	Date modified
assets_NET_Win	3/10/2025 8:32 PM
COBOL_Tests	3/12/2025 7:24 PM
ds - Shortcut	6/16/2023 6:52 PM

Folder: COBOL_Tests

Select the folder of scripts you want imported into the Code Conversion Workbench.

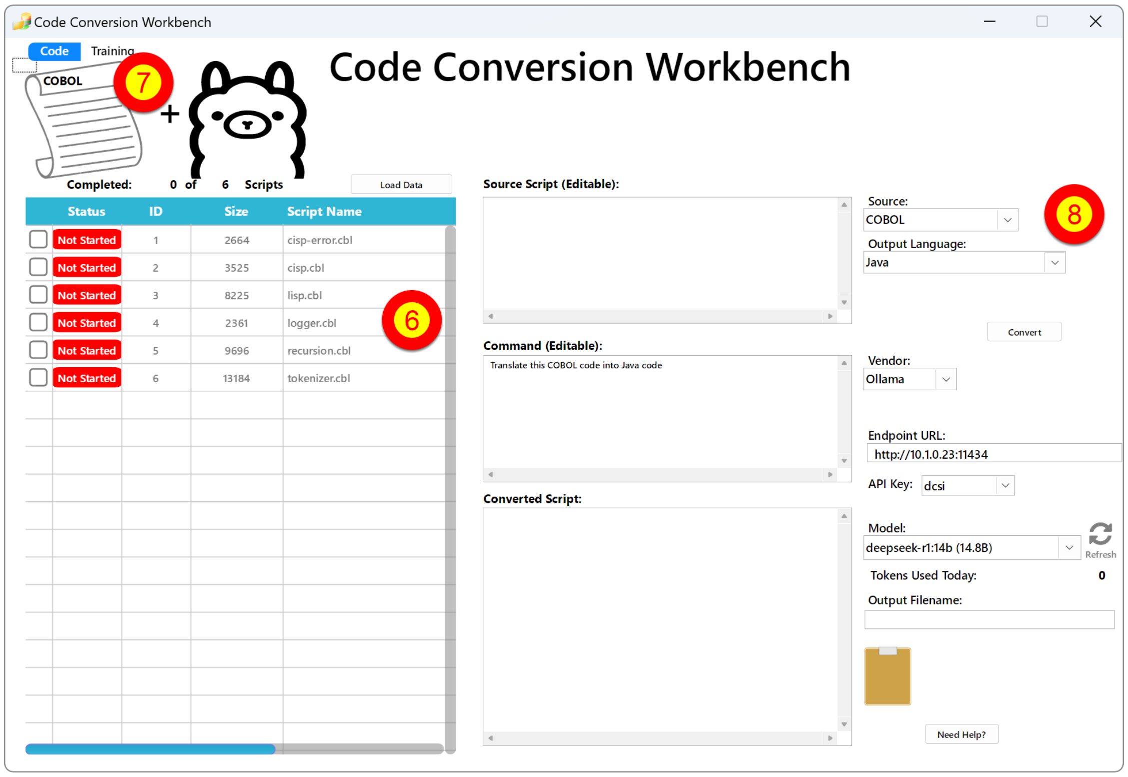
Click OK Button



Source: Other...
Programming Language: COBOL
File Extension: .cbl
Directory: C:/Users/dsimpson/Desktop/COBOL_Tests/

5) After selecting the scripts folder, click the (5) Ok button to perform the import. Scripts will be searched recursively starting with the selected folder.

Imported Scripts in Code Conversion Workbench



Code Conversion Workbench

Completed: 0 of 6 Scripts

Status	ID	Size	Script Name
Not Started	1	2664	cisp-error.cbl
Not Started	2	3525	cisp.cbl
Not Started	3	8225	lisp.cbl
Not Started	4	2361	logger.cbl
Not Started	5	9696	recursion.cbl
Not Started	6	13184	tokenizer.cbl

Source Script (Editable):

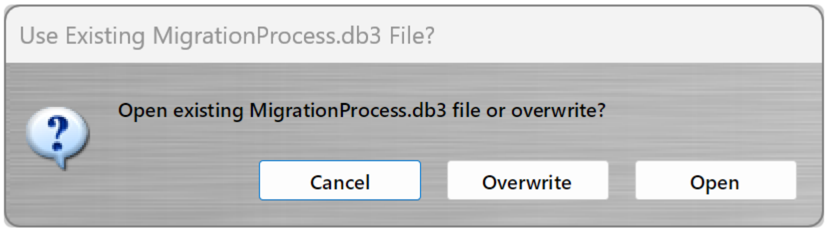
Command (Editable):
Translate this COBOL code into Java code

Converted Script:

Source: COBOL
Output Language: Java
Vendor: Ollama
Endpoint URL: http://10.1.0.23:11434
API Key: dcsi
Model: deepseek-r1:14b (14.8B)
Tokens Used Today: 0
Output Filename:

6) Imported scripts are displayed in the grid at the left side of the window. (7) The name of the programming language is displayed in the generic script icon above the grid and also (8) in the Source menu at the right side of the window.

Overwrite Project Dialog



Use Existing MigrationProcess.db3 File?

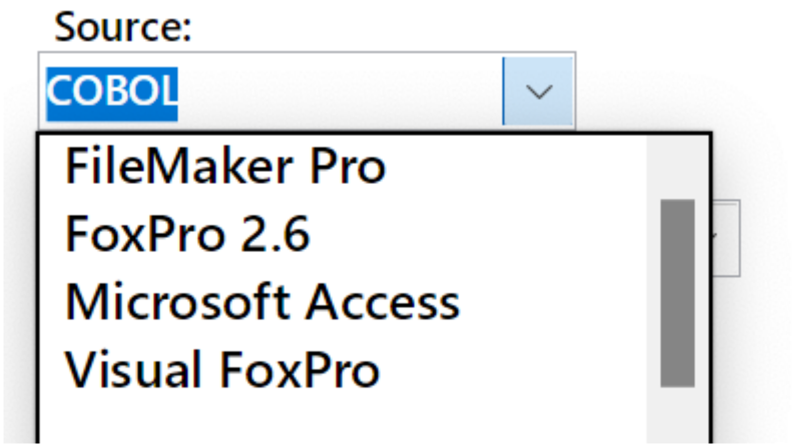
Open existing MigrationProcess.db3 file or overwrite?

FmPro Migrator uses a SQLite project file named MigrationProcess.db3 for storing conversion project metadata. When doing a Quick Start import, the MigrationProcess.db3 file will be written into the top level folder selected in the import dialog.

If you click:
Cancel - The MigrationProcess.db3 file won't be overwritten and no import of scripts will be performed.
Overwrite - The existing MigrationProcess.db3 file will be renamed using the current date/time, a new MigrationProcess.db3 file will be created and the scripts will be imported. This is a good

option if you haven't started the conversion project yet and you added scripts to the import folder since the previous import was performed.
Open - The existing MigrationProcess.db3 file will be opened and used by the Code Conversion Workbench and scripts won't be imported.

Editing the Programming Language



The Source language field is directly editable. Editing the contents of this field can be helpful if you accidentally selected one of the other standard options and you need to revert that change. The contents of this field will be automatically saved to preferences, ready to be restored the next time the Code Conversion Workbench is opened.

Using the Code Conversion Workbench

The Code Conversion Workbench is included with the AI Accelerated version of FmPro Migrator Platinum Edition.

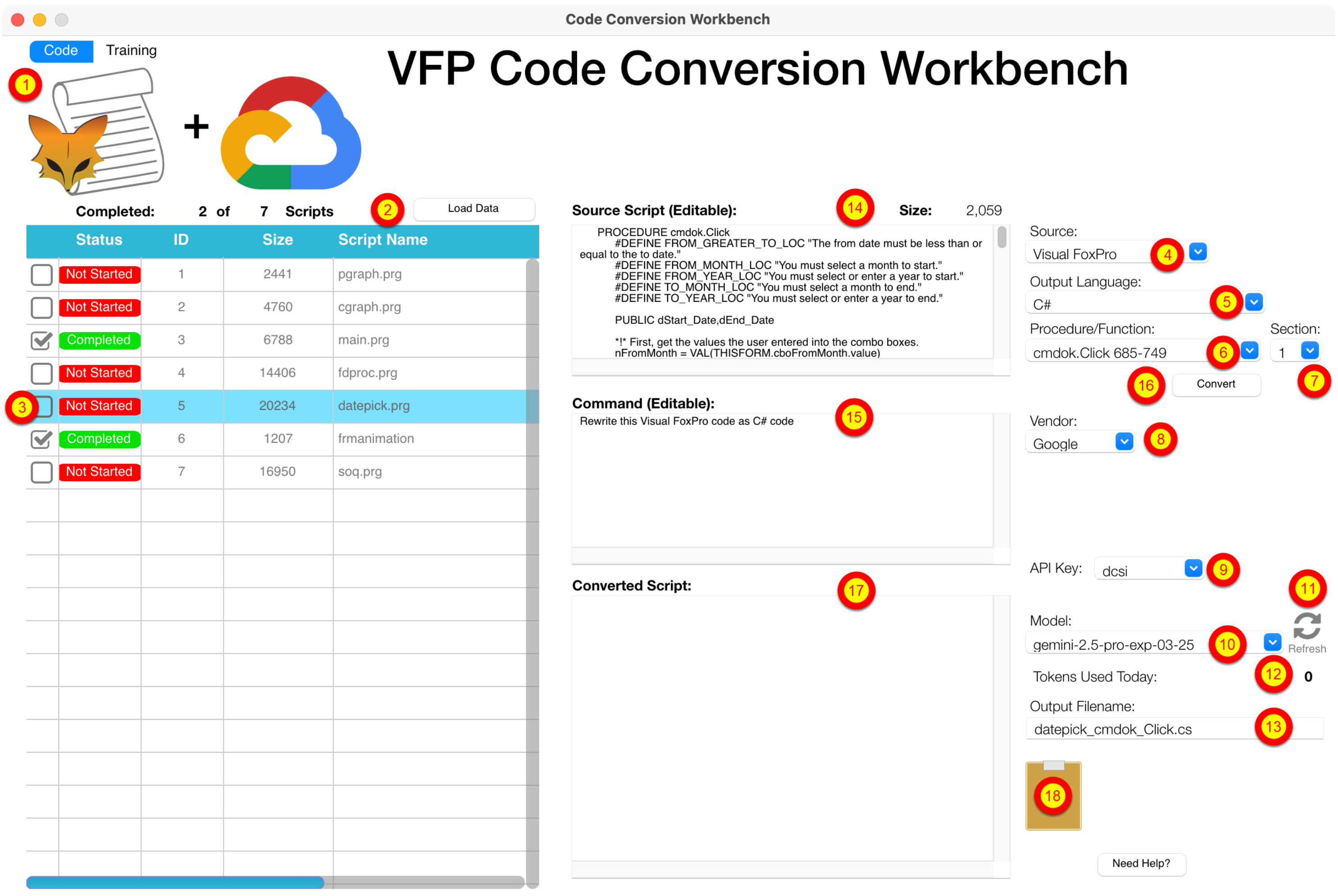
The Code Conversion Workbench consists of a single application window used to manage the conversion of hundreds or thousands of scripts into over 50 programming languages. As scripts are completed, they can be checked off in the grid of script on the left side of the window.

The source of scripts used by the Code Conversion Workbench is the MigrationProcess.db3 SQLite database file which is created and used by FmPro Migrator. Before performing code conversion, the scripts need to already be imported into the Scripts tab of the Migration Process window of FmPro Migrator.

Each source database is used to prefix the name of the code conversion, so if Visual FoxPro is the source database type, the title across the window will be: VFP Code Conversion Workbench.

Most screenshots in this manual are from macOS, but the tool works exactly the same way on Windows.

Code Conversion Workbench Window



VFP Code Conversion Workbench

Completed: 2 of 7 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	2441	pgraph.prg
☐	Not Started	2	4760	cgraph.prg
✓	Completed	3	6788	main.prg
☐	Not Started	4	14406	fdproc.prg
☐	Not Started	5	20234	datepick.prg
✓	Completed	6	1207	frmanimation
☐	Not Started	7	16950	soq.prg

Source Script (Editable): Size: 2,059

```
     PROCEDURE cmdok.Click
          #DEFINE FROM_GREATER_TO_LOC "The from date must be less than or
equal to the date."
          #DEFINE FROM_MONTH_LOC "You must select a month to start."
          #DEFINE FROM_YEAR_LOC "You must select or enter a year to start."
          #DEFINE TO_MONTH_LOC "You must select a month to end."
          #DEFINE TO_YEAR_LOC "You must select or enter a year to end."

          PUBLIC dStart_Date,dEnd_Date

          *!* First, get the values the user entered into the combo boxes.
          nFromMonth = VAL(THISFORM.cboFromMonth.value)
```

Command (Editable):

Rewrite this Visual FoxPro code as C# code

Converted Script:

Source: Visual FoxPro

Output Language: C#

Procedure/Function: cmdok.Click 685-749

Section: 1

Vendor: Google

API Key: dcsi

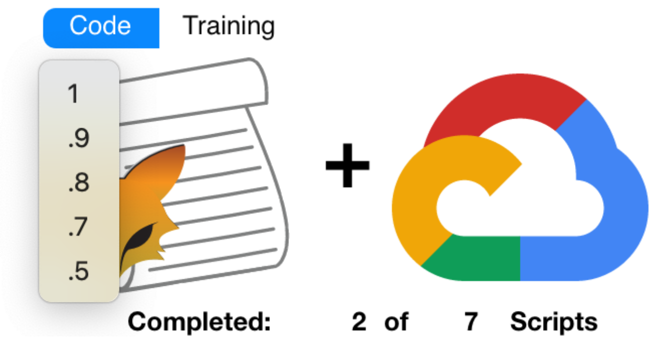
Model: gemini-2.5-pro-exp-03-25

Tokens Used Today: 0

Output Filename: datepick_cmdok_Click.cs

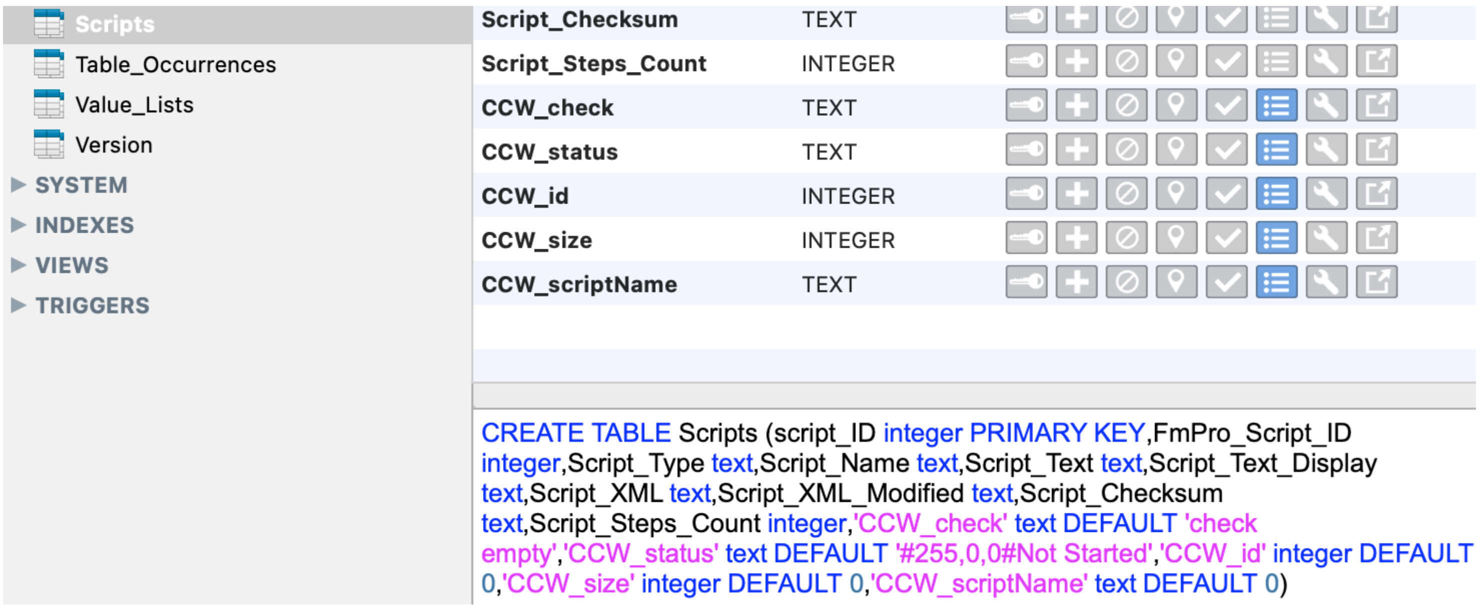
Most features in this window include tooltips, described in this manual.

1) Window Scale Factor - Pop-up Menu



The Code Conversion Workbench window will auto-scale in size based upon the size of your display. You can change the window scaling by selecting a different value from this popup menu. The Window Scale Factor Menu tooltip will be displayed when you move the cursor over this normally hidden popup menu.

2) Load Data Button



Script_Checksum	TEXT
Script_Steps_Count	INTEGER
CCW_check	TEXT
CCW_status	TEXT
CCW_id	INTEGER
CCW_size	INTEGER
CCW_scriptName	TEXT

```
CREATE TABLE Scripts (script_ID integer PRIMARY KEY,FmPro_Script_ID
integer,Script_Type text,Script_Name text,Script_Text text,Script_Text_Display
text,Script_XML text,Script_XML_Modified text,Script_Checksum
text,Script_Steps_Count integer,'CCW_check' text DEFAULT 'check
empty','CCW_status' text DEFAULT '#255,0,0#Not Started','CCW_id' integer DEFAULT
0,'CCW_size' integer DEFAULT 0,'CCW_scriptName' text DEFAULT 0)
```

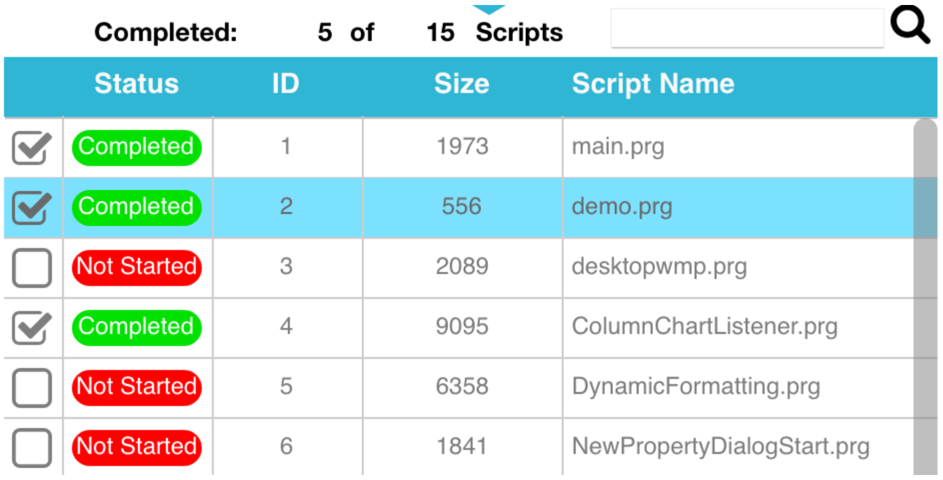
If there are scripts located within the FmPro Migrator MigrationProcess.db3 SQLite project file, the grid of scripts should be displayed automatically as soon as the window opens. If there weren't any scripts in the MigrationProcess.db3 file when the Code Conversion Workbench window was opened, the list of scripts would be empty. Once you import scripts using FmPro Migrator, click the Load Data button to populate this grid with the list of scripts.

The scripts and Status checkmarks/Completed/Not Started text displayed in this grid are read from the MigrationProcess.db3 file.

Note: If the Code Conversion Workbench is running in Demo mode, only a list of demo scripts will be displayed, not the actual scripts within the MigrationProcess.db3 file.

Note: Prior to FmPro Migrator 11.73, the ScriptsStatus.JSON file was used to store the status check marks. The contents of the ScriptsStatus.JSON file are now imported and updated within the extra columns added to the Scripts table of the MigrationProcess.db3 file. The previously used ScriptsStatus.JSON file is renamed to ScriptsStatus.JSON-old so that it won't be continuously re-imported. Triggers on the Script_Text column keep the CCW_Size column updated automatically.

3) Scripts Grid



Completed: 5 of 15 Scripts

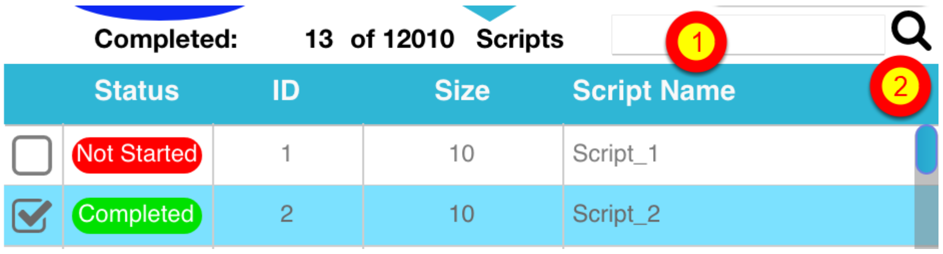
	Status	ID	Size	Script Name
✓	Completed	1	1973	main.prg
✓	Completed	2	556	demo.prg
☐	Not Started	3	2089	desktopwmp.prg
✓	Completed	4	9095	ColumnChartListener.prg
☐	Not Started	5	6358	DynamicFormatting.prg
☐	Not Started	6	1841	NewPropertyDialogStart.prg

The scripts grid provides a way for the developer to select specific scripts, convert them and check them off as being completed by clicking the checkmark column. The Completed ? of ? labels above the grid give a running count of the conversion status of the scripts.

Each script has an ID for reference purposes and its size in bytes and name are listed in the grid. Long script names are viewable with the scrollbar at the bottom of the grid.

Clicking any script causes it to be read from the database and displayed in the Source Script field. Clicking the script a 2nd time (to set its check/unchecked status) does not reload the script - in order to keep the contents of the Converted Script field visible.

Searching the Scripts Grid

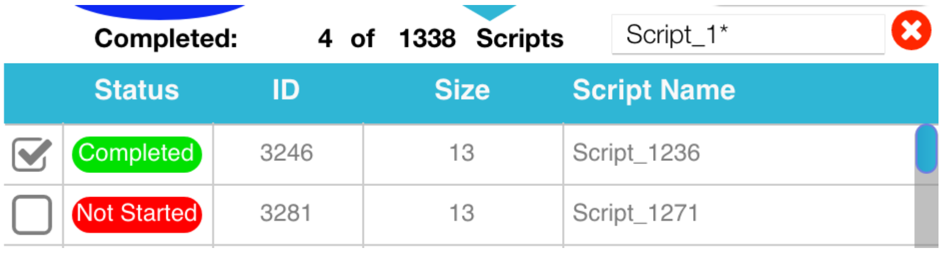


Completed: 13 of 12010 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	10	Script_1
✓	Completed	2	10	Script_2

(1) Click in the search field to search the scripts by name.

??* Begins With Search



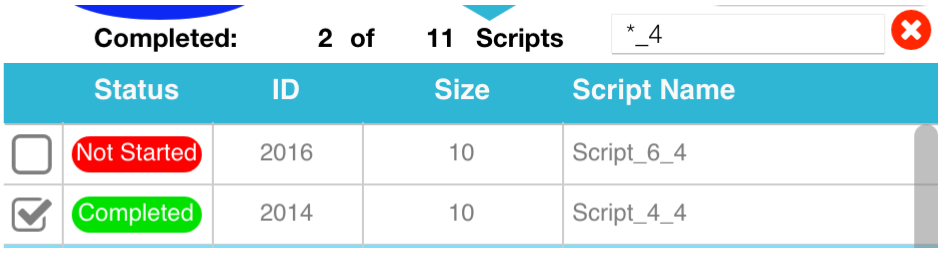
Completed: 4 of 1338 Scripts

Script_1*

	Status	ID	Size	Script Name
✓	Completed	3246	13	Script_1236
	Not Started	3281	13	Script_1271

Entering a * as the last character - performs a search for scripts which begin with the entered text. When the search results are displayed, the "Completed" totals will be recalculated above the grid. Clicking the checkmark column also updates the totals showing the number of completed scripts check marked within the total number of found scripts.

*?? Ends With Search



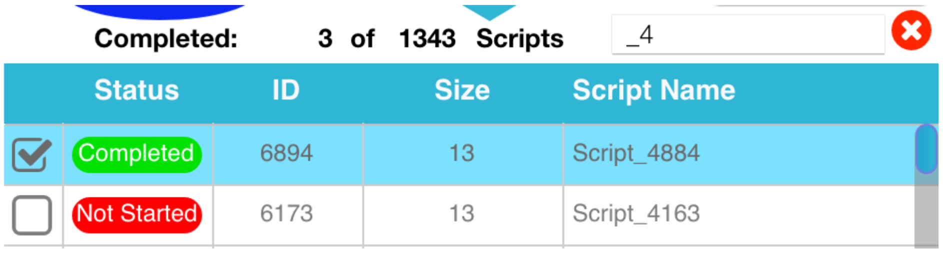
Completed: 2 of 11 Scripts

*_4

	Status	ID	Size	Script Name
	Not Started	2016	10	Script_6_4
✓	Completed	2014	10	Script_4_4

Entering a * as the first character - performs a search for scripts which end with the entered text.

Contains Text Search



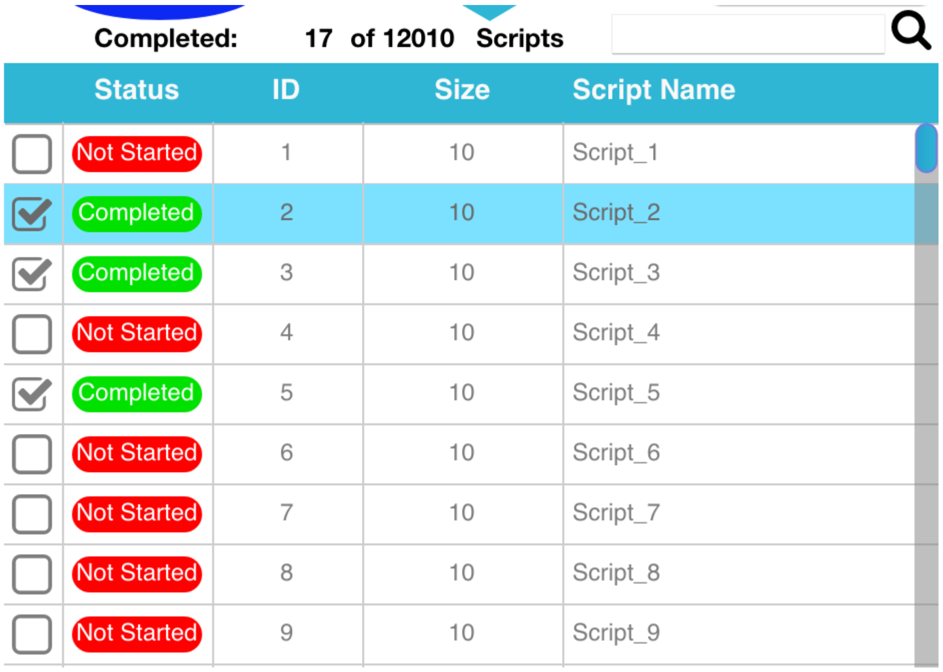
Completed: 3 of 1343 Scripts

_4

	Status	ID	Size	Script Name
✓	Completed	6894	13	Script_4884
	Not Started	6173	13	Script_4163

Search text without a modifier, performs a search for scripts which contain the entered text anywhere within the script name.

Grid Performance with Thousands of Scripts

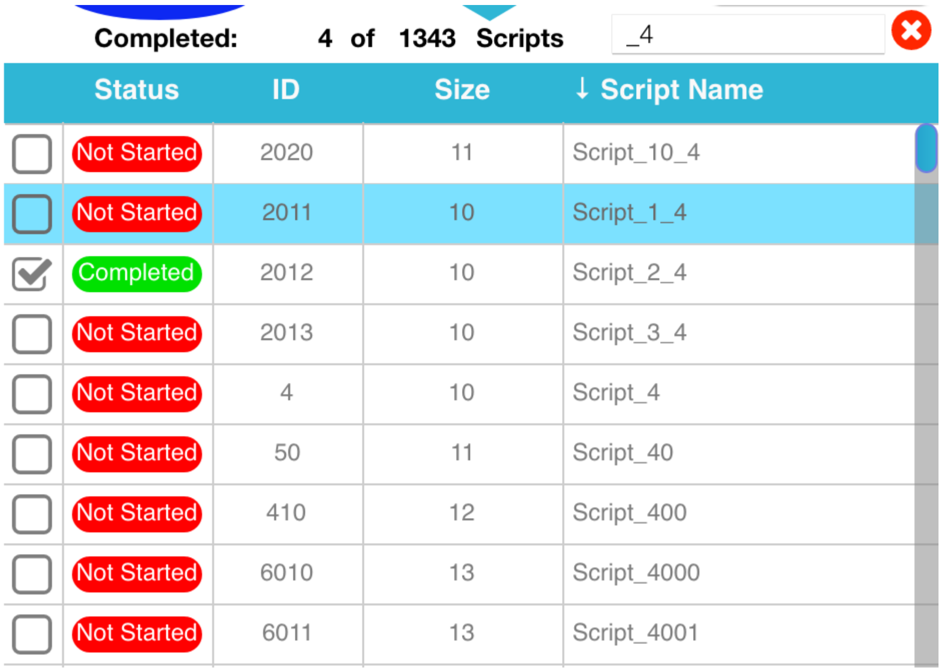


Completed: 17 of 12010 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	10	Script_1
☑	Completed	2	10	Script_2
☑	Completed	3	10	Script_3
☐	Not Started	4	10	Script_4
☑	Completed	5	10	Script_5
☐	Not Started	6	10	Script_6
☐	Not Started	7	10	Script_7
☐	Not Started	8	10	Script_8
☐	Not Started	9	10	Script_9

Agrid displaying 12,000 scripts will perform slower than a grid showing only a 1000 or fewer scripts! The synthetic test of over 12,000 scripts shown here takes about 6 seconds to perform a search and 11 seconds to register a checkmark click and about 6 seconds to register a click which displays a script in the Source Script field.
Performing a search showing about 1000 records improves the performance to be nearly instant. Therefore searching improves the usability of the UI while reducing the number of scripts to a manageable number.

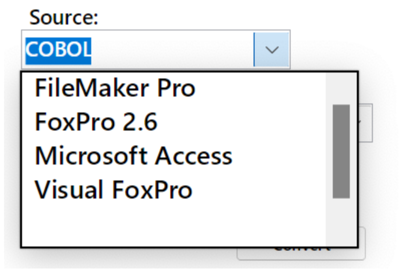
Scripts Grid Sort Feature


Completed: 4 of 1343 Scripts _4

	Status	ID	Size	↓ Script Name
☐	Not Started	2020	11	Script_10_4
☐	Not Started	2011	10	Script_1_4
☑	Completed	2012	10	Script_2_4
☐	Not Started	2013	10	Script_3_4
☐	Not Started	4	10	Script_4
☐	Not Started	50	11	Script_40
☐	Not Started	410	12	Script_400
☐	Not Started	6010	13	Script_4000
☐	Not Started	6011	13	Script_4001

Clicking the ID, Size or Script Name columns sorts the contents of the grid as a numeric sort (ID, Size columns) or a text sort (Script Name) column. Sorting works in search or non-search modes.

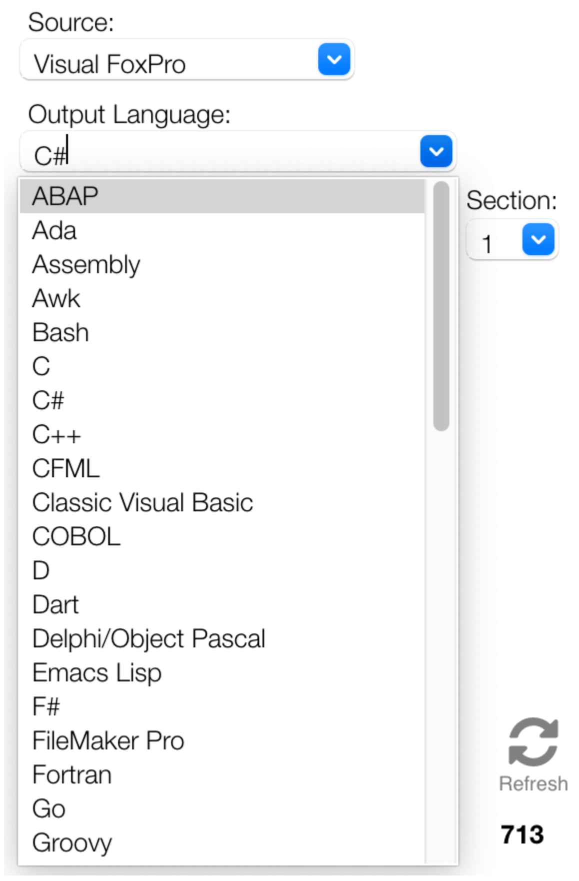
4) Source Database Type Menu



Source database types include: FileMaker Pro, FoxPro 2.6, Microsoft Access and Visual FoxPro as well as any type of script which has been imported into the Code Conversion Workbench like COBOL shown here. The Source field behind the menu is also directly editable.

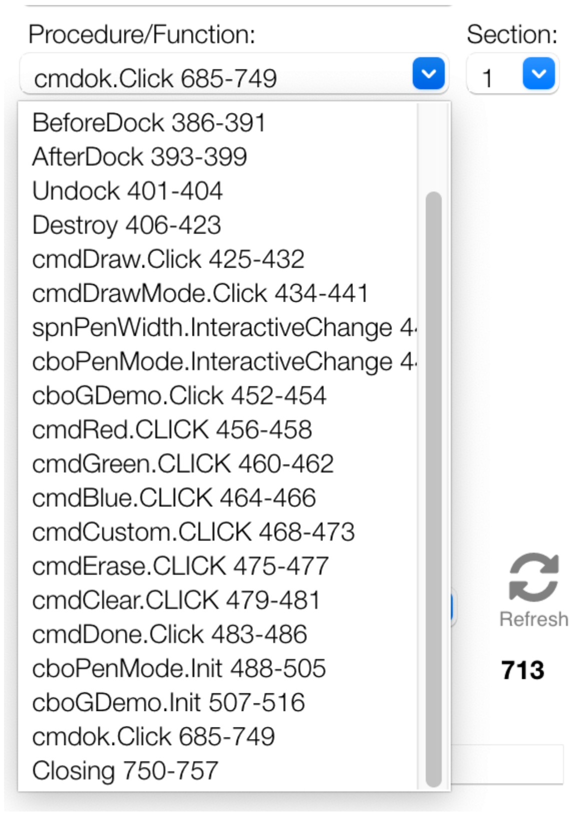
Selecting the Source Database changes the title across the top of the window and re-writes the contents of the Command window.

5) Output Language Menu


Source:
Visual FoxPro
Output Language:
C#
Section:
1

Selecting one of the over 50 output languages from the TIOBE index will rewrite the Command prompt for the selected language. Some of the languages include the specification of a framework and database name based upon the database selected on the main screen of FmPro Migrator as the output database. The text in the Command field can be manually updated to include any additional details required for the conversion.

6) Procedure/Function Menu

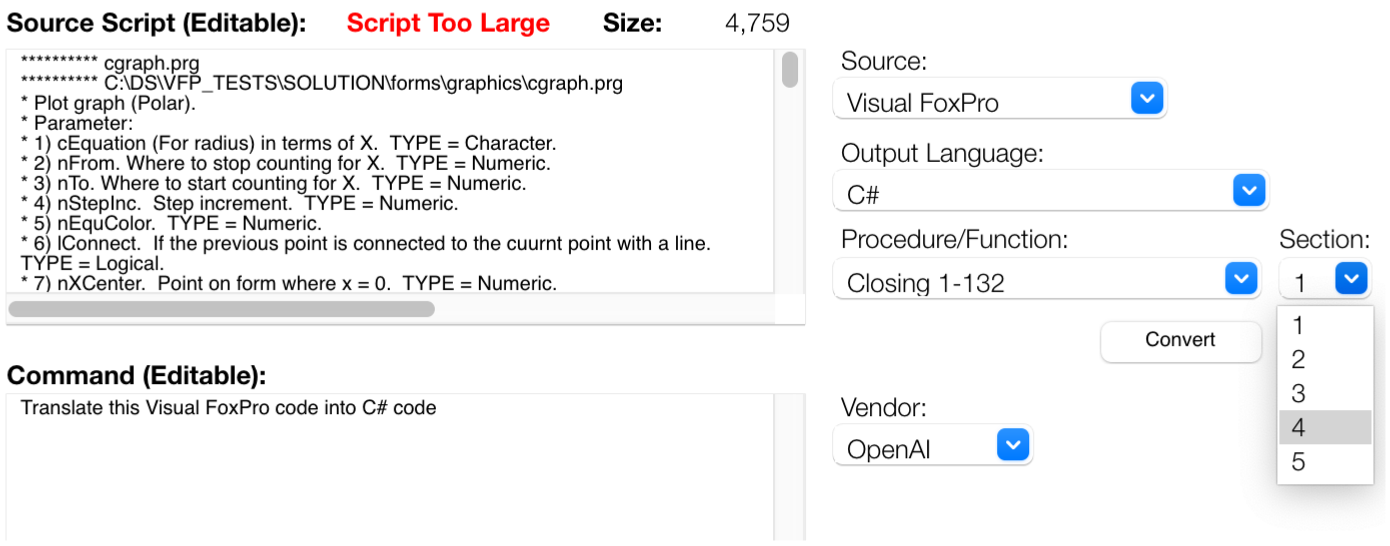


If the main script is too large for processing by a specific machine learning model, red text "Script Too Large" may be displayed above the Source script field. If this occurs, then it is possible to break of some scripts (like Visual FoxPro) automatically into individual procedures/functions individually.

Tip: Even if the "Script Too Large" text is displayed, go ahead and try converting the script anyhow - most of the time it will still work with modern LLMs.

Visual FoxPro scripts often contain multiple procedures/functions. If it is necessary to send smaller pieces of text to the AI models for processing, this menu enables selecting individual procedures/functions for sending to the AI model for processing. In general, it is best to send the entire script for processing so that the LLM has the context of the entire script.

7) Section Menu


Source Script (Editable): Script Too Large Size: 4,759

```
********** cgraph.prg
********** C:\DS\VFP_TESTS\SOLUTION\forms\graphics\cgraph.prg
* Plot graph (Polar).
* Parameter:
* 1) cEquation (For radius) in terms of X.  TYPE = Character.
* 2) nFrom. Where to stop counting for X.  TYPE = Numeric.
* 3) nTo. Where to start counting for X.  TYPE = Numeric.
* 4) nStepInc.  Step increment.  TYPE = Numeric.
* 5) nEquColor.  TYPE = Numeric.
* 6) lConnect.  If the previous point is connected to the cuurnt point with a line.
TYPE = Logical.
* 7) nXCenter.  Point on form where x = 0.  TYPE = Numeric.
```

Command (Editable):

Translate this Visual FoxPro code into C# code

Source: Visual FoxPro

Output Language: C#

Procedure/Function: Closing 1-132

Section: 1

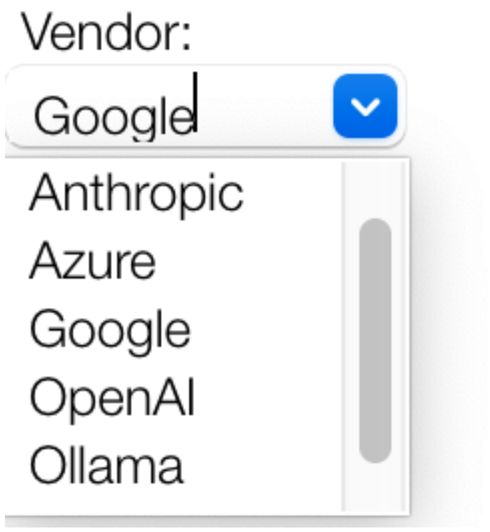
Vendor: OpenAI

Note: With recent timing improvements within the Code Conversion Workbench - breaking up the scripts further is not usually required anymore (unless you are using Ollama to run the models locally).

Some individual procedure/functions could still be too large, including the example shown here. If the "Script Too Large" script is displayed for an individual procedure/function then the Section sub-menu will also be displayed. This enables the large script to be broken up into 5 more sections (with opening/closing procedure/function text added to each section).

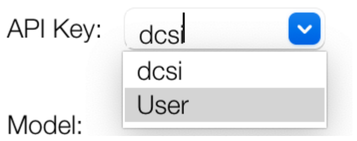
By the way, just because the "Script Too Large" text is displayed doesn't always mean that a script won't be processed by the AI model. You can always try sending a large script for processing, and then see if you get an error from the model.

8) Vendor Menu



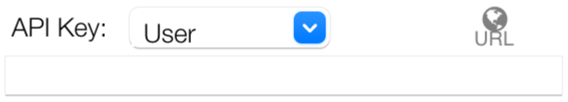
There are 7 AI vendors listed in the Vendor model at this time. Anthropic, Google, OpenAI, Ollama and xAI are the vendors supported at this time, as the others are intended for future enhancements.
Ollama enables running LLMs locally and is reserved for Custom Dev and Site License Editions of FmPro Migrator.

9) API Key Type



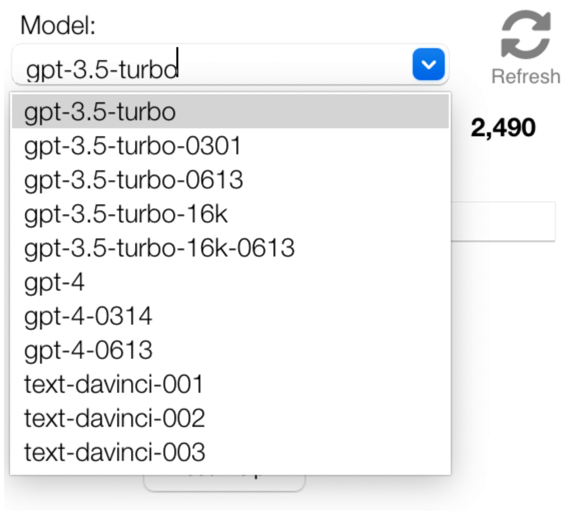
By default, the Code Conversion Workbench will use the API key created by .com Solutions Inc., so "dcsi" is the default option. If you run out of AI tokens, you can generate and pay for your own API key at OpenAI.

User API Key Field



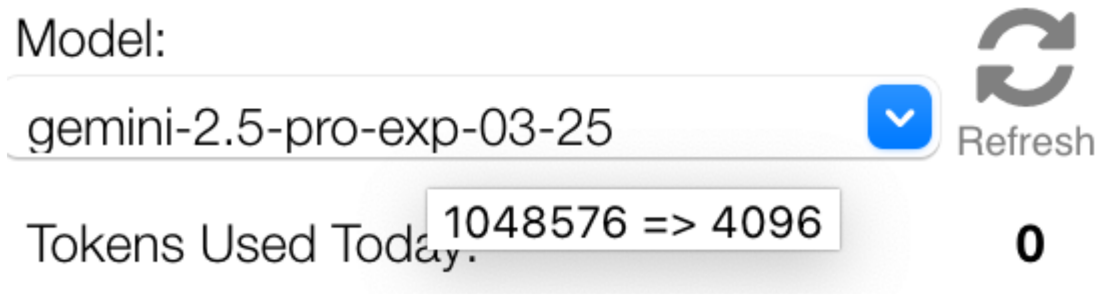
To enter your own API key, select the User menu item, a new API Key field will be shown. Clicking the URL button opens a web browser to the vendor website where you can create an account and generate your own API key. This key gets saved with the app preferences for each vendor so you only have to enter it once.

10) AI Model Name



gpt-3.5-turbo will be selected as the default model. gpt-4 is also available, and generally does a significantly better job when performing conversions. But sometimes one model or the other gets overloaded so you might want to switch to another one.

Model Token Size Tooltip

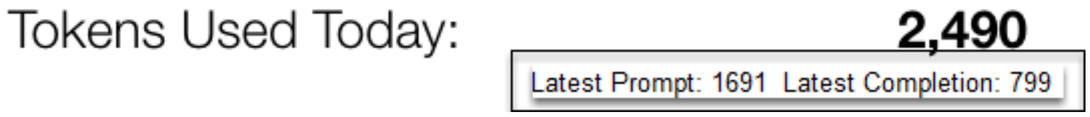


Hovering the cursor over the model menu, displays the number of tokens accepted by the selected AI model. The first number represents the number of input tokens, and the 2nd number represents the number of output tokens supported by the model. These numbers may not always be updated perfectly with each model as it is based upon values written into the Code Conversion Workbench - so new models could have higher values.

11) Model List Refresh Button

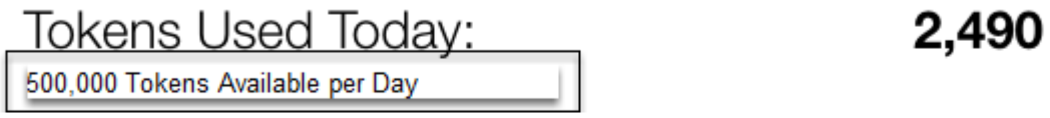
The list of available AI models is built into the app during development. You can refresh the list of models by clicking the Refresh button. The list of models displayed in the Model menu represents the models which are officially supported by the Code Conversion Workbench. Holding the Shift key when clicking the Refresh button displays all of the models available from the selected vendor. Selecting one of the unsupported models will generally result in an error message. Updating the list of models with the refresh button does save these models to the FmPro Migrator preferences stored on disk for each vendor. When first opened, the Code Conversion Workbench includes the lists of models which were available during development.

12) Tokens Used Today Label & Usage Tooltip



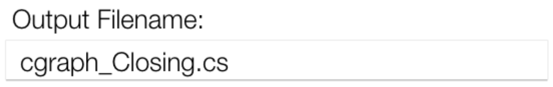
The number to the right of the Tokens Used Today text label shows the number of tokens used over the last 24 hours. Hovering the cursor over this number will display the number of tokens used during the most recent script conversion. In this example, 1691 tokens were used for the prompt and 799 tokens were used for the completed script which was returned by the model.

Tokens Available per Day Tooltip



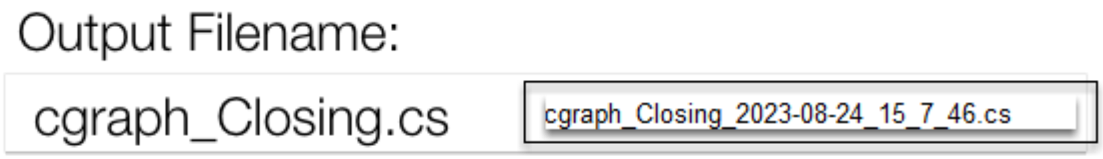
Hovering the cursor over the Tokens Used Today label will display the number of tokens which can be used per day for the currently selected AI model. At the present time, 500,000 tokens per day are provided for gpt3.5-turbo and the rest of the models except for gpt4. Since it currently costs 20 times as much to use the gpt4 model, the number of tokens available per day is reduced to 25,000 when gpt4 is being used. These calculations will change over time as costs change.

13) Output Filename Field



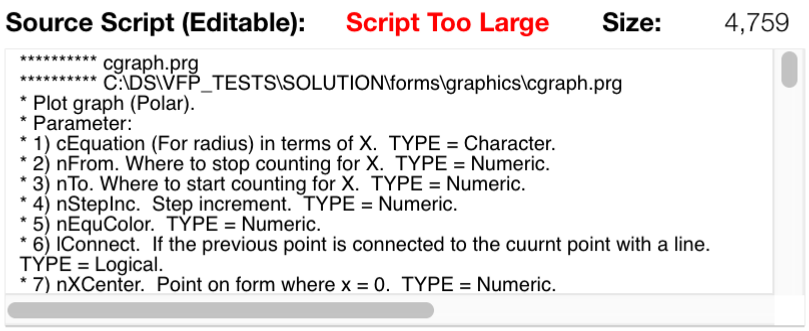
The name of the output script is created and saved automatically into the Converted_Scripts folder, but it can be changed manually before pressing the Convert button. In this example the filename consists of the name of the script followed by the name of the function and ending with the C# file extension.

Output Filename Tooltip



The tooltip shows the actual filename written to disk. Duplicated scripts receive a timestamp in order to preserve the original script, as you can see with this example.

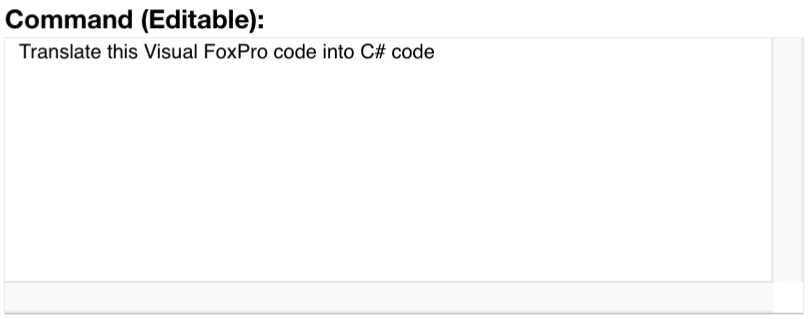
14) Source Script Field



```
********** cgraph.prg
********** C:\DS\VFP_TESTS\SOLUTION\forms\graphics\cgraph.prg
* Plot graph (Polar).
* Parameter:
* 1) cEquation (For radius) in terms of X.  TYPE = Character.
* 2) nFrom. Where to stop counting for X.  TYPE = Numeric.
* 3) nTo. Where to start counting for X.  TYPE = Numeric.
* 4) nStepInc.  Step increment.  TYPE = Numeric.
* 5) nEquColor.  TYPE = Numeric.
* 6) lConnect.  If the previous point is connected to the cuurnt point with a line.
TYPE = Logical.
* 7) nXCenter.  Point on form where x = 0.  TYPE = Numeric.
```

Each script is read from the MigrationProcess.db3 SQLite project file when it is clicked in the scripts list grid. The script is written into the Source Script field, which can be manually edited if needed.
The size of the script in characters is displayed above the top right of the field next to the Size label. If the size of the script is estimated to be too large for the number of tokens available for the selected model, the "Script Too Large" text will be displayed above the script. However most of the time, you can now ignore this warning text with modern LLMs which have large context sizes.

15) Command Field



Command (Editable):

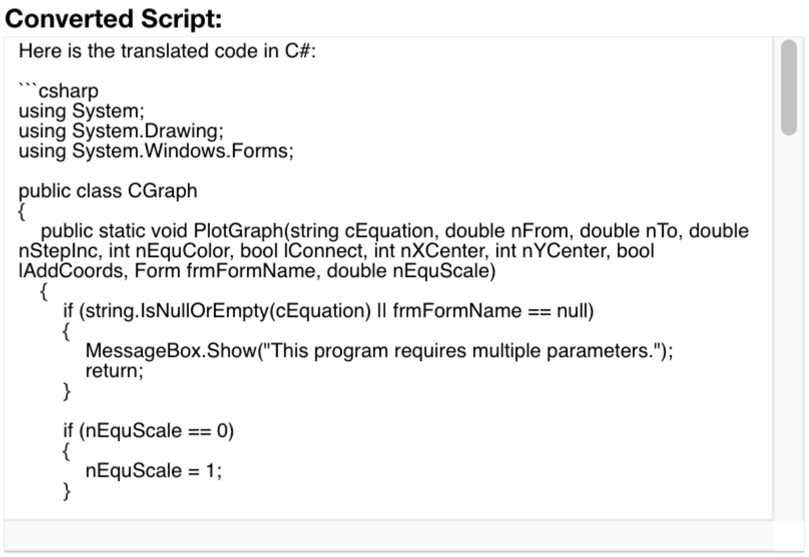
Translate this Visual FoxPro code into C# code

This field provides the prompt which will be sent to the AI model. The default command for this field is created automatically when selecting the Source database type or Output Language. It is also editable with additional descriptive commands such as the programming language framework or database type.

16) Convert Button

The Convert button sends the Source script, Command and system message/properties to the AI model for processing. The results are written into the Converted Script field.

17) Converted Script Field



Converted Script:

Here is the translated code in C#:

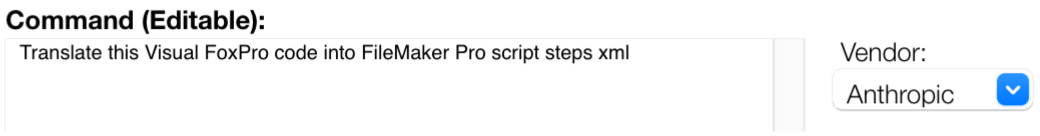
~~~
```csharp
using System;
using System.Drawing;
using System.Windows.Forms;

public class CGraph
{
    public static void PlotGraph(string cEquation, double nFrom, double nTo, double
nStepInc, int nEquColor, bool lConnect, int nXCenter, int nYCenter, bool
lAddCoords, Form frmFormName, double nEquScale)
    {
        if (string.IsNullOrEmpty(cEquation) || frmFormName == null)
        {
            MessageBox.Show("This program requires multiple parameters.");
            return;
        }

        if (nEquScale == 0)
        {
            nEquScale = 1;
        }
~~~

After clicking the Convert button, the generated script is put into the Converted Script field, and also written into the Converted_Scripts folder within the project directory.

18) Clipboard Icon

Clicking the clipboard icon copies the contents of the Converted Script field onto the clipboard, ready for pasting into your IDE of choice.
When FileMaker Pro is selected as the Output Language the converted script text is converted into commented FileMaker script XML code and placed onto the clipboard in a format which can be directly pasted into the FileMaker Pro Script Workspace window.

18) Clipboard Icon - Anthropic/Google Gemini 3.0 Pro & FileMaker Pro Script XML



Command (Editable):

Translate this Visual FoxPro code into FileMaker Pro script steps xml

Vendor: Anthropic

Anthropic and Google Gemini Pro 3 are the only models found so far which can produce executable script XML for FileMaker Pro databases. When using Anthropic/Google as a vendor with FileMaker Pro as the destination database, the prompt will include "<u>script steps xml</u>" instead of just "<u>script steps</u>" which is used for the other vendors. The Anthropic Claude 3.5+ and Google Gemini 3 Pro models will attempt to create executable FileMaker script XML code.

Since this code doesn't always work perfectly, it is recommended that you generate the script twice, once using the "script steps xml" prompt and once again using the more generic "script steps" prompt. If the Code Conversion Workbench finds the text "<fmxmlsnippet" within the script, it will put the script onto the clipboard as a functional FileMaker script - ready for pasting into the Script Workspace.

The 2nd time you generate the script, just get the text version of the script and paste it into FileMaker also. This way you can compare the two scripts to see if anything is missing when it gets pasted into FileMaker Pro.

Detailed Code Conversion Info

Using the VFP Code Conversion Workbench

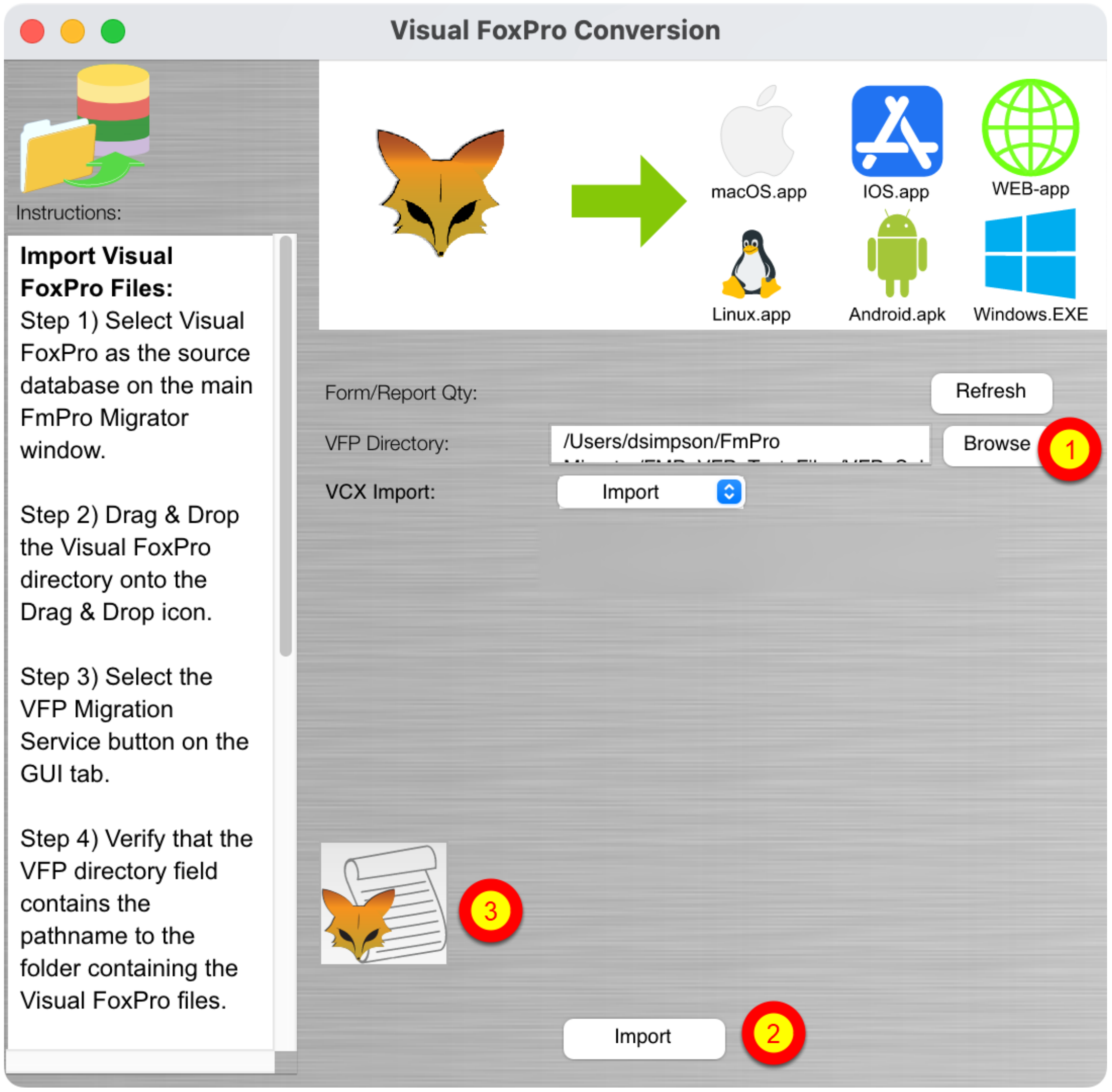
This chapter shows how to use the VFP Code Conversion Workbench. Before performing the code conversion with the VFP Code Conversion Workbench, you should have already imported the Visual FoxPro project into FmPro Migrator, including tables, relationships, value lists, forms/reports and scripts.

Visual FoxPro Conversion Button - GUI Tab

Visual FoxPro Conversion

Click the Visual FoxPro Conversion button on the GUI tab of the Migration Process window of FmPro Migrator.

Visual FoxPro Conversion Window



Instructions:

Import Visual FoxPro Files:
Step 1) Select Visual FoxPro as the source database on the main FmPro Migrator window.

Step 2) Drag & Drop the Visual FoxPro directory onto the Drag & Drop icon.

Step 3) Select the VFP Migration Service button on the GUI tab.

Step 4) Verify that the VFP directory field contains the pathname to the folder containing the Visual FoxPro files.

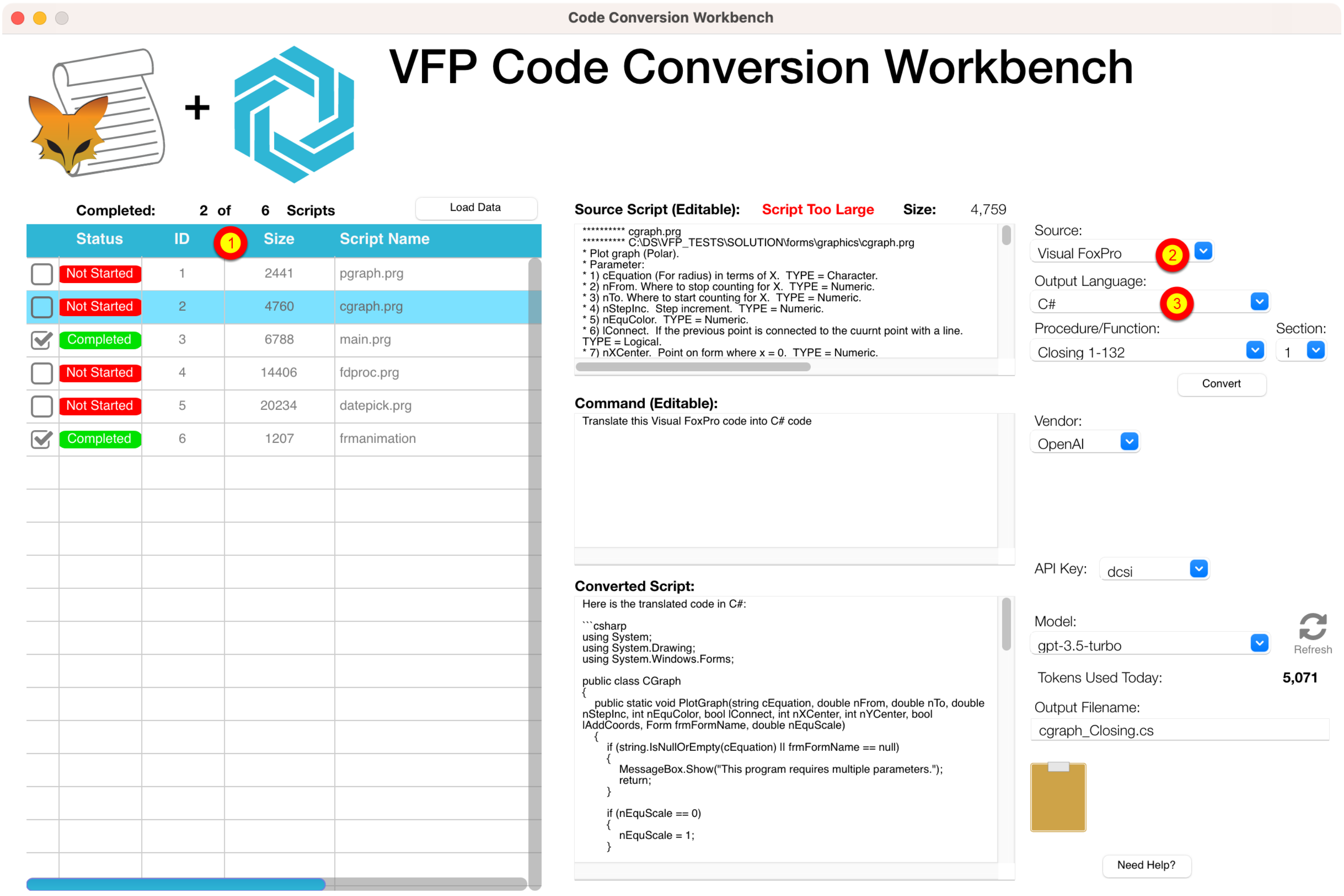
Form/Report Qty:

VFP Directory: /Users/dsimpson/FmPro

VCX Import: Import

Prior to reaching this step, you should have already selected the (1) selected the VFPExport.DBF project folder and (2) clicked the Import button to import the VFP project into FmPro Migrator. You may have already started converting the VFP project into another development environment which would mean that you are ready (3) to click the Code Conversion Workbench button.

VFP Code Conversion Workbench Window



Completed: 2 of 6 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	2441	pgraph.prg
☐	Not Started	2	4760	cgraph.prg
✓	Completed	3	6788	main.prg
☐	Not Started	4	14406	fdproc.prg
☐	Not Started	5	20234	datepick.prg
✓	Completed	6	1207	frmanimation

Source Script (Editable): Script Too Large Size: 4,759

```
********** cgraph.prg
********** C:\DS\VFP_TESTS\SOLUTION\forms\graphics\cgraph.prg
* Plot graph (Polar).
* Parameter:
* 1) cEquation (For radius) in terms of X.  TYPE = Character.
* 2) nFrom. Where to stop counting for X.  TYPE = Numeric.
* 3) nTo. Where to start counting for X.  TYPE = Numeric.
* 4) nStepInc.  Step increment.  TYPE = Numeric.
* 5) nEquColor.  TYPE = Numeric.
* 6) lConnect.  If the previous point is connected to the cuurnt point with a line.
TYPE = Logical.
* 7) nXCenter.  Point on form where x = 0.  TYPE = Numeric.
```

Command (Editable):

```
Translate this Visual FoxPro code into C# code
```

Converted Script:

~~~
Here is the translated code in C#:

```csharp
using System;
using System.Drawing;
using System.Windows.Forms;

public class CGraph
{
   public static void PlotGraph(string cEquation, double nFrom, double nTo, double
nStepInc, int nEquColor, bool lConnect, int nXCenter, int nYCenter, bool
lAddCoords, Form frmFormName, double nEquScale)
   {
      if (string.IsNullOrEmpty(cEquation) || frmFormName == null)
      {
         MessageBox.Show("This program requires multiple parameters.");
         return;
      }

      if (nEquScale == 0)
      {
         nEquScale = 1;
      }
~~~

Source: Visual FoxPro

Output Language: C#

Procedure/Function: Closing 1-132

Section: 1

Vendor: OpenAI

API Key: dcsi

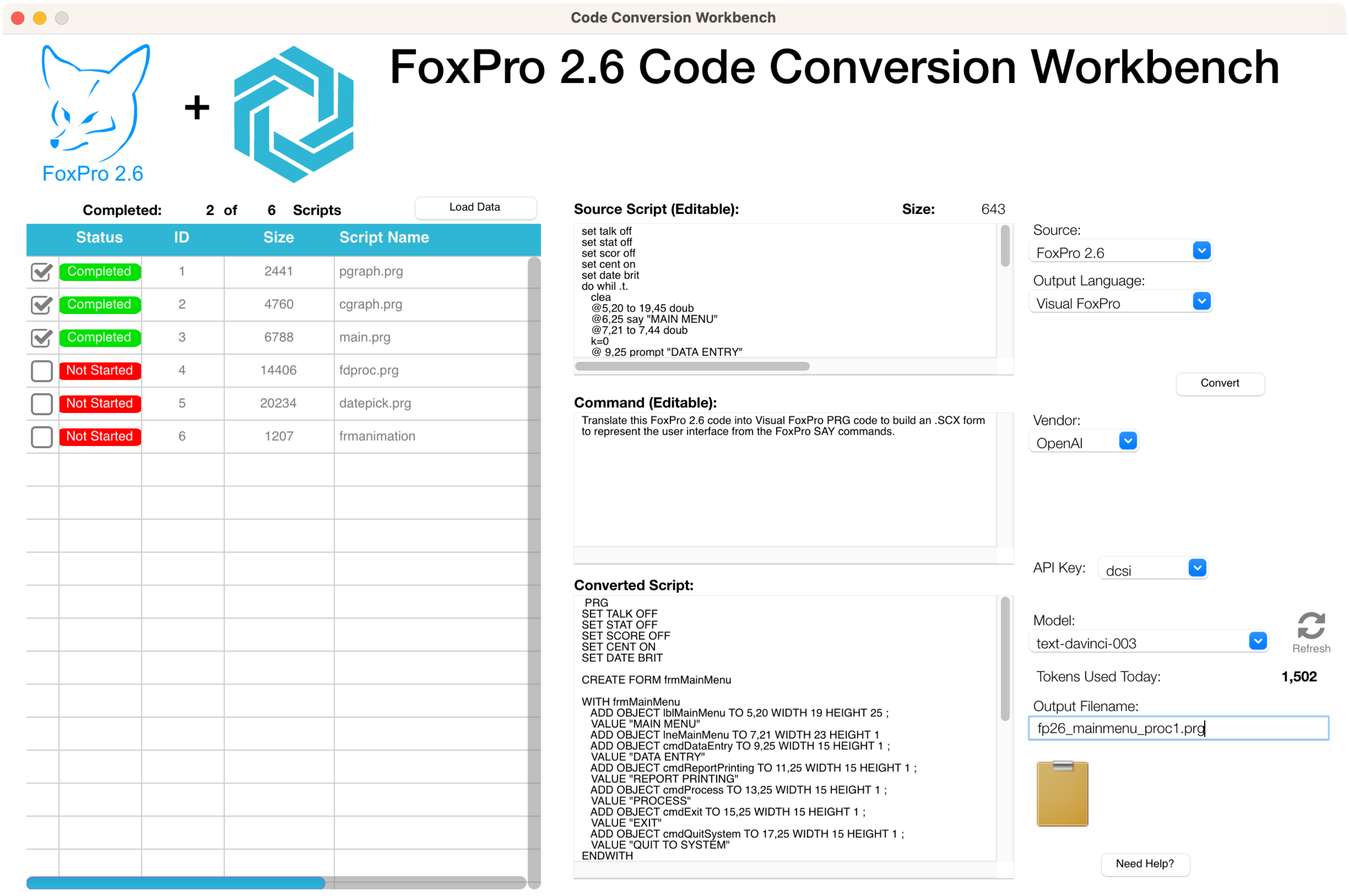
Model: gpt-3.5-turbo

Tokens Used Today: 5,071

Output Filename: cgraph_Closing.cs

By default, the VFP Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) Visual FoxPro selected as the Source Database Type and (3) C# selected as the destination language.
Depending upon the destination database selected on the main FmPro Migrator window, you could also see FileMaker Pro or Microsoft Access VBA selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them.
You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.



FoxPro 2.6 Code Conversion Workbench

Completed: 2 of 6 Scripts

Status	ID	Size	Script Name
✓ Completed	1	2441	pgraph.prg
✓ Completed	2	4760	cgraph.prg
✓ Completed	3	6788	main.prg
Not Started	4	14406	fdproc.prg
Not Started	5	20234	datepick.prg
Not Started	6	1207	frmanimation

Source Script (Editable): Size: 643

```
set talk off
set stat off
set scor off
set cent on
set date brit
do whil .t.
  clea
  @5,20 to 19,45 doub
  @6,25 say "MAIN MENU"
  @7,21 to 7,44 doub
  k=0
  @ 9,25 prompt "DATA ENTRY"
```

Command (Editable):

Translate this FoxPro 2.6 code into Visual FoxPro PRG code to build an .SCX form to represent the user interface from the FoxPro SAY commands.

Converted Script:

```
 PRG
SET TALK OFF
SET STAT OFF
SET SCORE OFF
SET CENT ON
SET DATE BRIT

CREATE FORM frmMainMenu

WITH frmMainMenu
  ADD OBJECT lblMainMenu TO 5,20 WIDTH 19 HEIGHT 25 ;
  VALUE "MAIN MENU"
  ADD OBJECT lneMainMenu TO 7,21 WIDTH 23 HEIGHT 1
  ADD OBJECT cmdDataEntry TO 9,25 WIDTH 15 HEIGHT 1 ;
  VALUE "DATA ENTRY"
  ADD OBJECT cmdReportPrinting TO 11,25 WIDTH 15 HEIGHT 1 ;
  VALUE "REPORT PRINTING"
  ADD OBJECT cmdProcess TO 13,25 WIDTH 15 HEIGHT 1 ;
  VALUE "PROCESS"
  ADD OBJECT cmdExit TO 15,25 WIDTH 15 HEIGHT 1 ;
  VALUE "EXIT"
  ADD OBJECT cmdQuitSystem TO 17,25 WIDTH 15 HEIGHT 1 ;
  VALUE "QUIT TO SYSTEM"
ENDWITH
```

Source: FoxPro 2.6
Output Language: Visual FoxPro
Vendor: OpenAI
API Key: dcsi
Model: text-davinci-003
Tokens Used Today: 1,502
Output Filename: fp26_mainmenu_proc1.prg

Selecting FoxPro 2.6 as the Source Database type enables the conversion of FoxPro 2.6 code into Visual FoxPro code, including the creation of forms from the FoxPro 2.6 command line code. Performing this conversion makes use of the text-davinci-003 model.
You will also need to break up the size of the script text manually, if there aren't any procedures/functions.

In order to prepare the FoxPro 2.6 project for import into FmPro Migrator, perform the following steps:

1) Install Visual FoxPro 9.
2) Make a copy of the FoxPro 2.6 project.
3) Open and convert the copied project in Visual FoxPro 9. This will upgrade all of the files including the DBF files. This is why you want to work with a copy of the project - because the DBF files won’t be readable in FoxPro after conversion. And you might need to continue developing the existing FoxPro app while performing your conversion.
4) Once the FoxPro 2.6 project has been converted into Visual FoxPro 9, create a DBC and add all of the DBF files to the DBC.

5) Manually add 1 empty form to the Visual FoxPro 9 project.

Now you are ready to follow the existing PDF manual showing how to import the project into FmPro Migrator for conversion.

VFP Code Conversion Workbench Demo

VFP Code Conversion Workbench Demo

If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the VFP Code Conversion Workbench will open in Demo mode. Apre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own Visual FoxPro project.

Clicking the Convert button will place the pre-converted script into the Converted Script field.

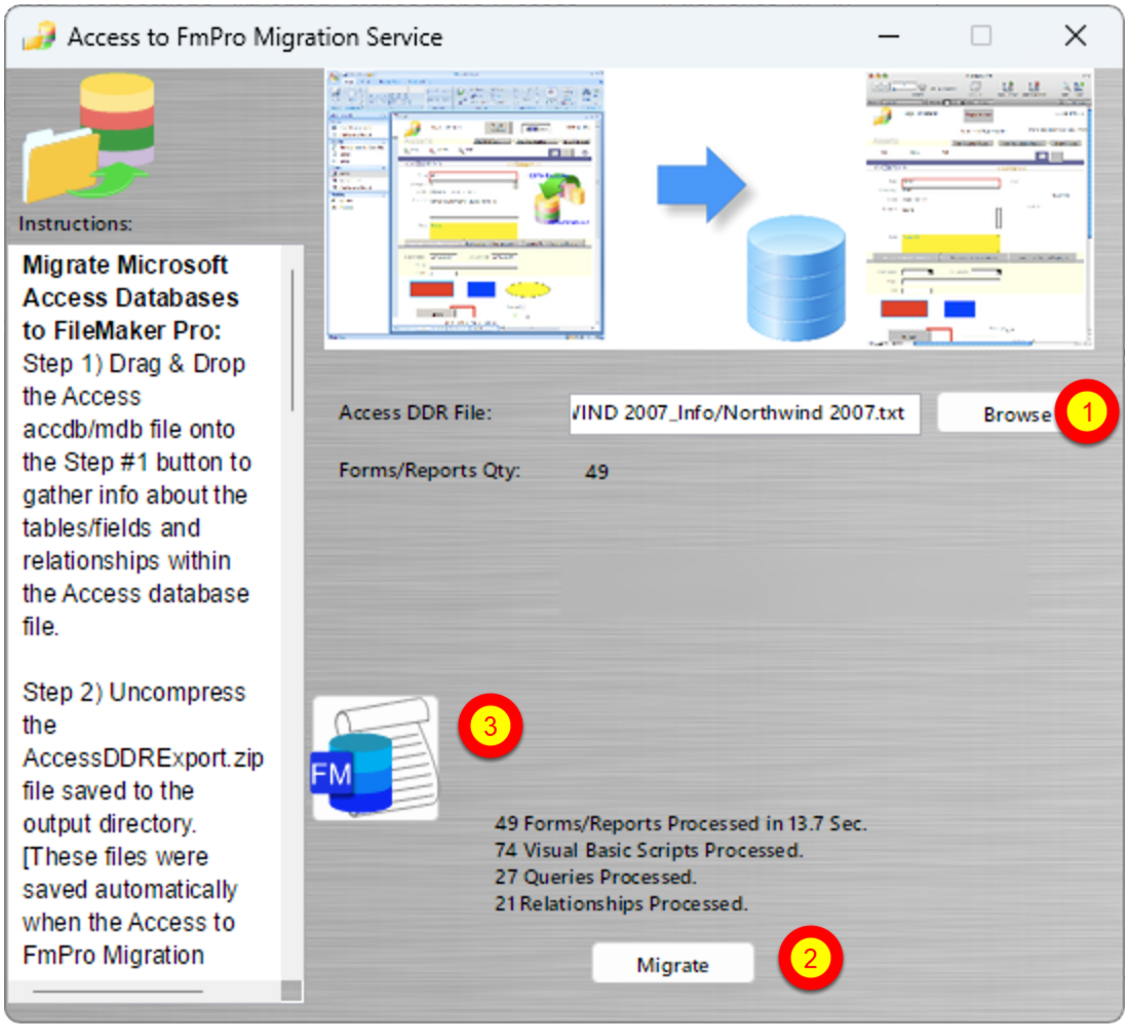
Using the Microsoft Access Code Conversion Workbench - Access to FmPro Conversions

This chapter shows how to use the Microsoft Access Code Conversion Workbench - included with the AI Accelerated version of FmPro Migrator. This feature replaces copying the unconverted scripts into the FileMaker Pro database as was done previously. Before performing the code conversion with the Microsoft Access Code Conversion Workbench, you should have already imported the Microsoft Access database into FmPro Migrator, including tables, relationships, value lists, forms/reports and scripts.

Access to FmPro Migration Button - GUI Tab

Click the Access to FmPro Migration button on the GUI tab of the Migration Process window of FmPro Migrator.

Access to FmPro Migration Window



Migrate Microsoft Access Databases to FileMaker Pro:
Step 1) Drag & Drop the Access accdb/mdb file onto the Step #1 button to gather info about the tables/fields and relationships within the Access database file.

Step 2) Uncompress the AccessDDRExport.zip file saved to the output directory. [These files were saved automatically when the Access to FmPro Migration

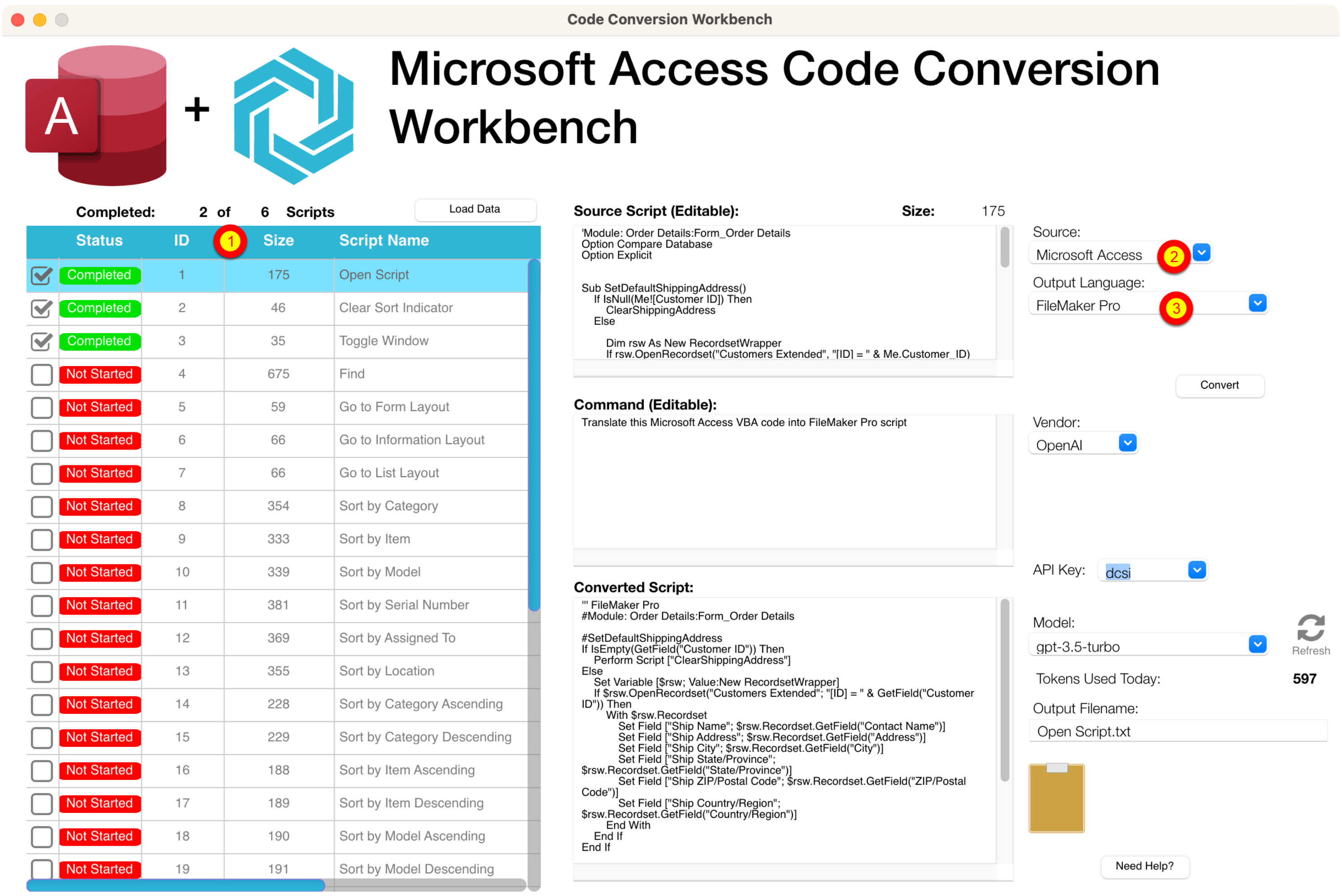
Access DDR File: VIND 2007_Info/Northwind 2007.txt

Forms/Reports Qty: 49

49 Forms/Reports Processed in 13.7 Sec.
74 Visual Basic Scripts Processed.
27 Queries Processed.
21 Relationships Processed.

Prior to reaching this step, you should have, (1) selected the AccessDDRExport text file and (2) clicked the Migrate button to import the Access database metadata into FmPro Migrator. You may have already started building the new FileMaker Pro database which would mean that you are ready (3) to click the Code Conversion Workbench button.

Microsoft Access Code Conversion Workbench Window



Microsoft Access Code Conversion Workbench

Completed: 2 of 6 Scripts

Status	ID	Size	Script Name
✓ Completed	1	175	Open Script
✓ Completed	2	46	Clear Sort Indicator
✓ Completed	3	35	Toggle Window
Not Started	4	675	Find
Not Started	5	59	Go to Form Layout
Not Started	6	66	Go to Information Layout
Not Started	7	66	Go to List Layout
Not Started	8	354	Sort by Category
Not Started	9	333	Sort by Item
Not Started	10	339	Sort by Model
Not Started	11	381	Sort by Serial Number
Not Started	12	369	Sort by Assigned To
Not Started	13	355	Sort by Location
Not Started	14	228	Sort by Category Ascending
Not Started	15	229	Sort by Category Descending
Not Started	16	188	Sort by Item Ascending
Not Started	17	189	Sort by Item Descending
Not Started	18	190	Sort by Model Ascending
Not Started	19	191	Sort by Model Descending

Source Script (Editable): **Size:** 175

```
'Module: Order Details:Form_Order Details
Option Compare Database
Option Explicit


Sub SetDefaultShippingAddress()
    If IsNull(Me![Customer ID]) Then
        ClearShippingAddress
    Else

        Dim rsw As New RecordsetWrapper
        If rsw.OpenRecordset("Customers Extended", "[ID] = " & Me.Customer_ID)
```

Command (Editable):

```
Translate this Microsoft Access VBA code into FileMaker Pro script
```

Converted Script:

```
''' FileMaker Pro
#Module: Order Details:Form_Order Details

#SetDefaultShippingAddress
If IsEmpty(GetField("Customer ID")) Then
    Perform Script ["ClearShippingAddress"]
Else
    Set Variable [$rsw; Value:New RecordsetWrapper]
    If $rsw.OpenRecordset("Customers Extended"; "[ID] = " & GetField("Customer
ID")) Then
        With $rsw.Recordset
            Set Field ["Ship Name"; $rsw.Recordset.GetField("Contact Name")]
            Set Field ["Ship Address"; $rsw.Recordset.GetField("Address")]
            Set Field ["Ship City"; $rsw.Recordset.GetField("City")]
            Set Field ["Ship State/Province";
$rsw.Recordset.GetField("State/Province")]
            Set Field ["Ship ZIP/Postal Code"; $rsw.Recordset.GetField("ZIP/Postal
Code")]
            Set Field ["Ship Country/Region";
$rsw.Recordset.GetField("Country/Region")]
        End With
    End If
End If
```

Source: Microsoft Access

Output Language: FileMaker Pro

Vendor: OpenAI

API Key: dcsi

Model: gpt-3.5-turbo

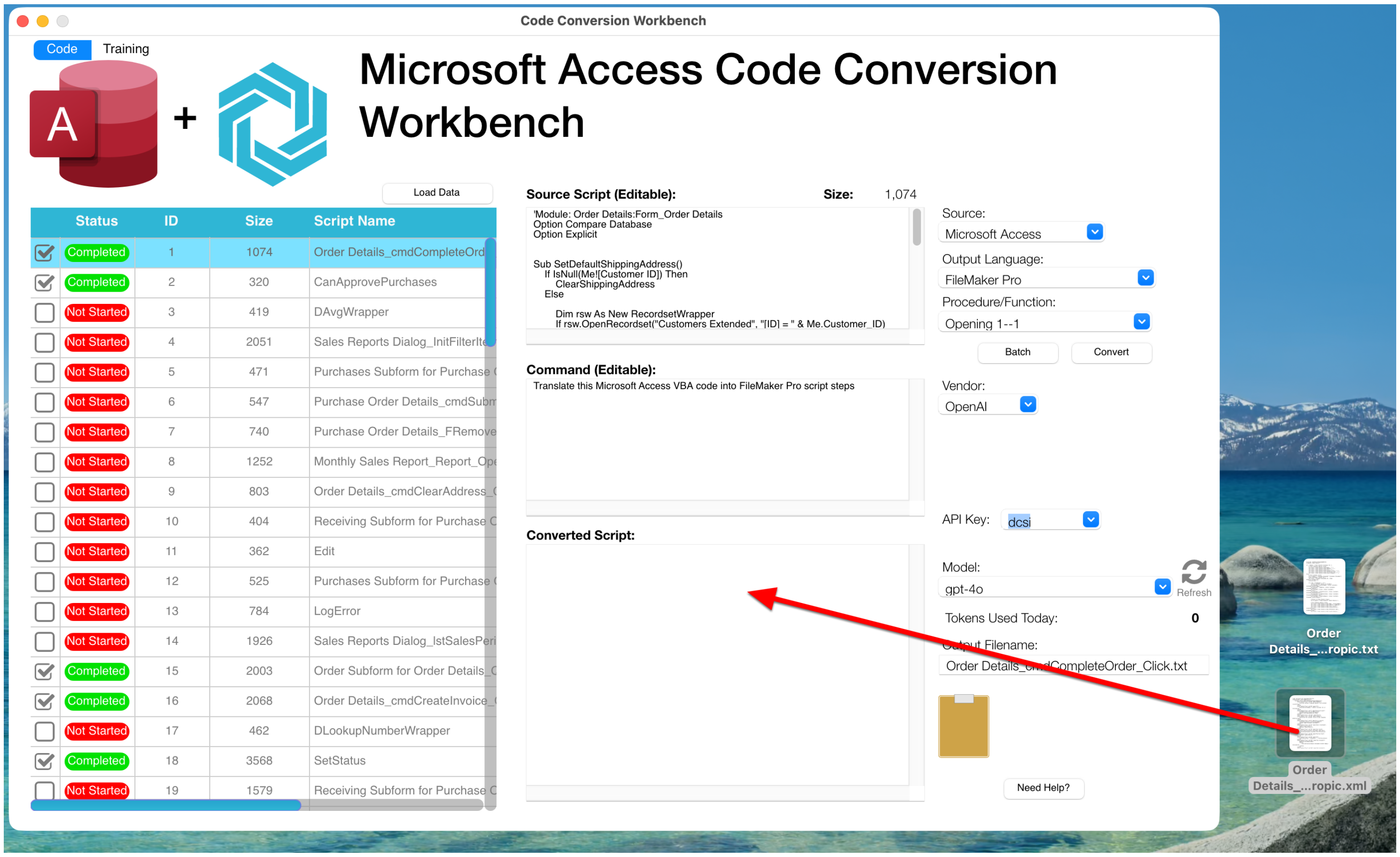
Tokens Used Today: 597

Output Filename: Open Script.txt

By default, the Microsoft Access Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) Microsoft Access selected as the Source Database Type and (3) FileMaker Pro selected as the destination language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

Using Drag & Drop with the Converted Script Field


Code Conversion Workbench
Code
Training
Load Data
Batch
Convert
Refresh
Need Help?
Order Details_...ropic.txt
Order Details_...ropic.xml

Microsoft Access Code Conversion Workbench

	Status	ID	Size	Script Name
✓	Completed	1	1074	Order Details_cmdCompleteOrd
✓	Completed	2	320	CanApprovePurchases
☐	Not Started	3	419	DAvgWrapper
☐	Not Started	4	2051	Sales Reports Dialog_InitFilterIt
☐	Not Started	5	471	Purchases Subform for Purchase
☐	Not Started	6	547	Purchase Order Details_cmdSubm
☐	Not Started	7	740	Purchase Order Details_FRemove
☐	Not Started	8	1252	Monthly Sales Report_Report_Op
☐	Not Started	9	803	Order Details_cmdClearAddress_
☐	Not Started	10	404	Receiving Subform for Purchase
☐	Not Started	11	362	Edit
☐	Not Started	12	525	Purchases Subform for Purchase
☐	Not Started	13	784	LogError
☐	Not Started	14	1926	Sales Reports Dialog_lstSalesPeri
✓	Completed	15	2003	Order Subform for Order Details_
✓	Completed	16	2068	Order Details_cmdCreateInvoice_
☐	Not Started	17	462	DLookupNumberWrapper
✓	Completed	18	3568	SetStatus
☐	Not Started	19	1579	Receiving Subform for Purchase

Source Script (Editable): **Size:** 1,074

```
'Module: Order Details:Form_Order Details
Option Compare Database
Option Explicit


Sub SetDefaultShippingAddress()
    If IsNull(Me![Customer ID]) Then
        ClearShippingAddress
    Else

        Dim rsw As New RecordsetWrapper
        If rsw.OpenRecordset("Customers Extended", "[ID] = " & Me.Customer_ID)
```

Command (Editable):

Translate this Microsoft Access VBA code into FileMaker Pro script steps

Converted Script:

Source: Microsoft Access

Output Language: FileMaker Pro

Procedure/Function: Opening 1--1

Vendor: OpenAI

API Key: dcsi

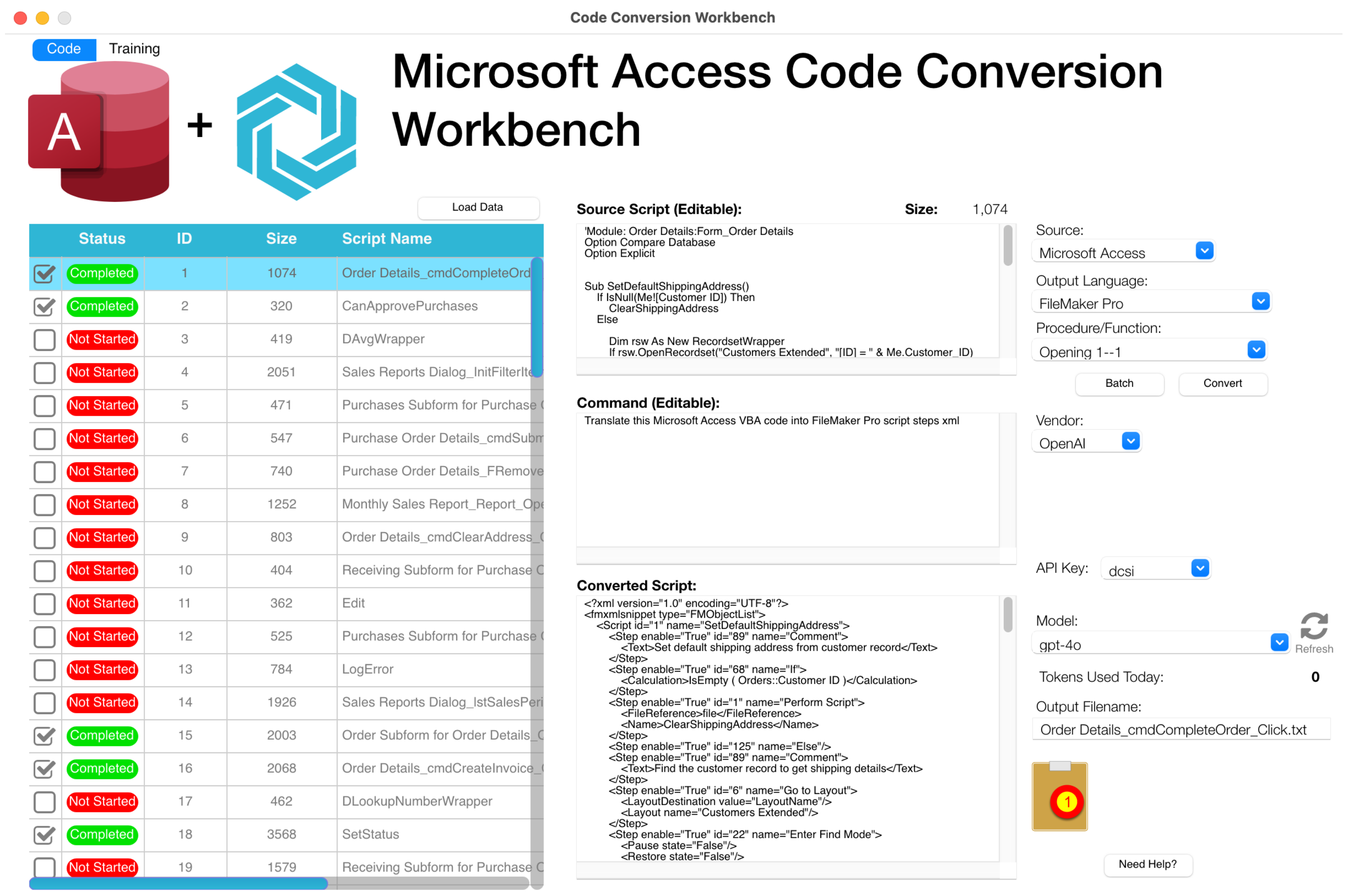
Model: gpt-4o

Tokens Used Today: 0

Output Filename: Order Details_cmdCompleteOrder_Click.txt

A text (.txt) file or XML (.xml) file can be imported into the Converted Script field from the desktop on macOS or Windows.

Copying Scripts into FileMaker Script Workspace via the Clipboard



Code Conversion Workbench

Microsoft Access Code Conversion Workbench

Status	ID	Size	Script Name
✓ Completed	1	1074	Order Details_cmdCompleteOrd
✓ Completed	2	320	CanApprovePurchases
Not Started	3	419	DAvgWrapper
Not Started	4	2051	Sales Reports Dialog_InitFilterIt
Not Started	5	471	Purchases Subform for Purchase
Not Started	6	547	Purchase Order Details_cmdSubm
Not Started	7	740	Purchase Order Details_FRemove
Not Started	8	1252	Monthly Sales Report_Report_Ope
Not Started	9	803	Order Details_cmdClearAddress_
Not Started	10	404	Receiving Subform for Purchase C
Not Started	11	362	Edit
Not Started	12	525	Purchases Subform for Purchase
Not Started	13	784	LogError
Not Started	14	1926	Sales Reports Dialog_lstSalesPeri
✓ Completed	15	2003	Order Subform for Order Details_
✓ Completed	16	2068	Order Details_cmdCreateInvoice_
Not Started	17	462	DLookupNumberWrapper
✓ Completed	18	3568	SetStatus
Not Started	19	1579	Receiving Subform for Purchase C

Source Script (Editable): **Size:** 1,074

```
'Module: Order Details:Form_Order Details
Option Compare Database
Option Explicit


Sub SetDefaultShippingAddress()
    If IsNull(Me![Customer ID]) Then
        ClearShippingAddress
    Else

        Dim rsw As New RecordsetWrapper
        If rsw.OpenRecordset("Customers Extended", "[ID] = " & Me.Customer_ID)
```

Command (Editable):

Translate this Microsoft Access VBA code into FileMaker Pro script steps xml

Converted Script:

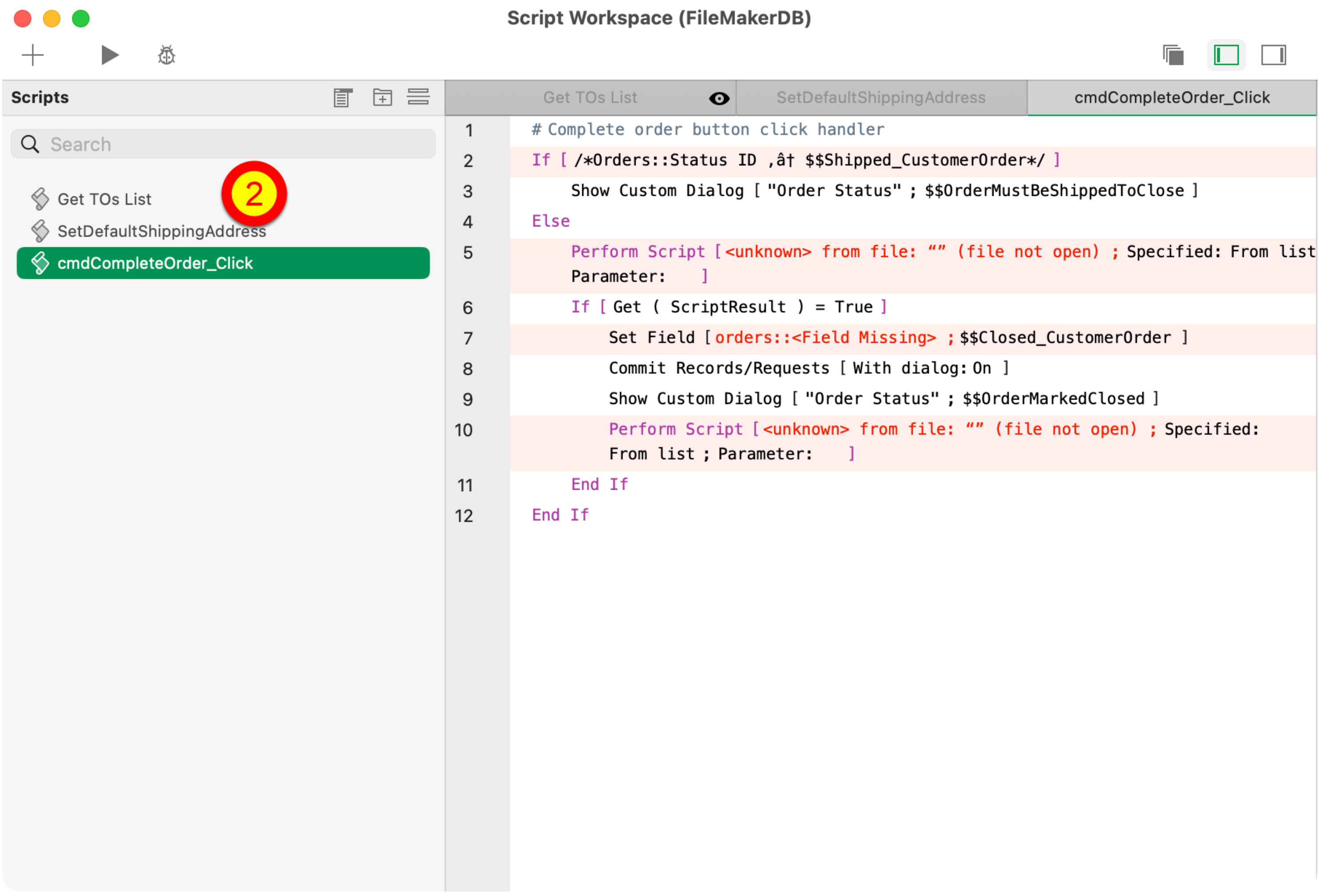
```
<?xml version="1.0" encoding="UTF-8"?>
<fmxmlsnippet type="FMObjectList">
    <Script id="1" name="SetDefaultShippingAddress">
        <Step enable="True" id="89" name="Comment">
            <Text>Set default shipping address from customer record</Text>
        </Step>
        <Step enable="True" id="68" name="If">
            <Calculation>IsEmpty ( Orders::Customer ID )</Calculation>
        </Step>
        <Step enable="True" id="1" name="Perform Script">
            <FileReference>file</FileReference>
            <Name>ClearShippingAddress</Name>
        </Step>
        <Step enable="True" id="125" name="Else"/>
        <Step enable="True" id="89" name="Comment">
            <Text>Find the customer record to get shipping details</Text>
        </Step>
        <Step enable="True" id="6" name="Go to Layout">
            <LayoutDestination value="LayoutName"/>
            <Layout name="Customers Extended"/>
        </Step>
        <Step enable="True" id="22" name="Enter Find Mode">
            <Pause state="False"/>
            <Restore state="False"/>
```

Source: Microsoft Access
Output Language: FileMaker Pro
Procedure/Function: Opening 1--1
Vendor: OpenAI
API Key: dcsi
Model: gpt-4o
Tokens Used Today: 0
Output Filename: Order Details_cmdCompleteOrder_Click.txt

1

Once the text or XML file has been imported into the Converted Script field, it can be copied and pasted into the FileMaker Pro Script Workspace with the (1) clipboard button.

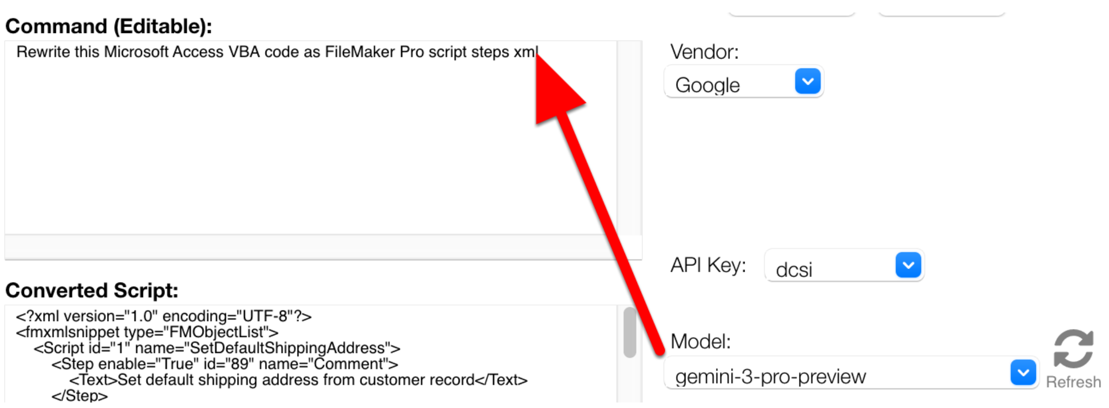
Pasting Script into Script Workspace



```
# Complete order button click handler
If [ /*Orders::Status ID ,ât $$Shipped_CustomerOrder*/ ]
    Show Custom Dialog [ "Order Status" ; $$OrderMustBeShippedToClose ]
Else
    Perform Script [ <unknown> from file: “” (file not open) ; Specified: From list
    Parameter:    ]
    If [ Get ( ScriptResult ) = True ]
        Set Field [ orders::<Field Missing> ; $$Closed_CustomerOrder ]
        Commit Records/Requests [ With dialog: On ]
        Show Custom Dialog [ "Order Status" ; $$OrderMarkedClosed ]
        Perform Script [ <unknown> from file: “” (file not open) ; Specified:
        From list ; Parameter:    ]
    End If
End If
```

(2) Click on an existing script in the list, then select Paste from the Edit menu. FileMaker Pro will read and import as much of the script XML as it can, and create the script.
Note: It is a good idea to also generate a text version of the script by the LLM and paste it as well so that both versions can be compared - in case some script steps were unreadable by the FileMaker database.

Google Gemini 3 Pro Generates Script XML for FileMaker Pro


Command (Editable):

Rewrite this Microsoft Access VBA code as FileMaker Pro script steps xml

Converted Script:

```
<?xml version="1.0" encoding="UTF-8"?>
<fmxmlsnippet type="FMObjectList">
  <Script id="1" name="SetDefaultShippingAddress">
    <Step enable="True" id="89" name="Comment">
      <Text>Set default shipping address from customer record</Text>
    </Step>
```

Vendor: Google

API Key: dcsi

Model: gemini-3-pro-preview

The Code Conversion Workbench automatically adds "xml" to the script conversion prompt for Google models when converting to FileMaker Pro, which is in addition to the Anthropic models. However it has been found that only the Google Gemini 3 Pro model generates script xml which can be pasted via the clipboard into FileMaker. The Gemini 3.0 Flash model doesn't usually generate XML for script steps which can be pasted into FileMaker.
The prompt is fully editable, so that removing the "xml" from the prompt and disabling the associated Training Record will also generate plain text which the Code Conversion Workbench also copies to the clipboard for pasting into FileMaker Pro.

Microsoft Access Code Conversion Workbench Demo

A + Microsoft Access Code Conversion Workbench Demo

If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the Microsoft Access Code Conversion Workbench will open in Demo mode. A previously converted set of sample scripts will be displayed in the grid instead of the scripts from your own Microsoft Access database. Clicking any of the scripts will display the text "This demo only includes a few scripts." in the Source Script field.
[At the present time, there aren't any Microsoft Access demo scripts.]

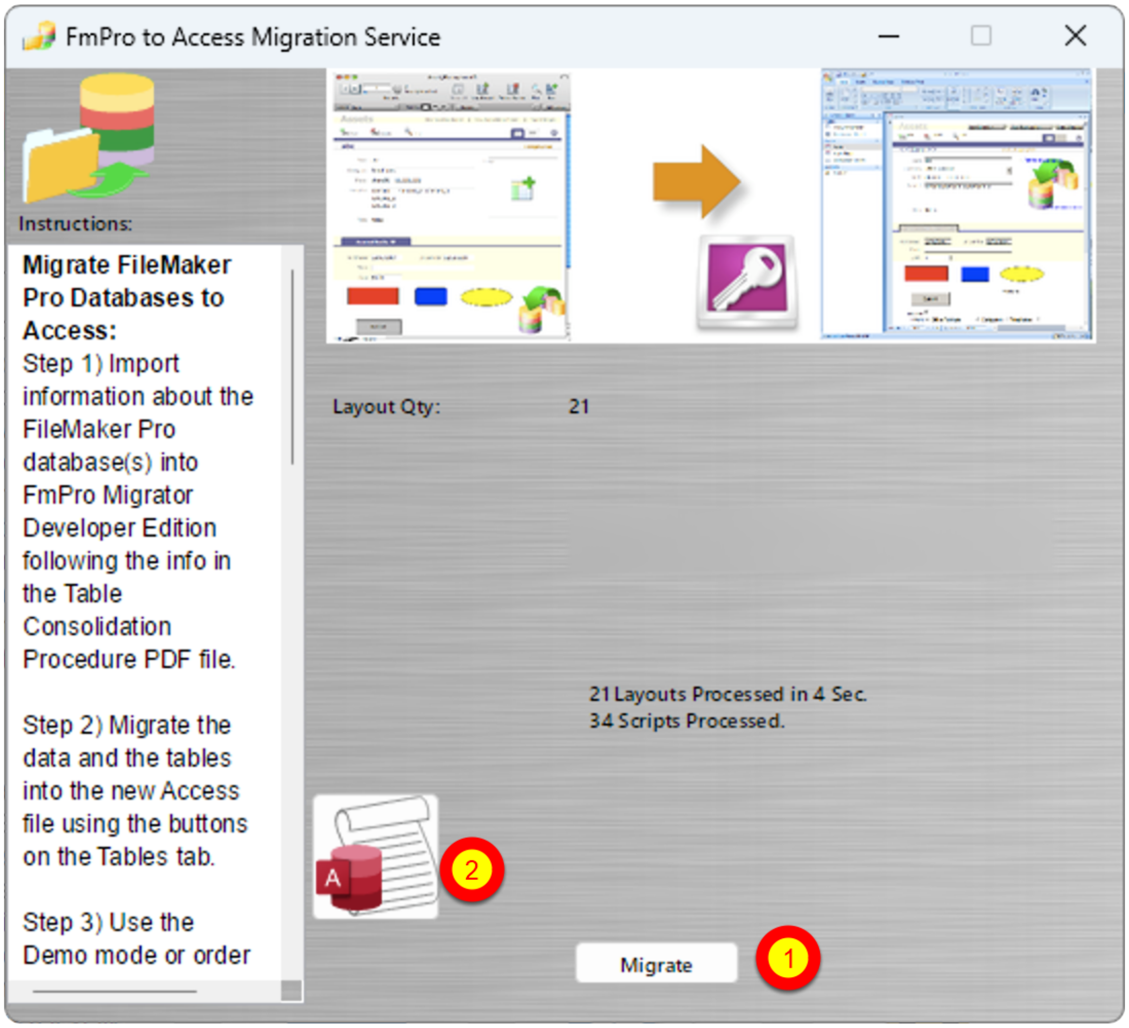
Using the FmPro Code Conversion Workbench - FmPro to Access Conversions

This chapter shows how to use the FmPro Code Conversion Workbench. Before performing the code conversion with the FmPro Code Conversion Workbench, you should have already imported the FileMaker Pro database into FmPro Migrator, including tables, relationships, value lists, layouts and scripts.

FmPro to Access Migration Button - GUI Tab

Click the FmPro to Access Migration button on the GUI tab of the Migration Process window of FmPro Migrator.

FmPro to Access Migration Window



Migrate FileMaker Pro Databases to Access:
Step 1) Import information about the FileMaker Pro database(s) into FmPro Migrator Developer Edition following the info in the Table Consolidation Procedure PDF file.

Step 2) Migrate the data and the tables into the new Access file using the buttons on the Tables tab.

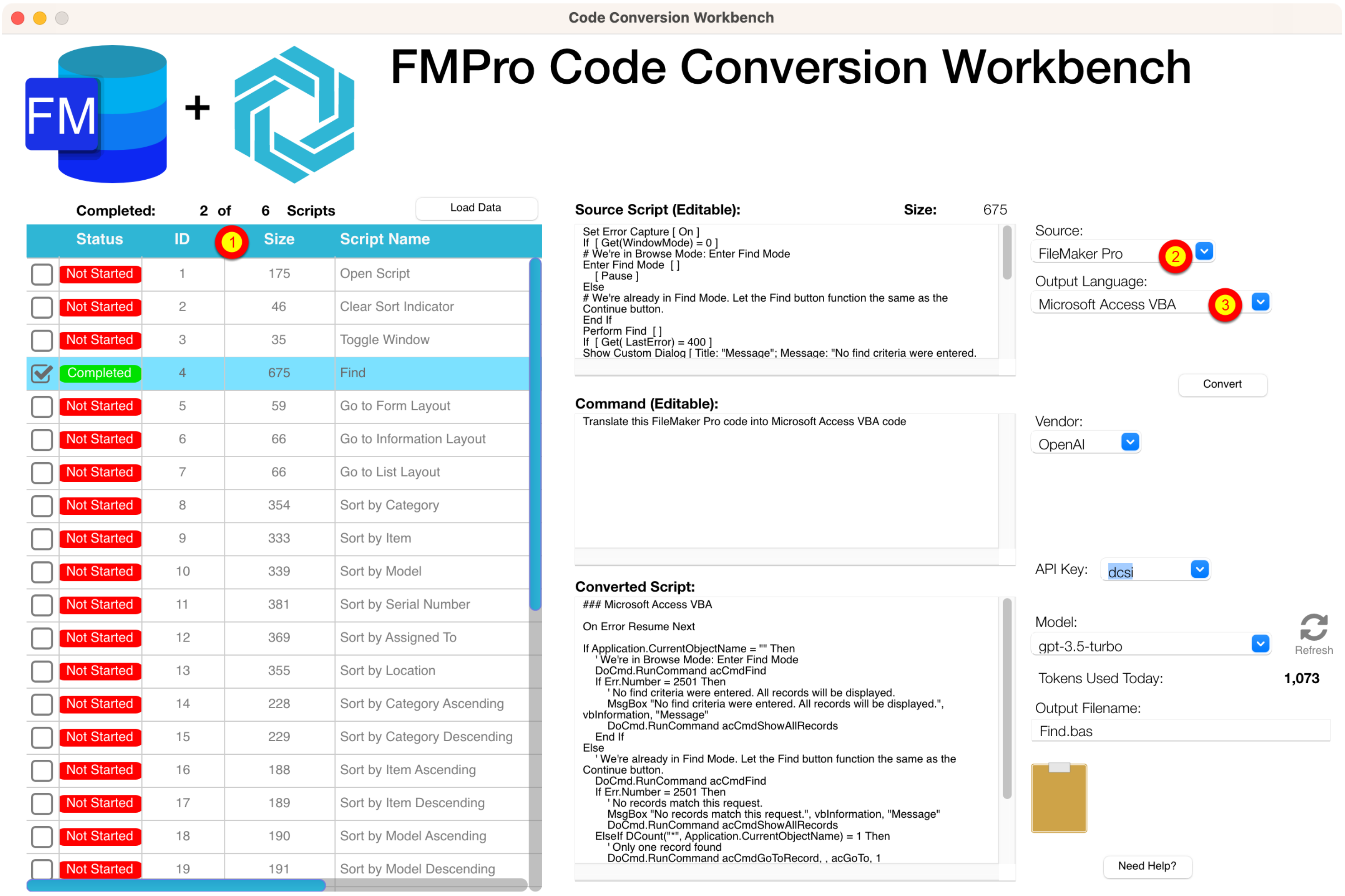
Step 3) Use the Demo mode or order

21 Layouts Processed in 4 Sec.
34 Scripts Processed.

Prior to reaching this step, you should have already imported the FileMaker Pro database into FmPro Migrator, (1) clicked the Migrate button to create the new Access database and run the _LoadAllFormsAndReports.bas script within the AccessDBFiles folder. After you have built the new Access database with all of the new forms/reports converted from the FileMaker Pro database you are ready (2) to click the Code Conversion Workbench button.

Note: Using the FmPro Code Conversion Workbench replaces the need to run the existing _FmProConvertedScriptsVBA.bas VBAscript.

FmPro Code Conversion Workbench Window



FMPro Code Conversion Workbench

Completed: 2 of 6 **Scripts**

	Status	ID	Size	Script Name
☐	Not Started	1	175	Open Script
☐	Not Started	2	46	Clear Sort Indicator
☐	Not Started	3	35	Toggle Window
✓	Completed	4	675	Find
☐	Not Started	5	59	Go to Form Layout
☐	Not Started	6	66	Go to Information Layout
☐	Not Started	7	66	Go to List Layout
☐	Not Started	8	354	Sort by Category
☐	Not Started	9	333	Sort by Item
☐	Not Started	10	339	Sort by Model
☐	Not Started	11	381	Sort by Serial Number
☐	Not Started	12	369	Sort by Assigned To
☐	Not Started	13	355	Sort by Location
☐	Not Started	14	228	Sort by Category Ascending
☐	Not Started	15	229	Sort by Category Descending
☐	Not Started	16	188	Sort by Item Ascending
☐	Not Started	17	189	Sort by Item Descending
☐	Not Started	18	190	Sort by Model Ascending
☐	Not Started	19	191	Sort by Model Descending

Source Script (Editable): **Size:** 675

```
Set Error Capture [ On ]
If  [ Get(WindowMode) = 0 ]
# We're in Browse Mode: Enter Find Mode
Enter Find Mode  [ ]
    [ Pause ]
Else
# We're already in Find Mode. Let the Find button function the same as the
Continue button.
End If
Perform Find  [ ]
If  [ Get( LastError) = 400 ]
Show Custom Dialog [ Title: "Message"; Message: "No find criteria were entered.
```

Command (Editable):

Translate this FileMaker Pro code into Microsoft Access VBA code

Converted Script:

```
### Microsoft Access VBA

On Error Resume Next

If Application.CurrentObjectName = "" Then
   ' We're in Browse Mode: Enter Find Mode
   DoCmd.RunCommand acCmdFind
   If Err.Number = 2501 Then
      ' No find criteria were entered. All records will be displayed.
      MsgBox "No find criteria were entered. All records will be displayed.",
vbInformation, "Message"
      DoCmd.RunCommand acCmdShowAllRecords
   End If
Else
   ' We're already in Find Mode. Let the Find button function the same as the
Continue button.
   DoCmd.RunCommand acCmdFind
   If Err.Number = 2501 Then
      ' No records match this request.
      MsgBox "No records match this request.", vbInformation, "Message"
      DoCmd.RunCommand acCmdShowAllRecords
   ElseIf DCount("*", Application.CurrentObjectName) = 1 Then
      ' Only one record found
      DoCmd.RunCommand acCmdGoToRecord, , acGoTo, 1
```

Source: FileMaker Pro (2)

Output Language: Microsoft Access VBA (3)

Vendor: OpenAI

API Key: dcsi

Model: gpt-3.5-turbo

Tokens Used Today: 1,073

Output Filename: Find.bas

By default, the FmPro Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) FileMaker Pro selected as the Source Database Type and (3) Microsoft Access VBA selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

FMPro Code Conversion Workbench Demo

If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the FmPro Code Conversion Workbench will open in Demo mode.
Apre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own FileMaker Pro database. Clicking any of the scripts will display the text "This demo only includes a few scripts." in the Source Script field.
[At the present time, there aren't any FileMaker Pro to Access demo scripts.]

Using the FmPro Code Conversion Workbench - FmPro to Servoy Conversions

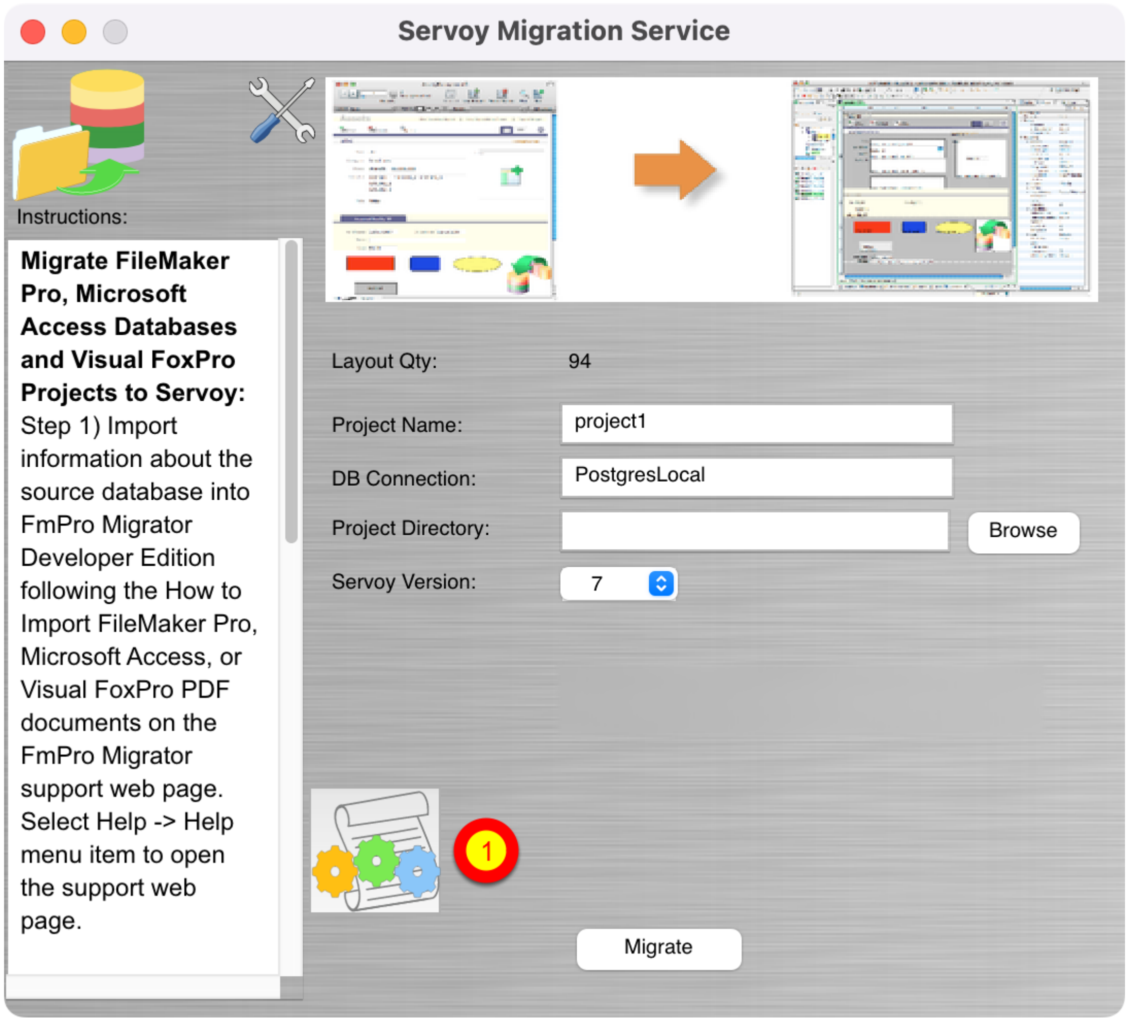
This chapter shows how to use the FmPro Code Conversion Workbench. Before performing the code conversion with the FmPro Code Conversion Workbench, you should have already imported the FileMaker Pro database into FmPro Migrator, including tables, relationships, value lists, layouts and scripts.

Just because this chapter of the manual is showing the conversion of FileMaker Pro scripts to Servoy, it is also possible to convert Visual FoxPro and Microsoft Access code to Servoy JavaScript.

Servoy Migration Button - GUI Tab

Click the Servoy Migration button on the GUI tab of the Migration Process window of FmPro Migrator.

Servoy Migration Window

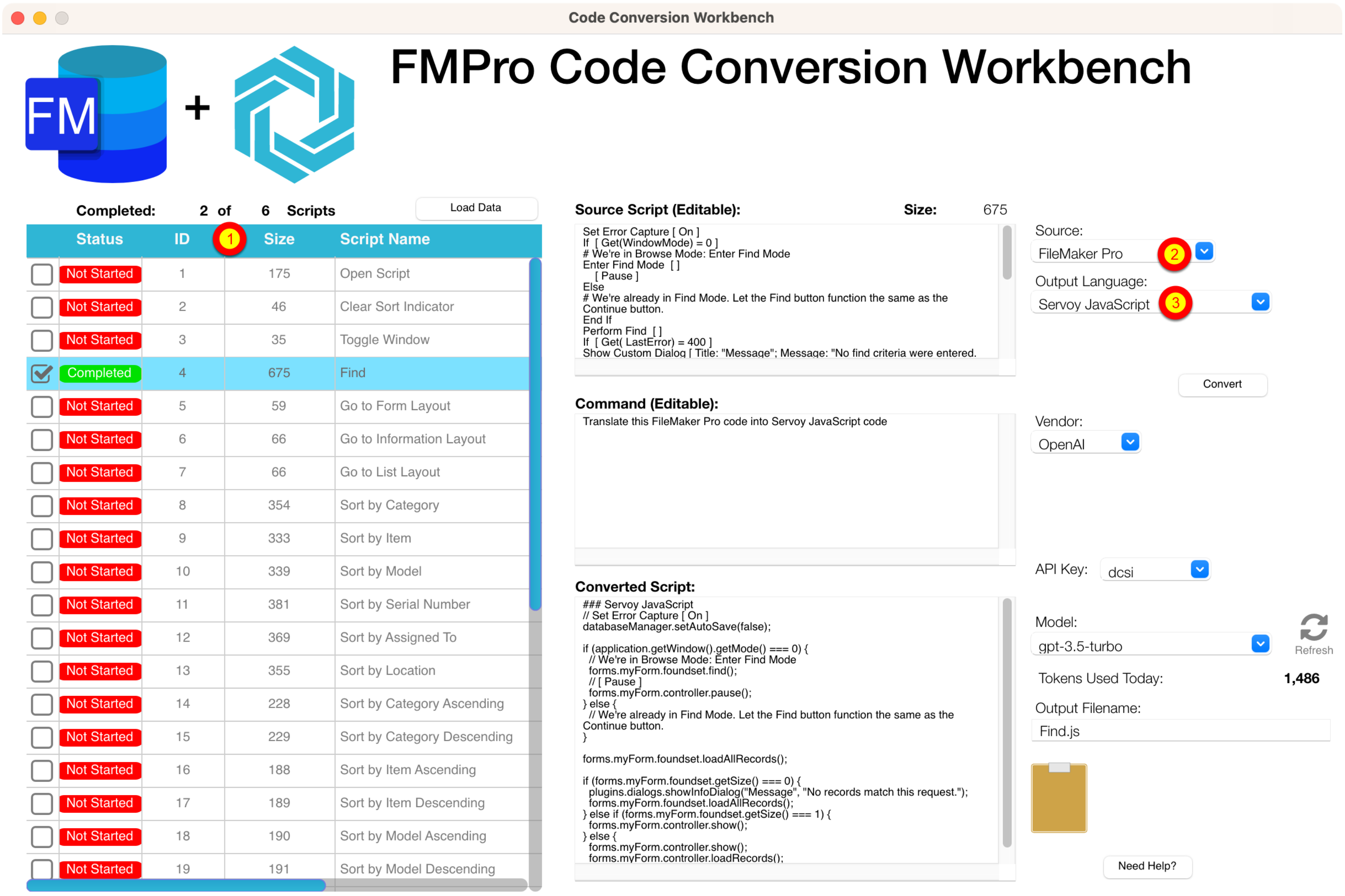


Migrate FileMaker Pro, Microsoft Access Databases and Visual FoxPro Projects to Servoy: Step 1) Import information about the source database into FmPro Migrator Developer Edition following the How to Import FileMaker Pro, Microsoft Access, or Visual FoxPro PDF documents on the FmPro Migrator support web page. Select Help -> Help menu item to open the support web page.

Layout Qty:	94
Project Name:	project1
DB Connection:	PostgresLocal
Project Directory:	
Servoy Version:	7

Prior to reaching this step, you should have already imported the FileMaker Pro database into FmPro Migrator and created your Servoy project. It is not necessary to click the Migrate button on this window, because you could have created the Servoy project manually. You are now ready (1) to click the Code Conversion Workbench button.

FmPro Code Conversion Workbench Window



Code Conversion Workbench

FMPro Code Conversion Workbench

Completed: 2 of 6 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	175	Open Script
☐	Not Started	2	46	Clear Sort Indicator
☐	Not Started	3	35	Toggle Window
✓	Completed	4	675	Find
☐	Not Started	5	59	Go to Form Layout
☐	Not Started	6	66	Go to Information Layout
☐	Not Started	7	66	Go to List Layout
☐	Not Started	8	354	Sort by Category
☐	Not Started	9	333	Sort by Item
☐	Not Started	10	339	Sort by Model
☐	Not Started	11	381	Sort by Serial Number
☐	Not Started	12	369	Sort by Assigned To
☐	Not Started	13	355	Sort by Location
☐	Not Started	14	228	Sort by Category Ascending
☐	Not Started	15	229	Sort by Category Descending
☐	Not Started	16	188	Sort by Item Ascending
☐	Not Started	17	189	Sort by Item Descending
☐	Not Started	18	190	Sort by Model Ascending
☐	Not Started	19	191	Sort by Model Descending

Source Script (Editable): Size: 675

```
Set Error Capture [ On ]
If  [ Get(WindowMode) = 0 ]
# We're in Browse Mode: Enter Find Mode
Enter Find Mode  [ ]
    [ Pause ]
Else
# We're already in Find Mode. Let the Find button function the same as the
Continue button.
End If
Perform Find  [ ]
If  [ Get( LastError) = 400 ]
Show Custom Dialog [ Title: "Message"; Message: "No find criteria were entered.
```

Command (Editable):

Translate this FileMaker Pro code into Servoy JavaScript code

Converted Script:

```
### Servoy JavaScript
// Set Error Capture [ On ]
databaseManager.setAutoSave(false);

if (application.getWindow().getMode() === 0) {
  // We're in Browse Mode: Enter Find Mode
  forms.myForm.foundset.find();
  // [ Pause ]
  forms.myForm.controller.pause();
} else {
  // We're already in Find Mode. Let the Find button function the same as the
Continue button.
}

forms.myForm.foundset.loadAllRecords();

if (forms.myForm.foundset.getSize() === 0) {
  plugins.dialogs.showInfoDialog("Message", "No records match this request.");
  forms.myForm.foundset.loadAllRecords();
} else if (forms.myForm.foundset.getSize() === 1) {
  forms.myForm.controller.show();
} else {
  forms.myForm.controller.show();
  forms.myForm.controller.loadRecords();
```

Source: FileMaker Pro

Output Language: Servoy JavaScript

Vendor: OpenAI

API Key: dcsi

Model: gpt-3.5-turbo

Tokens Used Today: 1,486

Output Filename: Find.js

By default, the FmPro Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) FileMaker Pro selected as the Source Database Type and (3) Servoy JavaScript selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

FMPro Code Conversion Workbench Demo

If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the FmPro Code Conversion Workbench will open in Demo mode.
Apre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own FileMaker Pro database. Clicking any of the scripts will display the text "This demo only includes a few scripts." in the Source Script field.
[At the present time, there aren't any FileMaker Pro to Servoy demo scripts.]

Using the FmPro Code Conversion Workbench - FmPro to PHP Conversions

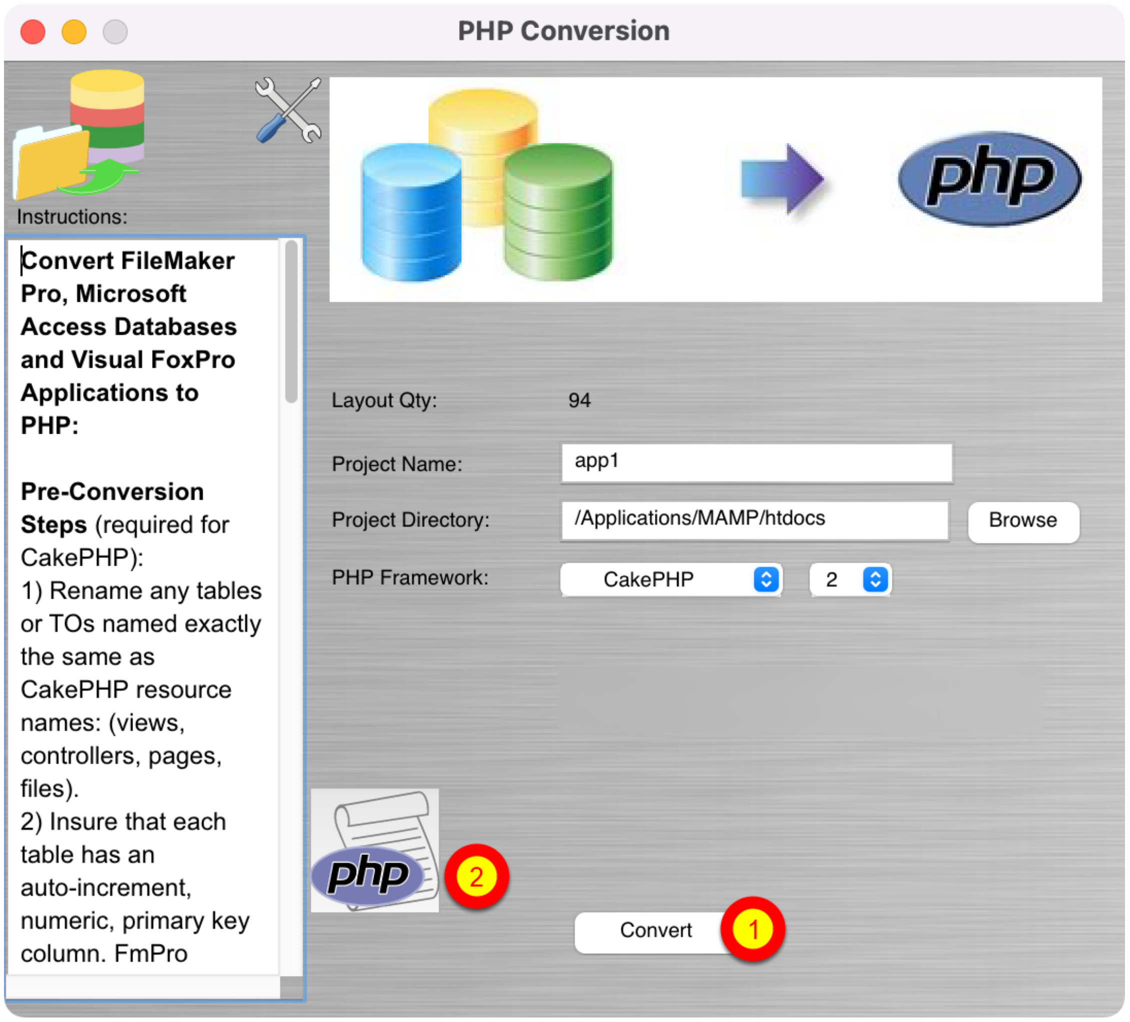
This chapter shows how to use the FmPro Code Conversion Workbench. Before performing the code conversion with the FmPro Code Conversion Workbench, you should have already imported the FileMaker Pro database into FmPro Migrator, including tables, relationships, value lists, layouts and scripts.

Just because this chapter of the manual is showing the conversion of FileMaker Pro scripts to PHP, it is also possible to convert Visual FoxPro and Microsoft Access code to PHP.

PHP Migration Button - GUI Tab

Click the PHP Conversion button on the GUI tab of the Migration Process window of FmPro Migrator.

PHP Conversion Window


PHP Conversion

Instructions:

Convert FileMaker Pro, Microsoft Access Databases and Visual FoxPro Applications to PHP:

Pre-Conversion Steps (required for CakePHP):
1) Rename any tables or TOs named exactly the same as CakePHP resource names: (views, controllers, pages, files).
2) Insure that each table has an auto-increment, numeric, primary key column. FmPro

Layout Qty: 94

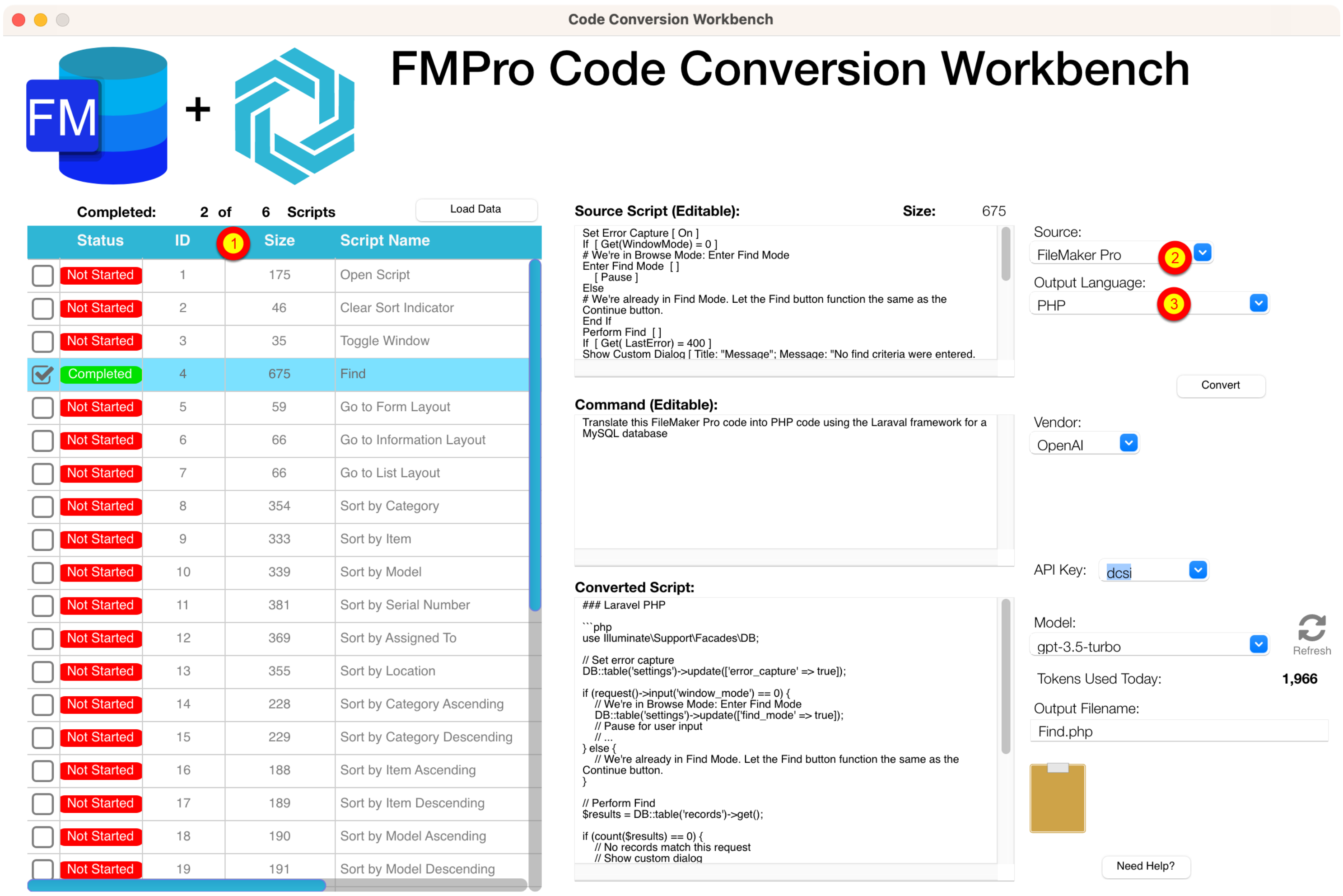
Project Name: app1

Project Directory: /Applications/MAMP/htdocs

PHP Framework: CakePHP 2

Prior to reaching this step, you should have already imported the FileMaker Pro database into FmPro Migrator, (1) created your PHP project with the Convert button. You are now ready to (2) click the Code Conversion Workbench button.

FmPro Code Conversion Workbench Window



Completed: 2 of 6 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	175	Open Script
☐	Not Started	2	46	Clear Sort Indicator
☐	Not Started	3	35	Toggle Window
✓	Completed	4	675	Find
☐	Not Started	5	59	Go to Form Layout
☐	Not Started	6	66	Go to Information Layout
☐	Not Started	7	66	Go to List Layout
☐	Not Started	8	354	Sort by Category
☐	Not Started	9	333	Sort by Item
☐	Not Started	10	339	Sort by Model
☐	Not Started	11	381	Sort by Serial Number
☐	Not Started	12	369	Sort by Assigned To
☐	Not Started	13	355	Sort by Location
☐	Not Started	14	228	Sort by Category Ascending
☐	Not Started	15	229	Sort by Category Descending
☐	Not Started	16	188	Sort by Item Ascending
☐	Not Started	17	189	Sort by Item Descending
☐	Not Started	18	190	Sort by Model Ascending
☐	Not Started	19	191	Sort by Model Descending

Source Script (Editable): **Size:** 675

```
Set Error Capture [ On ]
If  [ Get(WindowMode) = 0 ]
# We're in Browse Mode: Enter Find Mode
Enter Find Mode  [ ]
    [ Pause ]
Else
# We're already in Find Mode. Let the Find button function the same as the
Continue button.
End If
Perform Find  [ ]
If  [ Get( LastError) = 400 ]
Show Custom Dialog [ Title: "Message"; Message: "No find criteria were entered.
```

Command (Editable):

Translate this FileMaker Pro code into PHP code using the Laraval framework for a MySQL database

Converted Script:

~~~~
### Laravel PHP

```php
use Illuminate\Support\Facades\DB;

// Set error capture
DB::table('settings')->update(['error_capture' => true]);

if (request()->input('window_mode') == 0) {
    // We're in Browse Mode: Enter Find Mode
    DB::table('settings')->update(['find_mode' => true]);
    // Pause for user input
    // ...
} else {
    // We're already in Find Mode. Let the Find button function the same as the
Continue button.
}

// Perform Find
$results = DB::table('records')->get();

if (count($results) == 0) {
    // No records match this request
    // Show custom dialog
~~~~

Source: FileMaker Pro

Output Language: PHP

Vendor: OpenAI

API Key: dcsi

Model: gpt-3.5-turbo

Tokens Used Today: **1,966**

Output Filename: Find.php

By default, the FmPro Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) FileMaker Pro selected as the Source Database Type and (3) PHP selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

In this example, you can see that Laraval has been written into the conversion prompt and that MySQL is the output database. This text is fully modifiable, giving the developer the option to enter any PHP framework or database server.

FMPro Code Conversion Workbench Demo

If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the FmPro Code Conversion Workbench will open in Demo mode. A pre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own FileMaker Pro database. Clicking any of the scripts will display the text "This demo only includes a few scripts." in the Source Script field.

Using the FmPro Code Conversion Workbench - FmPro to LiveCode Conversions

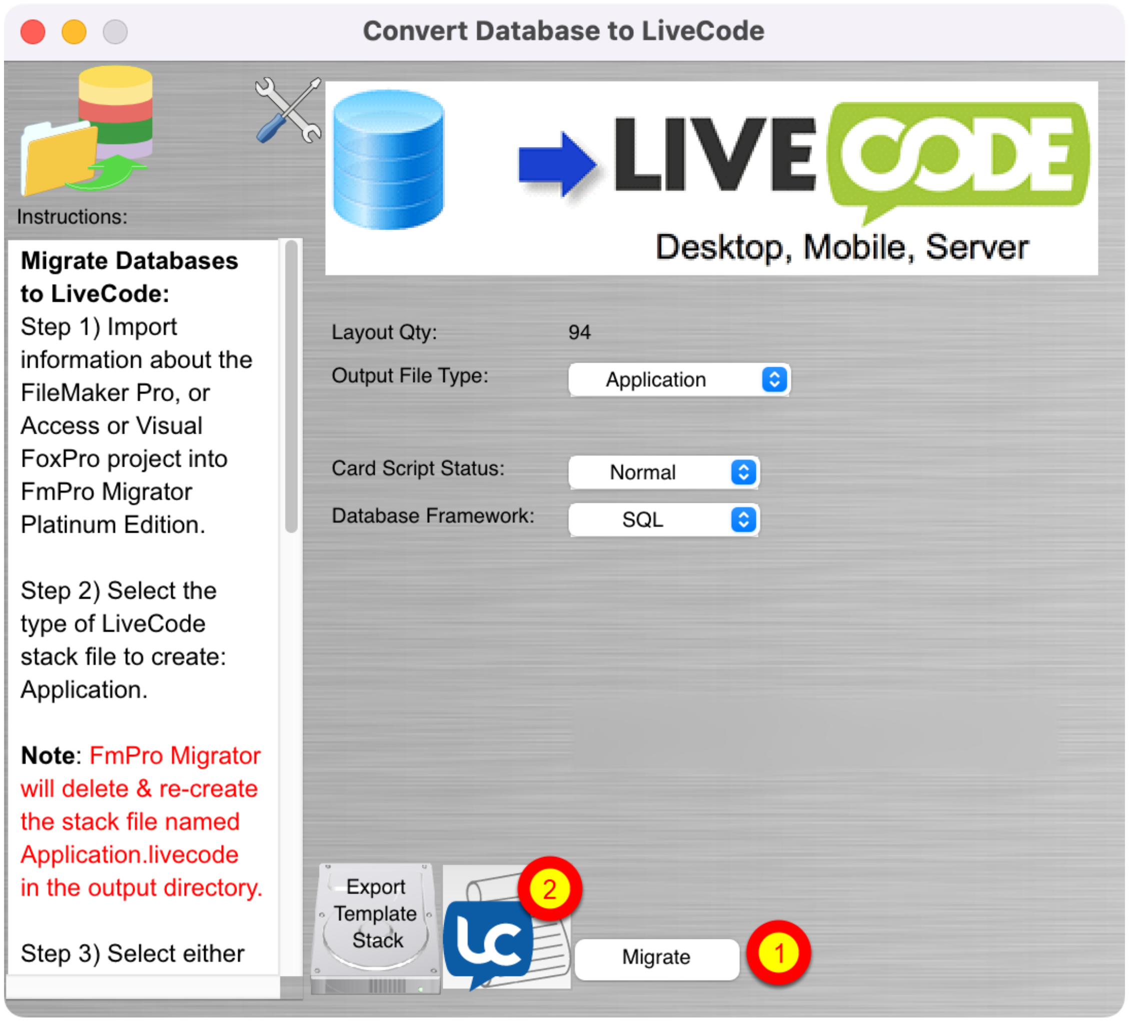
This chapter shows how to use the FmPro Code Conversion Workbench. Before performing the code conversion with the FmPro Code Conversion Workbench, you should have already imported the FileMaker Pro database into FmPro Migrator, including tables, relationships, value lists, layouts and scripts.

Just because this chapter of the manual is showing the conversion of FileMaker Pro scripts to LiveCode, it is also possible to convert Visual FoxPro and Microsoft Access code to LiveCode.

LiveCode Conversion Button - GUI Tab

Click the LiveCode Conversion button on the GUI tab of the Migration Process window of FmPro Migrator.

LiveCode Conversion Window


Convert Database to LiveCode

Instructions:

Migrate Databases to LiveCode:
Step 1) Import information about the FileMaker Pro, or Access or Visual FoxPro project into FmPro Migrator Platinum Edition.

Step 2) Select the type of LiveCode stack file to create: Application.

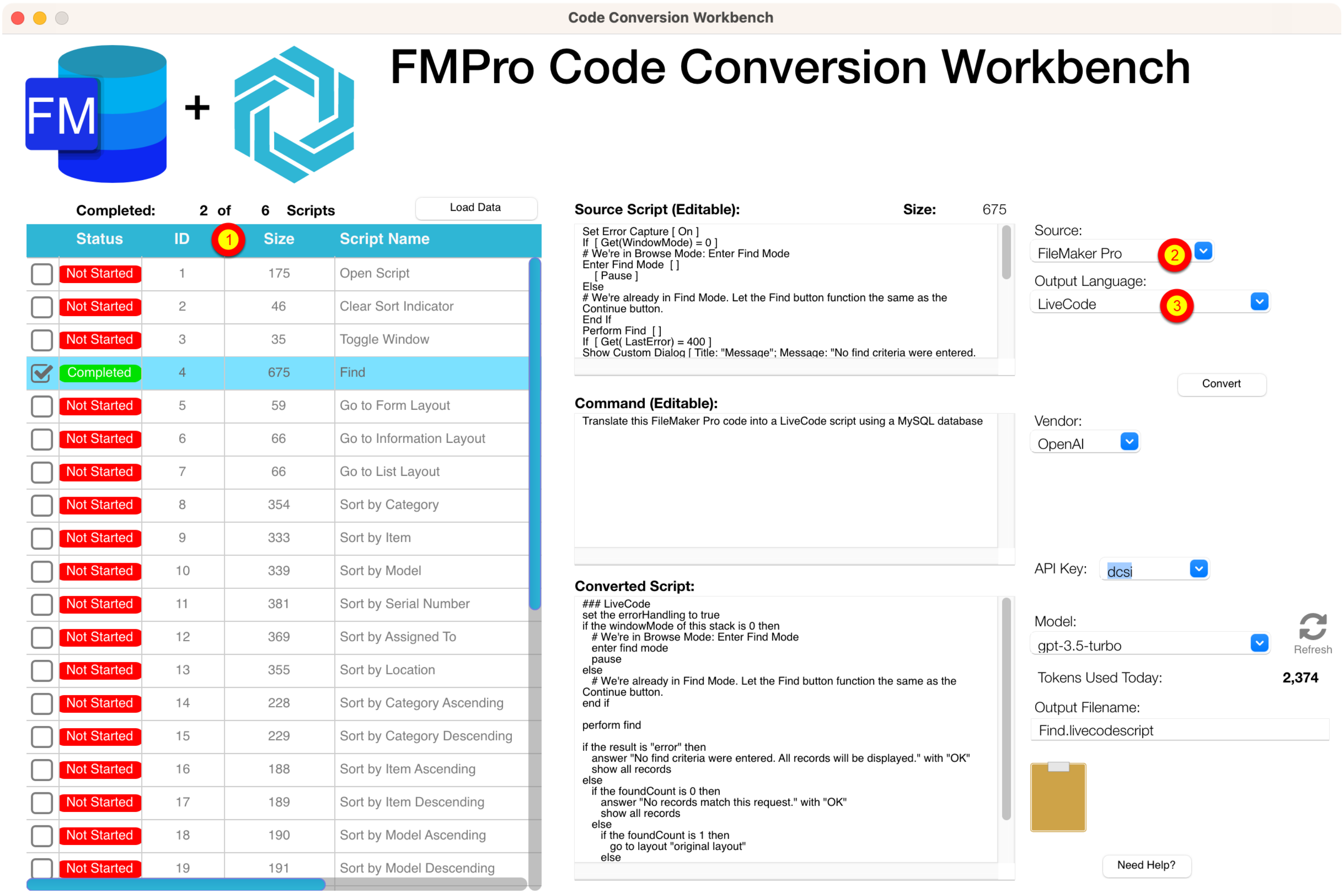
Note: FmPro Migrator will delete & re-create the stack file named Application.livecode in the output directory.

Step 3) Select either

Layout Qty: 94
Output File Type: Application
Card Script Status: Normal
Database Framework: SQL

Prior to reaching this step, you should have already imported the FileMaker Pro database into FmPro Migrator, (1) created your new LiveCode stack with the Migrate button. You are now ready to (2) click the Code Conversion Workbench button.

FmPro Code Conversion Workbench Window



FMPro Code Conversion Workbench

Completed: 2 of 6 Scripts

Status	ID	Size	Script Name
Not Started	1	175	Open Script
Not Started	2	46	Clear Sort Indicator
Not Started	3	35	Toggle Window
✓ Completed	4	675	Find
Not Started	5	59	Go to Form Layout
Not Started	6	66	Go to Information Layout
Not Started	7	66	Go to List Layout
Not Started	8	354	Sort by Category
Not Started	9	333	Sort by Item
Not Started	10	339	Sort by Model
Not Started	11	381	Sort by Serial Number
Not Started	12	369	Sort by Assigned To
Not Started	13	355	Sort by Location
Not Started	14	228	Sort by Category Ascending
Not Started	15	229	Sort by Category Descending
Not Started	16	188	Sort by Item Ascending
Not Started	17	189	Sort by Item Descending
Not Started	18	190	Sort by Model Ascending
Not Started	19	191	Sort by Model Descending

Source Script (Editable): **Size:** 675

```
Set Error Capture [ On ]
If  [ Get(WindowMode) = 0 ]
# We're in Browse Mode: Enter Find Mode
Enter Find Mode  [ ]
   [ Pause ]
Else
# We're already in Find Mode. Let the Find button function the same as the
Continue button.
End If
Perform Find  [ ]
If  [ Get( LastError) = 400 ]
Show Custom Dialog [ Title: "Message"; Message: "No find criteria were entered.
```

Command (Editable):

Translate this FileMaker Pro code into a LiveCode script using a MySQL database

Converted Script:

```
### LiveCode
set the errorHandling to true
if the windowMode of this stack is 0 then
  # We're in Browse Mode: Enter Find Mode
  enter find mode
  pause
else
  # We're already in Find Mode. Let the Find button function the same as the
Continue button.
end if

perform find

if the result is "error" then
  answer "No find criteria were entered. All records will be displayed." with "OK"
  show all records
else
  if the foundCount is 0 then
    answer "No records match this request." with "OK"
    show all records
  else
    if the foundCount is 1 then
      go to layout "original layout"
    else
```

Source: FileMaker Pro
Output Language: LiveCode
Vendor: OpenAI
API Key: dcsi
Model: gpt-3.5-turbo
Tokens Used Today: 2,374
Output Filename: Find.livecodescript

By default, the FmPro Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) FileMaker Pro selected as the Source Database Type and (3) LiveCode selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

In this example, you can see that LiveCode has been written into the conversion prompt and that MySQL is the output database. This text is fully modifiable giving the developer the option to enter any database server.

FMPro Code Conversion Workbench Demo

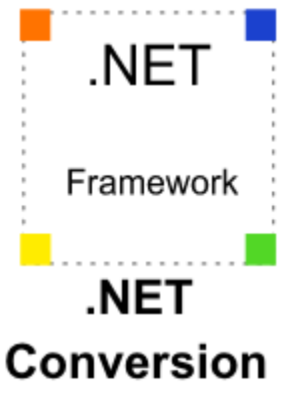
If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the FmPro Code Conversion Workbench will open in Demo mode.
Apre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own FileMaker Pro database. Clicking any of the scripts will display the text "This demo only includes a few scripts." in the Source Script field.
[At the present time, there aren't any FileMaker Pro to LiveCode demo scripts.]

Using the VFP Code Conversion Workbench - Visual FoxPro to .NET Conversions

This chapter shows how to use the VFP Code Conversion Workbench. Before performing the code conversion with the FmPro Code Conversion Workbench, you should have already imported the Visual FoxPro project into FmPro Migrator, including tables, relationships, value lists, forms/reports and scripts.

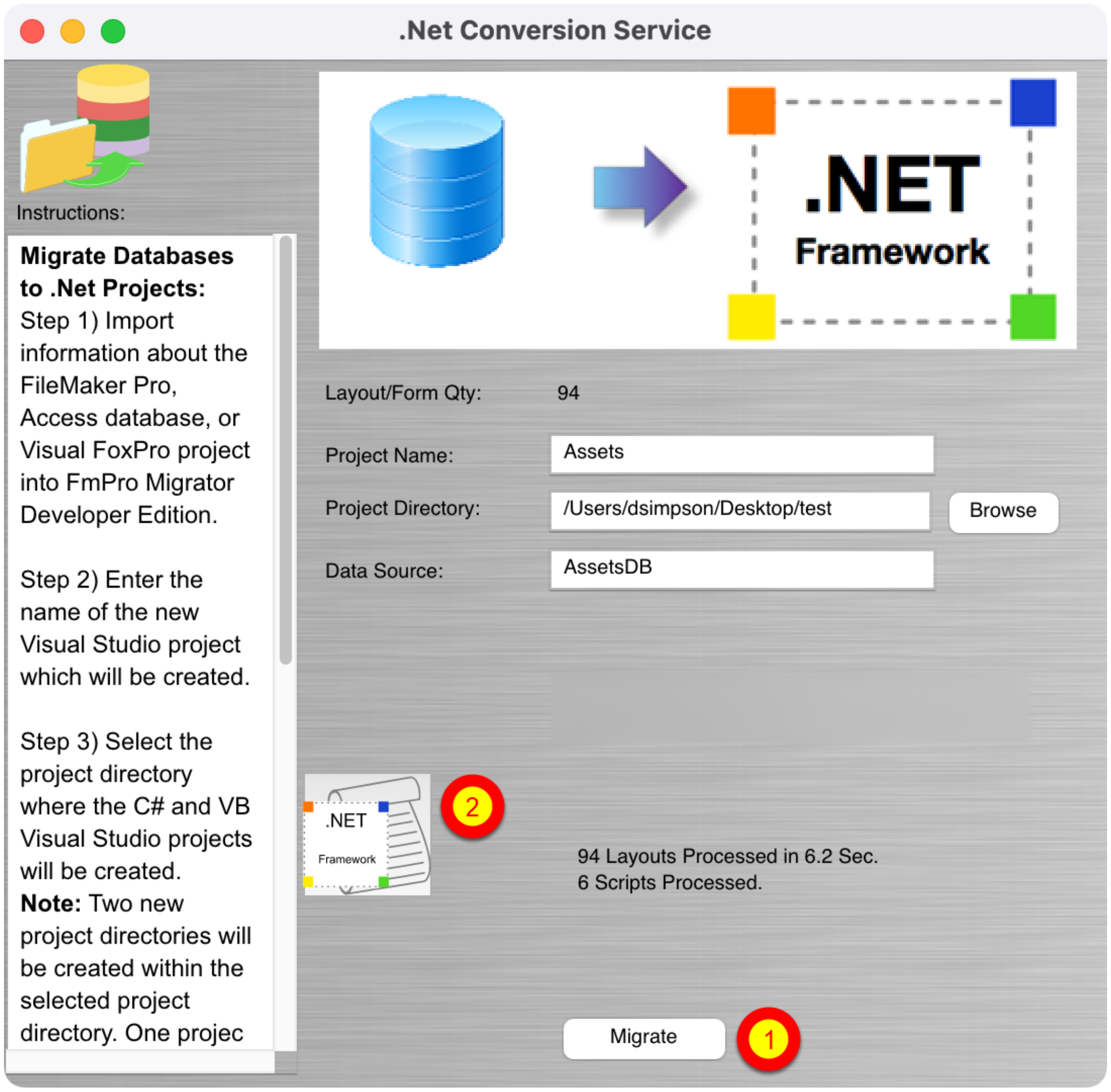
Just because this chapter of the manual is showing the conversion of Visual FoxPro scripts to .NET, it is also possible to convert FileMaker Pro and Microsoft Access code to .NET.

LiveCode Conversion Button - GUI Tab



Click the .NET Conversion button on the GUI tab of the Migration Process window of FmPro Migrator.

.NET Conversion Window



Migrate Databases to .Net Projects:
Step 1) Import information about the FileMaker Pro, Access database, or Visual FoxPro project into FmPro Migrator Developer Edition.

Step 2) Enter the name of the new Visual Studio project which will be created.

Step 3) Select the project directory where the C# and VB Visual Studio projects will be created.
Note: Two new project directories will be created within the selected project directory. One projec

Layout/Form Qty: 94

Project Name: Assets

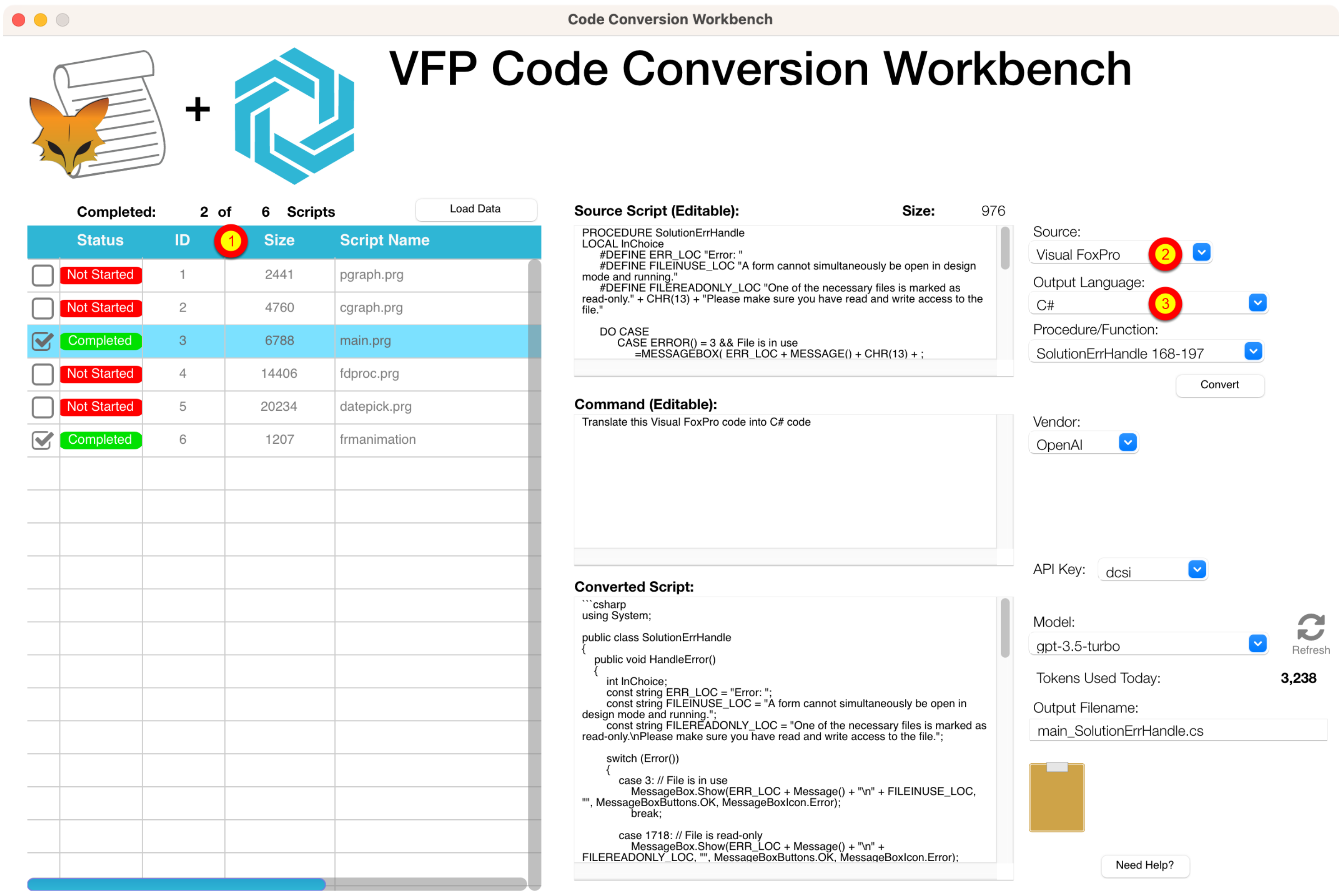
Project Directory: /Users/dsimpson/Desktop/test

Data Source: AssetsDB

94 Layouts Processed in 6.2 Sec.
6 Scripts Processed.

Prior to reaching this step, you should have already imported the Visual FoxPro project into FmPro Migrator using the Visual FoxPro Conversion button on the GUI tab, (1) created your new .NET project with the Migrate button. You are now ready to (2) click the Code Conversion Workbench button.

FmPro Code Conversion Workbench Window



Completed: 2 of 6 Scripts

	Status	ID	Size	Script Name
☐	Not Started	1	2441	pgraph.prg
☐	Not Started	2	4760	cgraph.prg
✓	Completed	3	6788	main.prg
☐	Not Started	4	14406	fdproc.prg
☐	Not Started	5	20234	datepick.prg
✓	Completed	6	1207	frmanimation

Source Script (Editable): **Size:** 976

```
PROCEDURE SolutionErrHandle
LOCAL lnChoice
    #DEFINE ERR_LOC "Error: "
    #DEFINE FILEINUSE_LOC "A form cannot simultaneously be open in design
mode and running."
    #DEFINE FILEREADONLY_LOC "One of the necessary files is marked as
read-only." + CHR(13) + "Please make sure you have read and write access to the
file."

    DO CASE
        CASE ERROR() = 3 && File is in use
            =MESSAGEBOX( ERR_LOC + MESSAGE() + CHR(13) + ;
```

Command (Editable):

Translate this Visual FoxPro code into C# code

Converted Script:

~~~~
```csharp
using System;

public class SolutionErrHandle
{
    public void HandleError()
    {
        int lnChoice;
        const string ERR_LOC = "Error: ";
        const string FILEINUSE_LOC = "A form cannot simultaneously be open in
design mode and running.";
        const string FILEREADONLY_LOC = "One of the necessary files is marked as
read-only.\nPlease make sure you have read and write access to the file.";

        switch (Error())
        {
            case 3: // File is in use
                MessageBox.Show(ERR_LOC + Message() + "\n" + FILEINUSE_LOC,
"", MessageBoxButtons.OK, MessageBoxIcon.Error);
                break;

            case 1718: // File is read-only
                MessageBox.Show(ERR_LOC + Message() + "\n" +
FILEREADONLY_LOC, "", MessageBoxButtons.OK, MessageBoxIcon.Error);
~~~~

Source: Visual FoxPro

Output Language: C#

Procedure/Function: SolutionErrHandle 168-197

Vendor: OpenAI

API Key: dcsi

Model: gpt-3.5-turbo

Tokens Used Today: 3,238

Output Filename: main_SolutionErrHandle.cs

By default, the VFP Code Conversion Workbench window opens with the (1) scripts listed in the grid, (2) Visual FoxPro selected as the Source Database Type and (3) C# selected as the Output Language.

Just because the options are selected a certain way when the window opens, doesn't mean you can't change them. You can easily change the default settings on this window for Output Language, Vendor, API Key Type, or AI Model - and the changes will be saved into the project or application preferences for FmPro Migrator.

VFP Code Conversion Workbench Demo

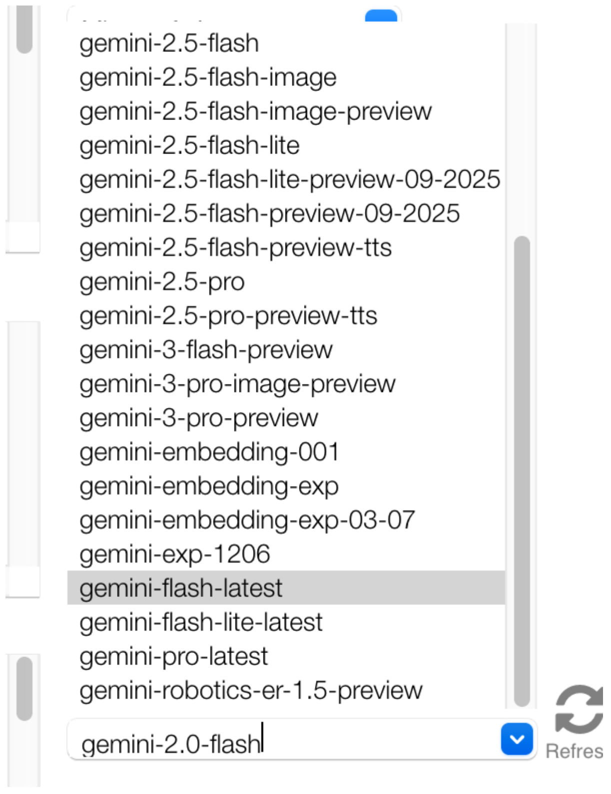
If you forgot to enter your license key or didn't have a license for the AI Accelerated version of FmPro Migrator, then the VFP Code Conversion Workbench will open in Demo mode. Apre-converted set of sample scripts will be displayed in the grid instead of the scripts from your own Visual FoxPro project.

Clicking the Convert button will place the pre-converted script into the Converted Script field.

Using Google Gemini Models

Google Gemini models are supported within the Code Code Conversion Workbench.

Google Models



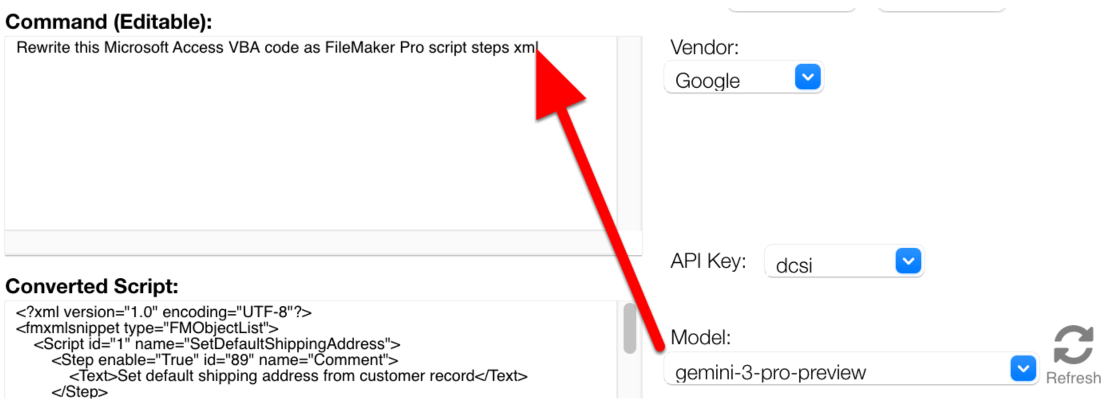
When selecting Google as the AI Vendor, the gemini-10-pro model will be automatically selected. Click the Refresh button next to the model menu to see any updates to the list of available models.

Based upon testing experience, the gemini-1.0-pro models are not very useful for automated code conversion.

At the present time, the gemini-3.0-pro, gemini 3.0-flash, models are the best Google models to use for code conversion tasks.

Note:
Holding down the shift key when clicking the Refresh button will display a much longer list of models. These additional models are not functional for code conversion tasks and will return an error if you attempt to use them.

Google Gemini 3 Pro Generates Script XML for FileMaker Pro



Rewrite this Microsoft Access VBA code as FileMaker Pro script steps xml

Vendor: Google

API Key: dcsi

Model: gemini-3-pro-preview

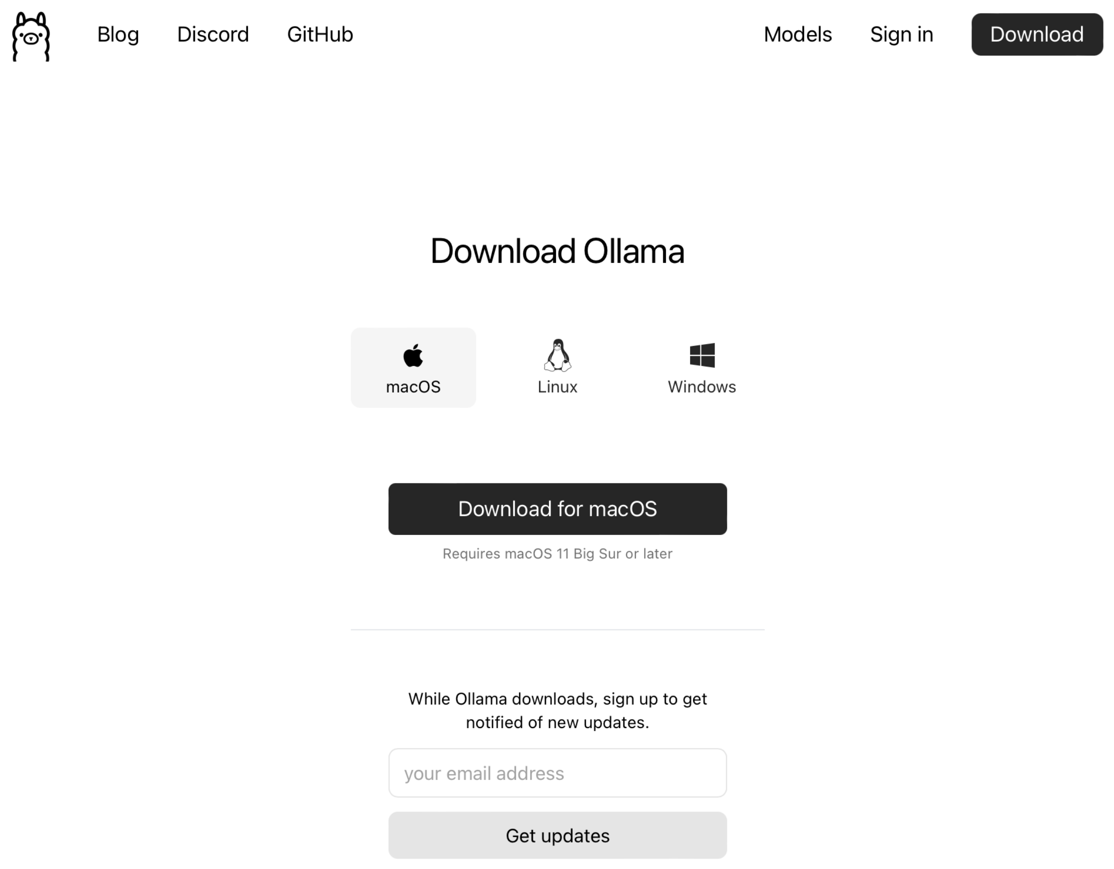
```
<?xml version="1.0" encoding="UTF-8"?>
<fmxmlsnippet type="FMObjectList">
   <Script id="1" name="SetDefaultShippingAddress">
      <Step enable="True" id="89" name="Comment">
         <Text>Set default shipping address from customer record</Text>
      </Step>
```

The Code Conversion Workbench automatically adds "xml" to the script conversion prompt for Google models when converting to FileMaker Pro, which is in addition to the Anthropic models. However it has been found that only the Google Gemini 3 Pro model generates script xml which can be pasted via the clipboard into FileMaker. The Gemini 3.0 Flash model doesn't usually generate XML for script steps which can be pasted into FileMaker.
The prompt is fully editable, so that removing the "xml" from the prompt and disabling the associated Training Record will also generate plain text which the Code Conversion Workbench also copies to the clipboard for pasting into FileMaker Pro.

Installing Ollama for Running LLMs Locally

Ollama is a software package which manages downloading, managing and serving open source LLMs on your local computer.

Installing Ollama



Download Ollama

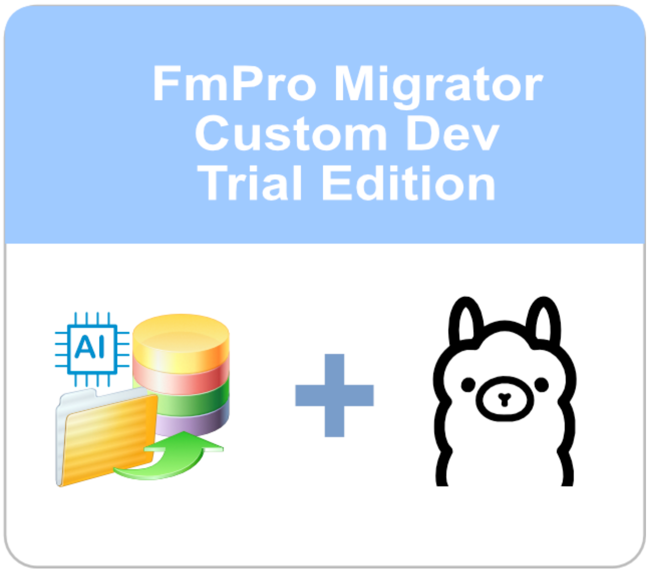
Requires macOS 11 Big Sur or later

While Ollama downloads, sign up to get notified of new updates.

Download and install Ollama from the Ollama.com website. Client software is available for macOS, Windows.
There are some considerations regarding the computer you choose for running local LLMs. Computers should have enough RAM to load the models into memory. Having one or more fast GPUs will also improve performance. Afast SSD will improve the performance of loading the model into memory.

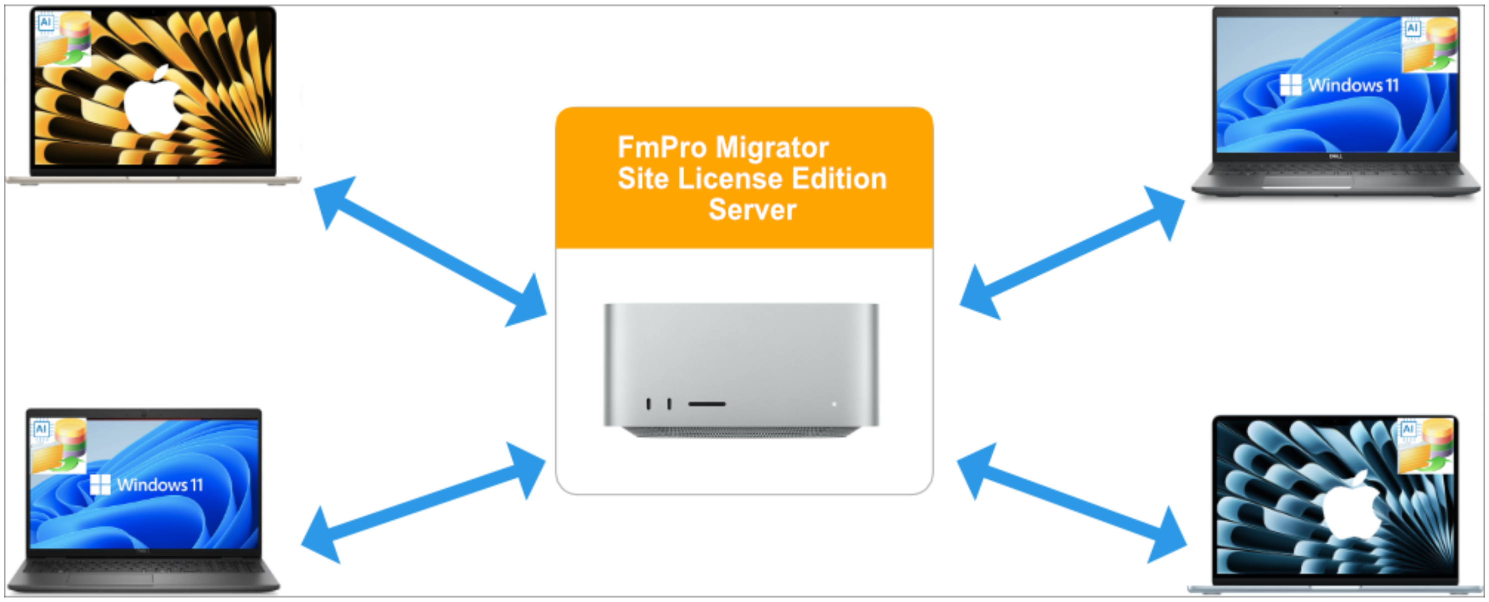
Available FmPro Migrator Editions for Using Ollama

Using Ollama to serve local LLMs is available with the FmPro Migrator Custom Development License and FmPro Migrator Site License Edition.



The FmPro Migrator Custom Dev Trial Edition also supports running LLMs locally with Ollama.

FmPro Migrator Site License Edition Server Diagram



FmPro Migrator Site License Edition is available with high performance server hardware optimized for running LLMs locally. FmPro Migrator software executes on each client machine, requesting code conversions from the server and retrieving the resulting code. This configuration ensures that the high-performance CPU/GPU hardware is accessible to an entire development team.

The server in this diagram is running Ollama software and Ollama is managing the installation, loading and unloading of LLMs into memory. The server is providing the CPU/GPU processing of the scripts and sending back the results to the client computers. The scripts being converted **ARE NOT** being stored on the server. Only the LLM files and a Ollama log file are being stored on the server.

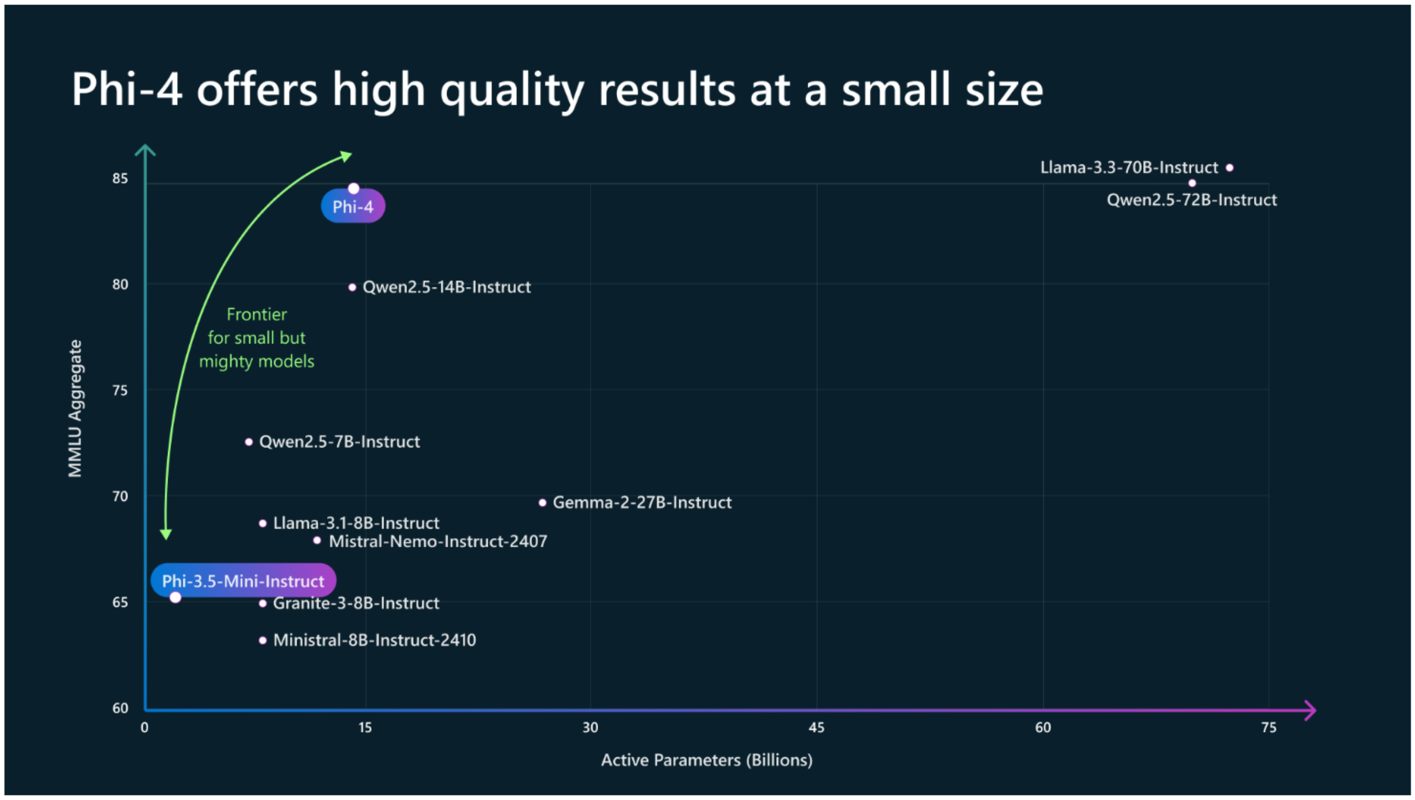
The Ollama log file is located at the following path:
/Users/<USER>/.ollama/logs/server.log

Use the cat command to read the contents of the log file.
```
cat ~/.ollama/logs/server.log
```

Use the tail command to observe the additions to the log file as it is running.
```
tail ~/.ollama/logs/server.log
```

Downloading Model - Microsoft Phi4 14.7B

Phi-4 is a 14B parameter, state-of-the-art open model built upon a blend of synthetic datasets, data from filtered public domain websites, and acquired academic books and Q&A datasets.



Microsoft's open source phi4 14B model has been distilled from the OpenAI GPT-4 model. Essentially, Phi-4 takes the knowledge from GPT-4 and compresses it into a smaller, more efficient model running locally instead of in the cloud.

Clicking the Models link at the top of the Ollama web page opens a list of available models.

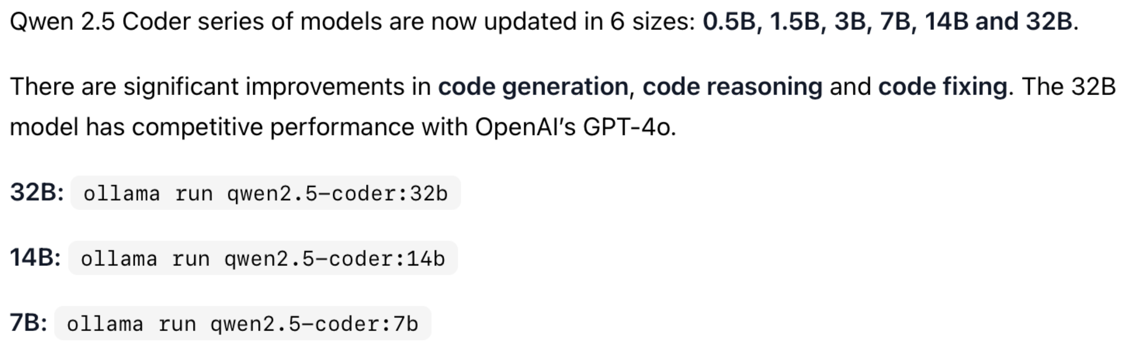
Type the following command into a command prompt/terminal window to download a local copy of a model. For this example, the phi4 model is being used.

```
ollama run phi4:14b
```

This command loads a 9.1GB phi4 file with 14.7B parameters. The phi4 model generally seems to provide results better than with OpenAI gpt-4o and Google Gemini 1.5-pro, and it works especially well at following the instructions provided in the training records.

If you have more VRAM or Unified Memory as used on Apple Silicon machines, you can load larger versions of this model than shown in this example.

Downloading Model - Qwen2.5-Coder



Qwen 2.5 Coder series of models are now updated in 6 sizes: **0.5B, 1.5B, 3B, 7B, 14B and 32B.**

There are significant improvements in **code generation**, **code reasoning** and **code fixing**. The 32B model has competitive performance with OpenAI's GPT-4o.

32B: `ollama run qwen2.5-coder:32b`

14B: `ollama run qwen2.5-coder:14b`

7B: `ollama run qwen2.5-coder:7b`

```
ollama run qwen2.5-coder:14b
```

This command loads a 9 GB qwen2.5-coder file with 14B parameters. The Qwen2.5-Coder models generally seem to provide results better than DeepSeek-R1, and they work especially well at following training records. In testing with the Code Conversion Workbench this model comes in #2 behind the Microsoft Phi4 model.

If you have more VRAM or Unified Memory as used on Apple Silicon machines, you can load larger versions of this model than shown in this example.

Downloading Model - DeepSeek-R1

DeepSeek’s first-generation reasoning models, achieving performance comparable to OpenAI-o1 across math, code, and reasoning tasks.

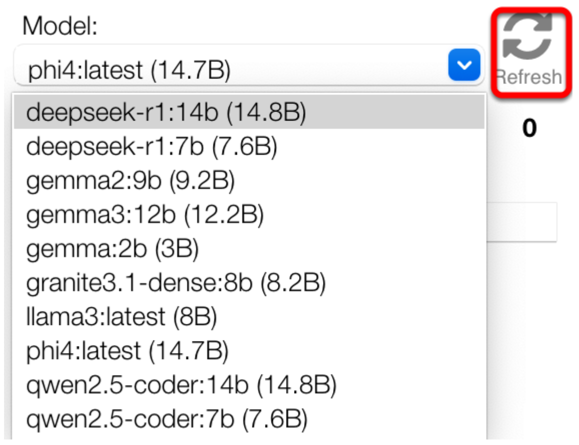
The following command:

```
ollama run deepseek-r1:7b
```

Loads the DeepSeek-R1 4.7GB model with 7B parameters. This model is a thinking model and generally **DOES NOT** closely follow the training records. DeepSeek-R1 has a lower output token count of 4k versus 16k for Microsoft's Phi4 model

If you have more VRAM or Unified Memory as used on Apple Silicon machines, you can load larger versions of this model than shown in this example. But overall, the Phi4 and Qwen2.5-Coder models have worked better overall - especially at following the training records.

Refreshing Models List

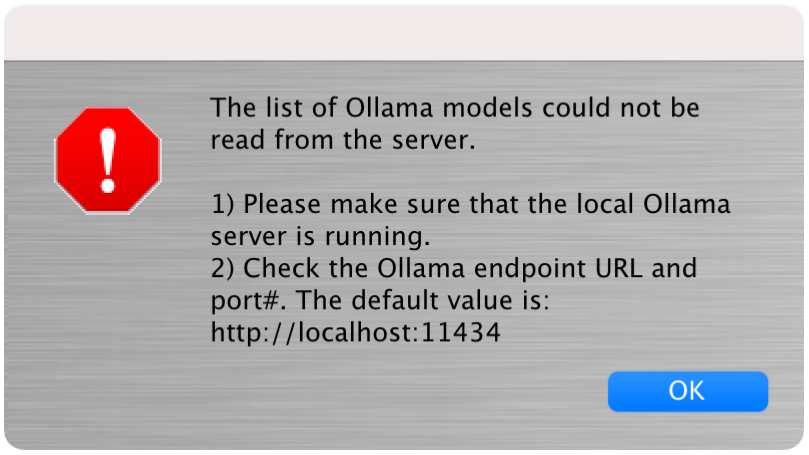


The first step to perform when using the Ollama models is to refresh the list of models. The default list represents the models built into FmPro Migrator during development, and won't reflect the actual models you have installed on your computer. Once you install models into Ollama, the list of available models will be stored in the FmPro Migrator preferences file on your local computer.

If you select an Ollama model which is not available, you will see an error like the following:

model 'codegemma:7b-code' not found, try pulling it first

Ollama Localhost API Endpoint Connectivity



The list of Ollama models could not be read from the server.

1) Please make sure that the local Ollama server is running.
2) Check the Ollama endpoint URL and port#. The default value is:
http://localhost:11434

Problem Symptoms:

1) The models Refresh button displays the above error message because the Ollama server cannot be located.
2) The red error result text "Script Not Converted - Too Long?" will immediately be displayed above the Converted Script field as soon as the Convert button is clicked.

FmPro Migrator needs to have local network access to the Ollama server endpoint when using Ollama machine learning models. The default URL for this local server is:

http://localhost:11434

Troubleshooting:
1) Make sure that the Ollama server is running. The server is started when launching the Ollama app, and a menu item will be displayed enabling control of the server. After the initial installation/configuration the app will quit as soon as it is launched - leaving the menu item.
2) You can go to the endpoint URL with a web browser and verify that you see the text:
Ollama is running

Note: When initially configured, "localhost" is the only address from which the Ollama server will respond.
Running the following command on macOS enables connections from other computers - but this command doesn't persist across restarts:
launchctl setenv OLLAMA_HOST "0.0.0.0"
The text of the next section can be placed into a file named ollama.plist which will persist these

settings across restarts. The plist file gets installed in the path:
/Library/LaunchDaemons/ollama.plist

On Windows, the various parameters are configured with environment variables according to this article:
https://docs.dify.ai/tutorials/model-configuration/ollama

In the plist info - the user name has been replaced with <USER> which needs changed to represent the username under which Ollama is installed.

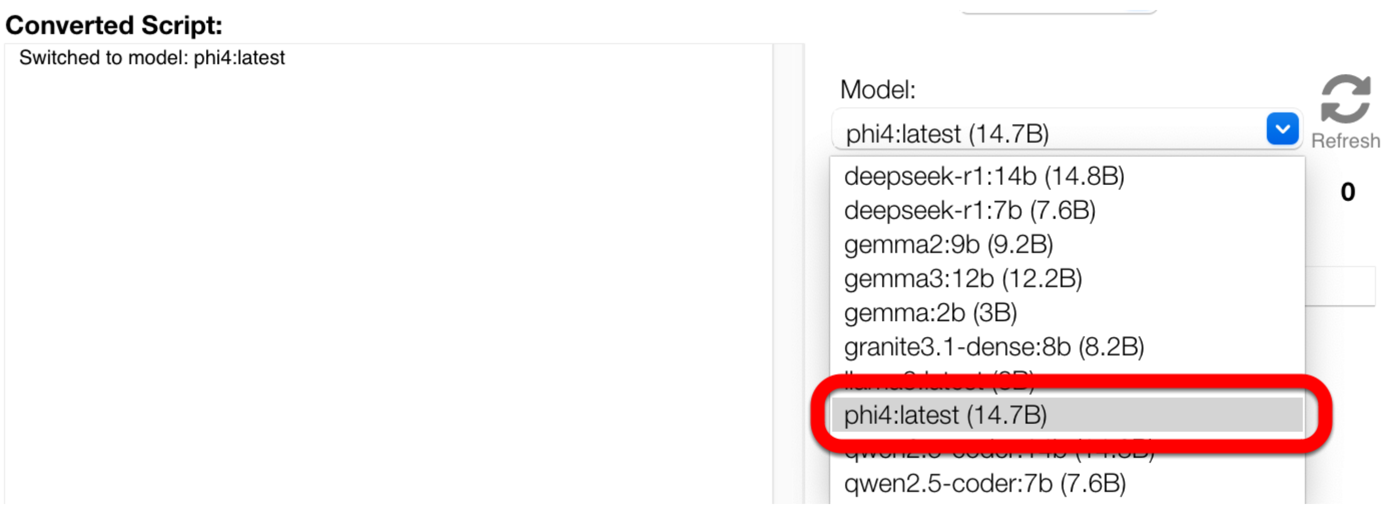
The localhost:11434 URL doesn't need to be entered into the Endpoint URL field, if it is left blank the default value shown above will be used automatically.

Ollama plist for macOS - Enables Multi-User Connections

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
   <key>Label</key>
   <string>ollama</string>
   <key>StandardOutPath</key>
   <string>/Users/<USER>/.ollama/launchd.stdout.log</string>
   <key>StandardErrorPath</key>
   <string>/Users/<USER>/.ollama/launchd.stderr.log</string>
   <key>EnvironmentVariables</key>
   <dict>
      <key>OLLAMA_HOST</key>
      <string>0.0.0.0:11434</string>
   </dict>
   <key>ProgramArguments</key>
   <array>
     <string>/usr/local/bin/ollama</string>
     <string>serve</string>
   </array>
   <key>UserName</key>
   <string><USER></string>
   <key>GroupName</key>
   <string>staff</string>
   <key>ExitTimeOut</key>
   <integer>30</integer>
   <key>Disabled</key>
```

```
    <false />
    <key>KeepAlive</key>
    <true />
  </dict>
</plist>
```

Selecting an Ollama Model


Converted Script:

Switched to model: phi4:latest

Ollama provides a way to load models into memory with the command:

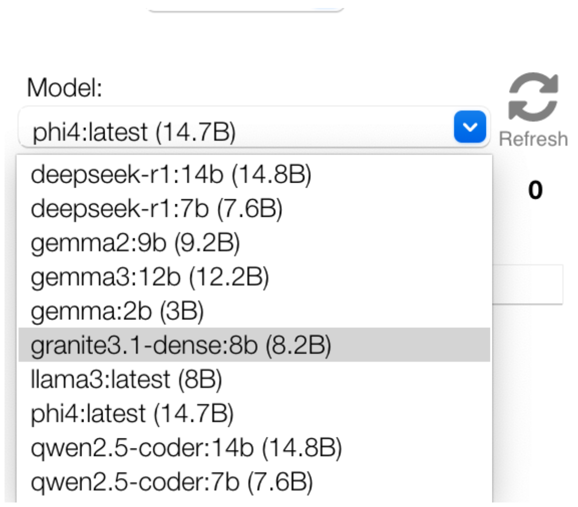
ollama run <model name>

The Code Conversion Workbench GUI automates this task. Just select a model from the list and in a few seconds it will be loaded. The message:

Switched to model: <model name>

will be displayed in the Converted script field.

Ollama Models to Avoid



The granite3.1-dense:8b model is one of the default models built into the Model menu during development.

This model was used during testing and has not been found to be suitable for code conversion tasks - so please don't install it. Feel free to install other models suitable for your computer hardware if you like.

Some of the models to avoid include:
granite3.1
gemma2
gemma:2b

Some models which have worked well for code conversion with training records include:
phi4
qwen2.5-coder
llama3

Note: Each Ollama supported model name consists of 2 pieces of information.
The left portion of the menu shows the name of the model as it has been installed locally.
The right portion of the menu (9B, 3B, 8B) shows the number of parameters used to train the model. Generally higher numbers produce better code but take longer to run. This way you can easily determine the best mix of size and performance you want to use for your conversion project.

Conclusions

- DeepSeek-R1 14B - Doesn't run on modest hardware when running locally.
- **`[Myth Busted - about better efficiency]`**

- DeepSeek-R1 7B - Fails to follow the training records.

- Microsoft's Phi4 - 14B - Follows all of the requested training records.
- **`Best Overall Model`**

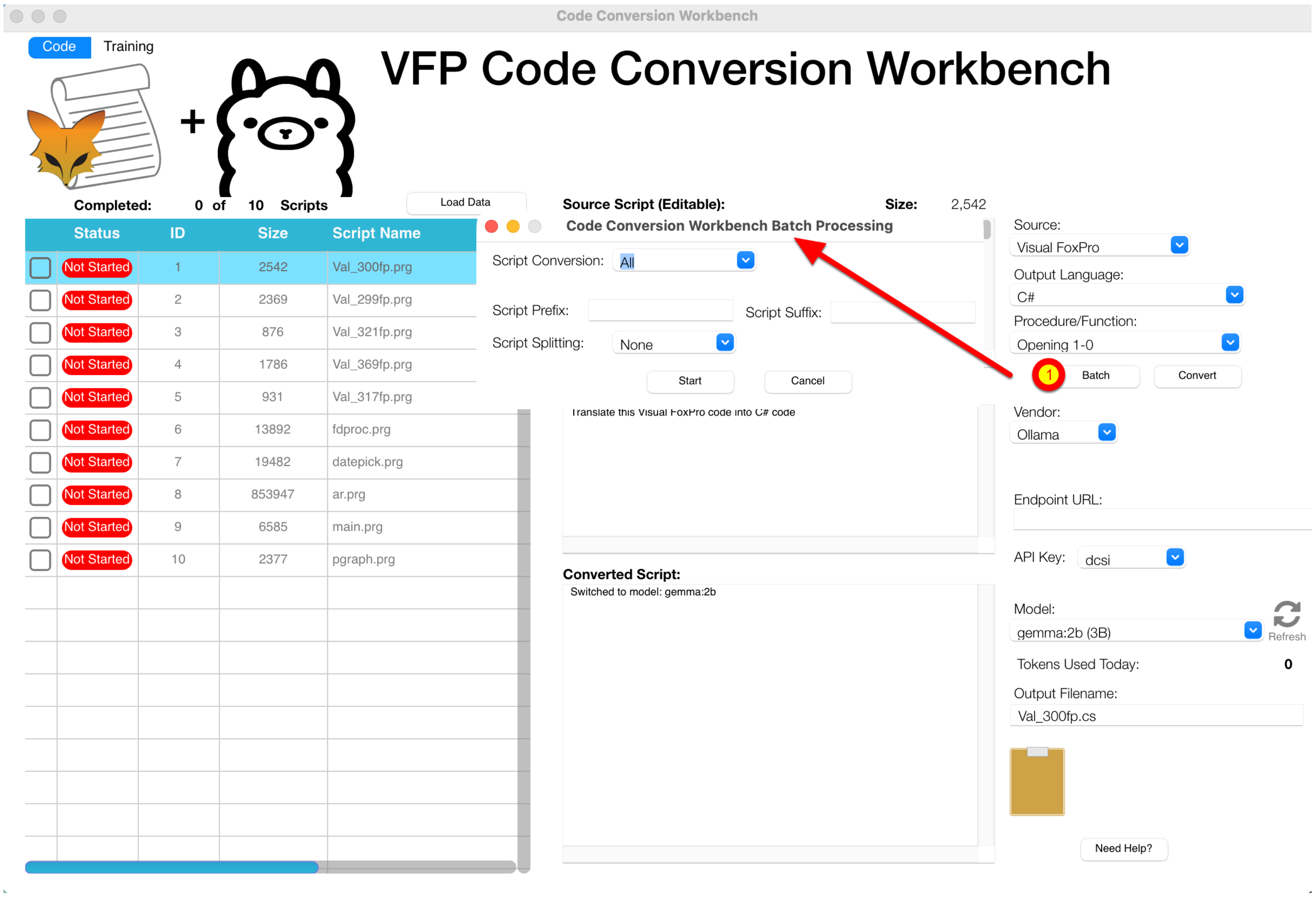
- Qwen 2.5-Coder 7B & 14B - Are also very good at following the training records.

- Qwen 2.5-Coder 14B - Is slightly better at following training records compared with the 7B version of the model - in regards to adding () around comparisons.

Batch Processing of Scripts

When using Ollama to run local LLMs, the Code Conversion Workbench enables the batch processing feature.

Batch Processing Button



Completed: 0 of 10 Scripts

Status	ID	Size	Script Name
Not Started	1	2542	Val_300fp.prg
Not Started	2	2369	Val_299fp.prg
Not Started	3	876	Val_321fp.prg
Not Started	4	1786	Val_369fp.prg
Not Started	5	931	Val_317fp.prg
Not Started	6	13892	fdproc.prg
Not Started	7	19482	datepick.prg
Not Started	8	853947	ar.prg
Not Started	9	6585	main.prg
Not Started	10	2377	pgraph.prg

Source Script (Editable): Size: 2,542

Code Conversion Workbench Batch Processing

Script Conversion: All

Script Prefix: Script Suffix:

Script Splitting: None

Translate this Visual FoxPro code into C# code

Converted Script:

Switched to model: gemma:2b

Source: Visual FoxPro

Output Language: C#

Procedure/Function: Opening 1-0

Vendor: Ollama

Endpoint URL:

API Key: dcsi

Model: gemma:2b (3B)

Tokens Used Today: 0

Output Filename: Val_300fp.cs

Clicking the Batch button opens the Code Conversion Workbench Batch Processing window. Batch processing of scripts includes the following features:

All - Convert all scripts

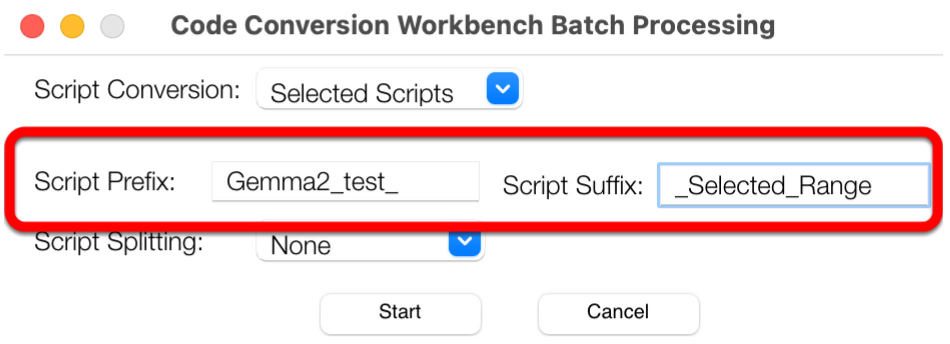
Selected Scripts - Convert all of the scripts selected in the grid. Shift - select to select a contiguous range of scripts, Control Select to select addition non-contiguous scripts in the list.

None - Converts none of the scripts

Completed - Converts the scripts in the grid which have a status of Completed.

Not Started - Converts the scripts in the grid which have a status of Not Started.

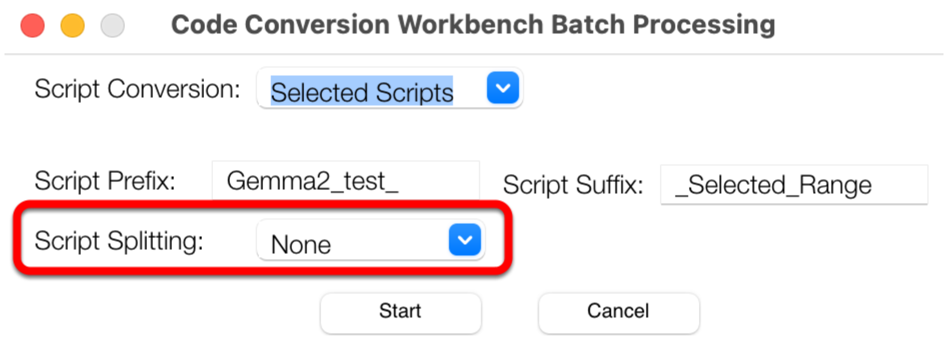
Script Prefix/Script Suffix



Script Conversion: Selected Scripts
Script Prefix: Gemma2_test_
Script Suffix: _Selected_Range
Script Splitting: None

Each converted script may be named with script prefix or suffix text by entering values into these two fields of the Batch Processing window.

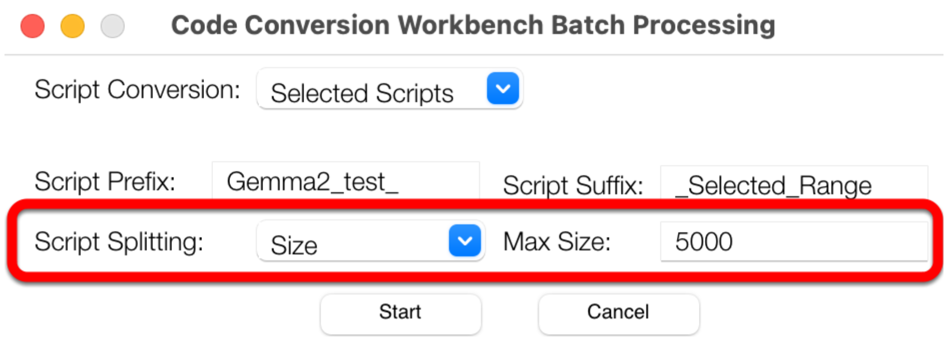
Script Splitting Option



Script Conversion: Selected Scripts
Script Prefix: Gemma2_test_
Script Suffix: _Selected_Range
Script Splitting: None

The Script Splitting menu provides for splitting of individual scripts in order to reduce the amount of text sent to the LLM to be manageable for the context size of the LLM.
These splitting options include:
None - Scripts won't be split and will be sent in their entirety to the LLM for processing.
Procedure - [Visual FoxPro only] Scripts will be split by procedure or function and sent individually to the LLM.
Size - Scripts may also be split by size. This is the least desirable splitting method, because the LLM won't have the entire script available at one time for it to process. But sometimes this is necessary in order to match the context size of the model you have chosen.

Script Splitting by Size



Script Conversion: Selected Scripts

Script Prefix: Gemma2_test_

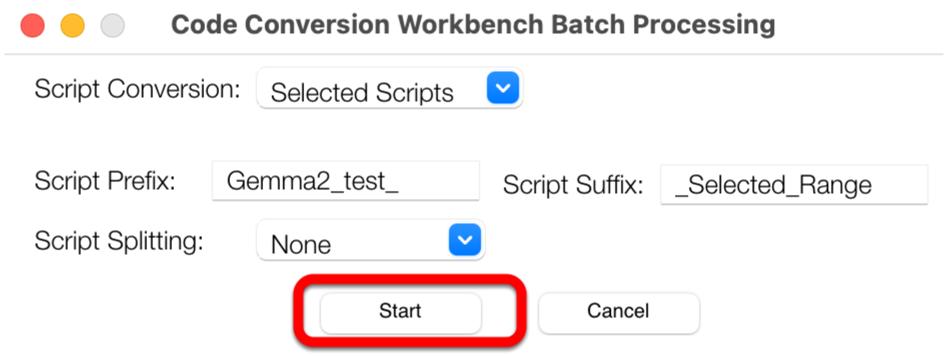
Script Suffix: _Selected_Range

Script Splitting: Size

Max Size: 5000

When the Size option has been selected from the Script Splitting menu, the Max Size field will be visible. In this example, 5000 characters has been entered so the scripts will be split into 5000 character chunks for processing. The resulting output files will include the splitting info (1_10, 2_10, 3_10) etc for example if the file was split into 10 pieces.

Start Batch Processing



Script Conversion: Selected Scripts

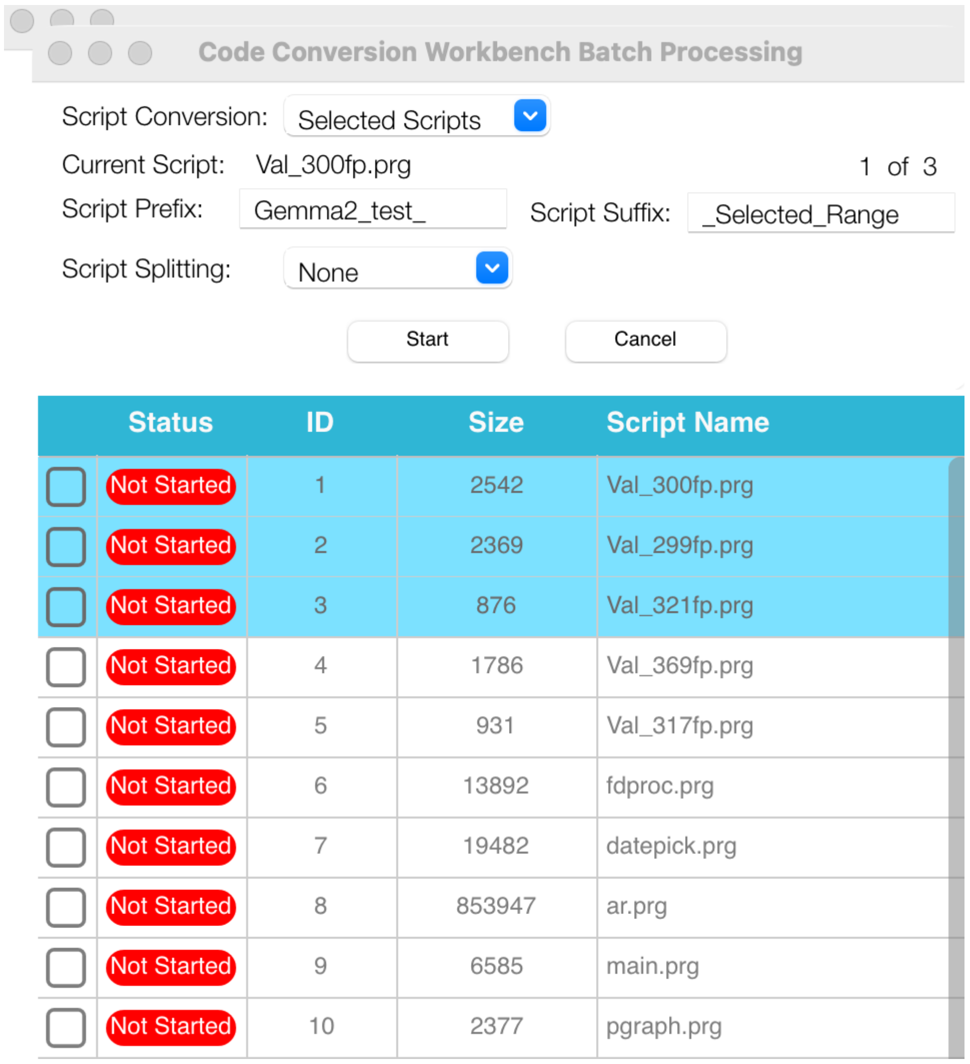
Script Prefix: Gemma2_test_

Script Suffix: _Selected_Range

Script Splitting: None

Once the options have been selected, click the Start button. The converted scripts and Log file will be written into the Converted_Scripts folder.

Processing Window Progress Details



Script Conversion: Selected Scripts

Current Script: Val_300fp.prg 1 of 3

Script Prefix: Gemma2_test_ Script Suffix: _Selected_Range

Script Splitting: None

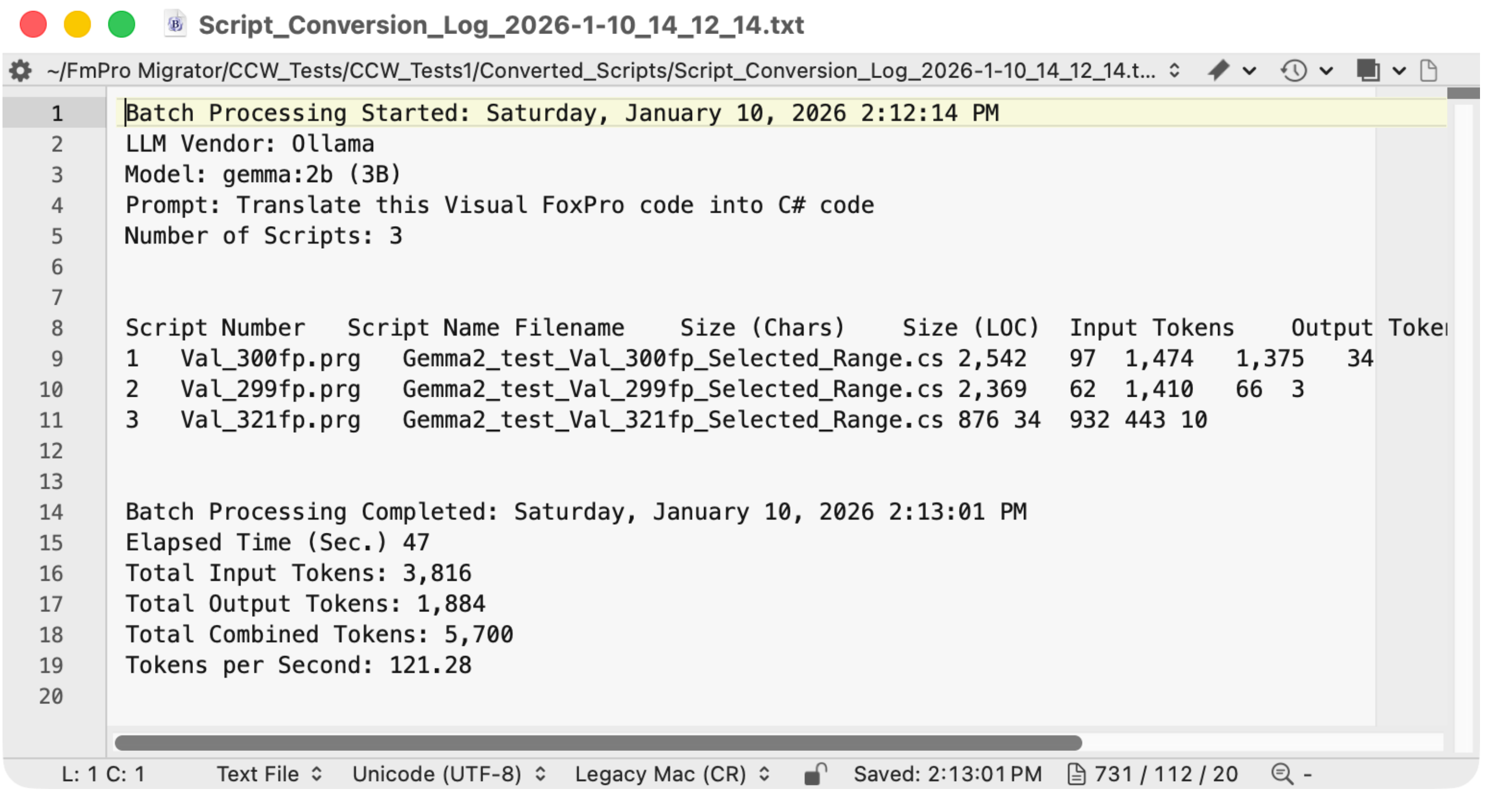
	Status	ID	Size	Script Name
☐	Not Started	1	2542	Val_300fp.prg
☐	Not Started	2	2369	Val_299fp.prg
☐	Not Started	3	876	Val_321fp.prg
☐	Not Started	4	1786	Val_369fp.prg
☐	Not Started	5	931	Val_317fp.prg
☐	Not Started	6	13892	fdproc.prg
☐	Not Started	7	19482	datepick.prg
☐	Not Started	8	853947	ar.prg
☐	Not Started	9	6585	main.prg
☐	Not Started	10	2377	pgraph.prg

As the processing is being done, progress info will be displayed in the Batch Processing window.

Processing Window Cancel

Clicking the Cancel button immediately closes the Batch Processing window, but processing of the current script will continue until the LLM has completed. Starting a new batch is possible before the LLM finishes the current script.

Log File Example - Entire Scripts


Script_Conversion_Log_2026-1-10_14_12_14.txt

~/FmPro Migrator/CCW_Tests/CCW_Tests1/Converted_Scripts/Script_Conversion_Log_2026-1-10_14_12_14.t...

```
Batch Processing Started: Saturday, January 10, 2026 2:12:14 PM
LLM Vendor: Ollama
Model: gemma:2b (3B)
Prompt: Translate this Visual FoxPro code into C# code
Number of Scripts: 3

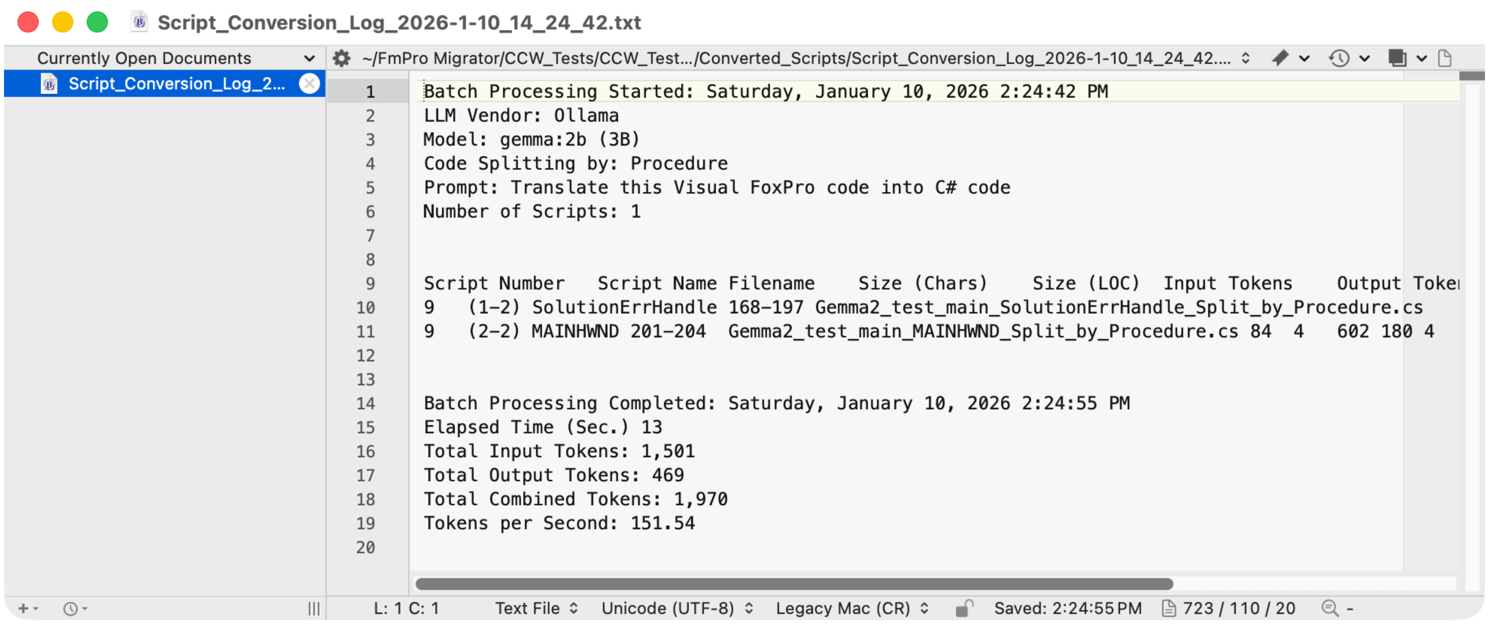

Script Number   Script Name Filename    Size (Chars)    Size (LOC)  Input Tokens    Output Toke
1   Val_300fp.prg   Gemma2_test_Val_300fp_Selected_Range.cs 2,542   97  1,474   1,375   34
2   Val_299fp.prg   Gemma2_test_Val_299fp_Selected_Range.cs 2,369   62  1,410   66  3
3   Val_321fp.prg   Gemma2_test_Val_321fp_Selected_Range.cs 876 34  932 443 10


Batch Processing Completed: Saturday, January 10, 2026 2:13:01 PM
Elapsed Time (Sec.) 47
Total Input Tokens: 3,816
Total Output Tokens: 1,884
Total Combined Tokens: 5,700
Tokens per Second: 121.28
```

When doing batch processing, a log file is created with the results of each processed script. The log file is named: Script_Conversion_Log_YYYY-MM-DD_TIME. There is header/footer text info in the log file with the remainder of the info output as TAB delimited data - ready for pasting into a spreadsheet.

This log shows the model name, total processing time, token usage by file and total usage along with the script prefix/suffix info entered into the Batch Processing window.

Log File Example - Splitting by Procedure



Script_Conversion_Log_2026-1-10_14_24_42.txt

```
Batch Processing Started: Saturday, January 10, 2026 2:24:42 PM
LLM Vendor: Ollama
Model: gemma:2b (3B)
Code Splitting by: Procedure
Prompt: Translate this Visual FoxPro code into C# code
Number of Scripts: 1

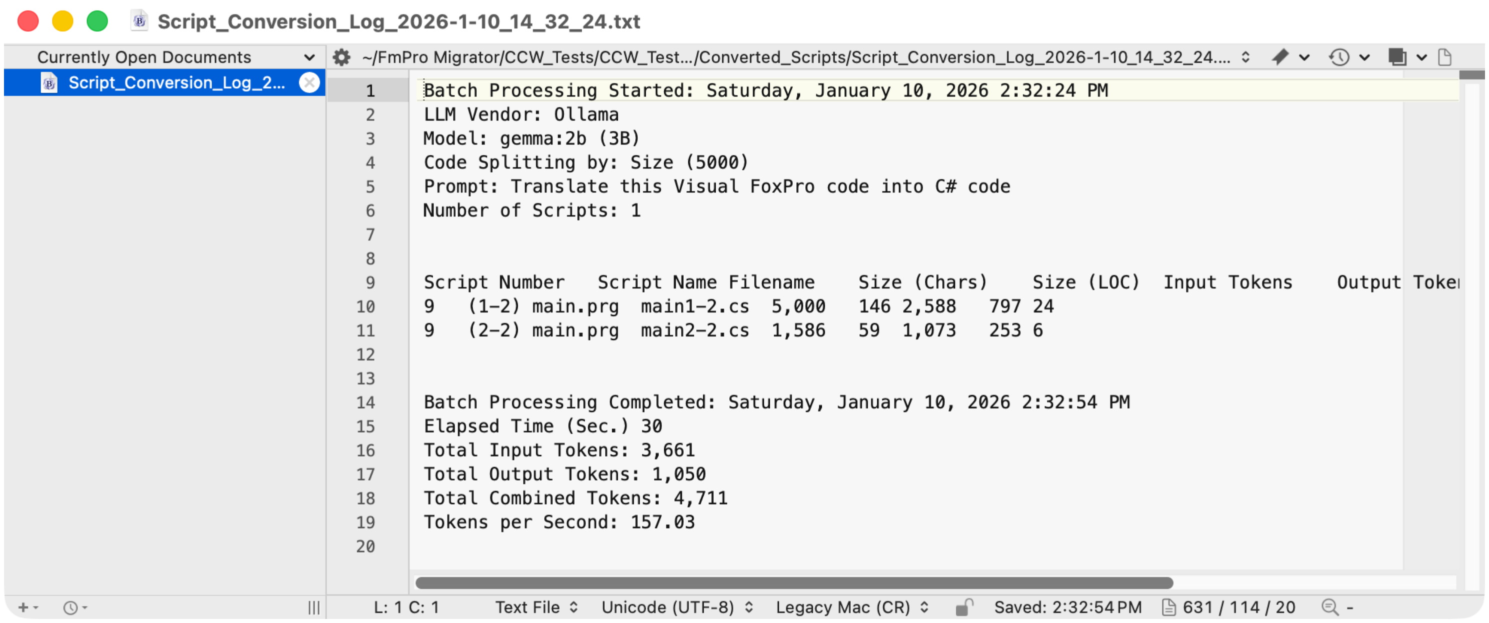

Script Number   Script Name Filename    Size (Chars)    Size (LOC)  Input Tokens    Output Toke
9   (1-2) SolutionErrHandle 168-197 Gemma2_test_main_SolutionErrHandle_Split_by_Procedure.cs
9   (2-2) MAINHWND 201-204  Gemma2_test_main_MAINHWND_Split_by_Procedure.cs 84  4   602 180 4


Batch Processing Completed: Saturday, January 10, 2026 2:24:55 PM
Elapsed Time (Sec.) 13
Total Input Tokens: 1,501
Total Output Tokens: 469
Total Combined Tokens: 1,970
Tokens per Second: 151.54
```

When splitting scripts by procedure, the Opening/Closing blocks of code shown in the Procedures/Functions menu are omitted because they often duplicate existing code within the procedure/function and might include all of the code in the original script - which defeats the purpose of splitting the code.

Looking at this log, you can see that there were only 2 procedures in the file and they have been saved with descriptive names showing the original script name followed by the procedure/function name.

Log File Example - Splitting by Size


Script_Conversion_Log_2026-1-10_14_32_24.txt
Currently Open Documents
Script_Conversion_Log_2...
~/FmPro Migrator/CCW_Tests/CCW_Test.../Converted_Scripts/Script_Conversion_Log_2026-1-10_14_32_24....
L: 1 C: 1
Text File
Unicode (UTF-8)
Legacy Mac (CR)
Saved: 2:32:54 PM
631 / 114 / 20

```
Batch Processing Started: Saturday, January 10, 2026 2:32:24 PM
LLM Vendor: Ollama
Model: gemma:2b (3B)
Code Splitting by: Size (5000)
Prompt: Translate this Visual FoxPro code into C# code
Number of Scripts: 1


Script Number   Script Name Filename    Size (Chars)    Size (LOC)  Input Tokens    Output Toke
9   (1-2) main.prg  main1-2.cs  5,000   146 2,588   797 24
9   (2-2) main.prg  main2-2.cs  1,586   59  1,073   253 6


Batch Processing Completed: Saturday, January 10, 2026 2:32:54 PM
Elapsed Time (Sec.) 30
Total Input Tokens: 3,661
Total Output Tokens: 1,050
Total Combined Tokens: 4,711
Tokens per Second: 157.03
```

The script Prefix/Suffix info has been removed for this test, showing that the file was split into 2 pieces named main1-2.cs, and main2-2.cs based upon processing 5000 characters at a time.

Training

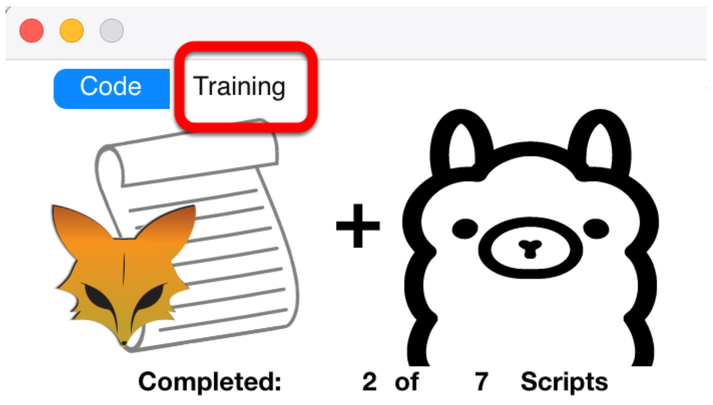
Training Machine Learning Models

The Code Conversion Workbench Training feature empowers developers to customize the automated conversion process, extending the output of LLMs beyond the instructions provided in the prompt.
This feature allows developers to define unique features to be performed by the machine learning model using plain English text examples. It is LLM model-agnostic and instructs the model precisely on how the converted code should be written. Developers can add as many training records as necessary and enable or disable them as required during code conversion.

Several advantages of utilizing the Training feature include:
1) The quality of generated code using Training records is virtually 100% accurate.
2) Mistakes in the output code generated by LLMs can generally be improved to the desired quality level by adding Training records.
3) The overall quality of output generated by LLMs can be enhanced. Faster-performing models running locally through Ollama can achieve a quality comparable to higher-cost public models by using Training records. This is particularly important in situations where customers cannot utilize public LLMs due to security and privacy concerns.

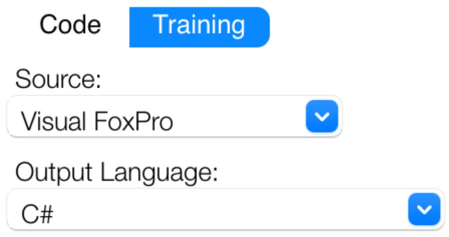
Mastering this feature requires practice and experience. The most challenging aspect is adapting to providing instructions in English to LLMs instead of programming syntax.

Opening the Training Tab



The Training feature is opened by clicking the Training tab at the top left corner of the Code Conversion Workbench window.

Associating Training Records with a Conversion Type



Code | Training

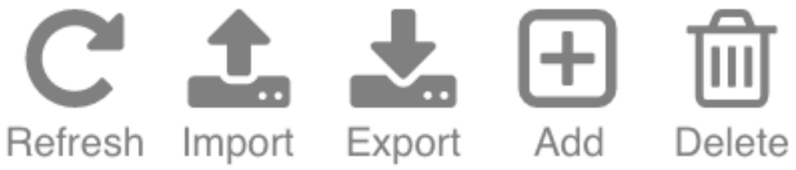
Source: Visual FoxPro

Output Language: C#

Training records are associated with a conversion type selected on the Code Conversion Workbench window (either the Code or Training screens). So for instance if you are converting Visual FoxPro to C# and you create training records for that type of conversion - those records will only be displayed when these menus are selected for Visual FoxPro and C#.

If you wanted to convert Visual FoxPro to FileMaker Pro - then you might have created a separate set of training records for that exact type of conversion project. As soon as the menus are selected, the list of records will be displayed for that combination of Source and Output Language. This is why you will see extra records in the JSON file which might not have been displayed for your exact conversion project.

Importing/Exporting/Adding/Deleting Training Records



By default, the Code Conversion Workbench won't display any training records until the records are either imported or created by the developer. Customers can request the latest training records from .com Solutions Inc.

Training records are stored in the file named CCWTraining.JSON in the following location:

macOS
/Users/<USER>/Library/Preferences/FmPro Migrator/CCWTraining.JSON

Windows
Documents/FmPro Migrator/CCWTraining.JSON

The training records are opened automatically when the Code Conversion Workbench window is opened.

The toolbar above the Training Records grid enables management of these records.

Refresh - Refreshes the list of training records, though in most cases this button won't be needed.

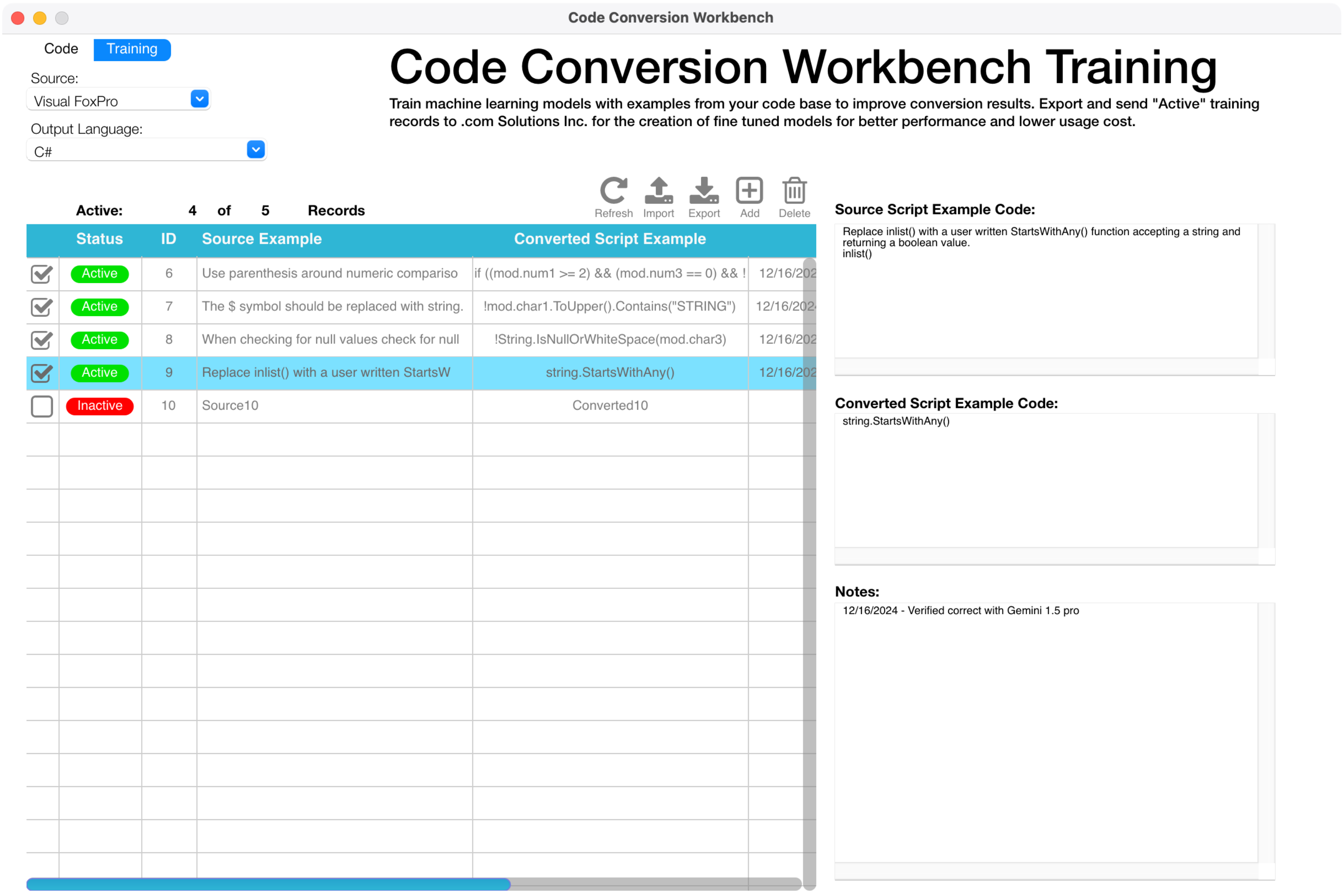
Import - Imports training records sent by .com Solutions Inc. or other developers on a conversion team for instance if the Site License Edition is being used. Only non-duplicated records are imported, based upon a check sum value included within each exported record in the JSON file.

Export - Exports the green colored "Active" records in the grid.

Add - Creates a new blank record ready for adding text. Training records are associated with a Source and Output Language. Please check the Source and Output Language menus when creating a new training record - as these parameters determine when the records will be displayed and used for conversions.

Delete - Deletes the currently selected training record.

Training Tab - FoxPro to C# Examples



Code Conversion Workbench Training

Train machine learning models with examples from your code base to improve conversion results. Export and send "Active" training records to .com Solutions Inc. for the creation of fine tuned models for better performance and lower usage cost.

Source: Visual FoxPro

Output Language: C#

Active: 4 of 5 Records

	Status	ID	Source Example	Converted Script Example	
✓	Active	6	Use parenthesis around numeric compariso	if ((mod.num1 >= 2) && (mod.num3 == 0) && !	12/16/202
✓	Active	7	The $ symbol should be replaced with string.	!mod.char1.ToUpper().Contains("STRING")	12/16/2024
✓	Active	8	When checking for null values check for null	!String.IsNullOrWhiteSpace(mod.char3)	12/16/202
✓	Active	9	Replace inlist() with a user written StartsW	string.StartsWithAny()	12/16/202
	Inactive	10	Source10	Converted10	

Source Script Example Code:

Replace inlist() with a user written StartsWithAny() function accepting a string and returning a boolean value.
inlist()

Converted Script Example Code:

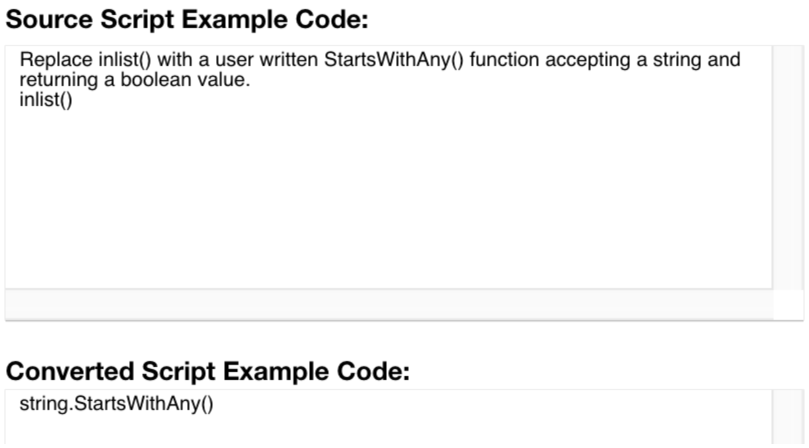
string.StartsWithAny()

Notes:

12/16/2024 - Verified correct with Gemini 1.5 pro

These are some example records used for FoxPro to C# conversions. 4 of the records are Active - which means that they will be used during the conversion process. Records can be Checked/Unchecked as needed for testing purposes to verify how the training records are being used during the conversion. Most of the time these records don't interfere with each other - but if you gave conflicting instructions this could occur.

FoxPro to C# - inlist() Example Training Record



Source Script Example Code:

```
Replace inlist() with a user written StartsWithAny() function accepting a string and
returning a boolean value.
inlist()
```

Converted Script Example Code:

```
string.StartsWithAny()
```

For this example training record, we want the LLMs to do the following:

Convert this FoxPro code:
inlist(job.option,'ABC','DEF','XYZ')

into this C# code:
StartsWithAny(job.option, "ABC", "DEF", "XYZ")

Instead of this default code:
(job.option == "ABC" || job.option == "DEF" || job.option == "XYZ")

This way we can have more control over the automated conversion processing. If we don't add this training record, the LLMs will usually convert the inlist() function to C# in the default manner shown above, which is a little harder to read. We can also have more control over our code base by adding additional features within the StartsWithAny() method in the future. This way we can handle edge cases without having to manually re-write the entire code base looking for all of the individual comparison statements.

Elements of a Good Training Record - inlist() to StartsWithAny()

The goal with adding a training record is to provide an example combined with an English explanation of how we want the code converted.

In this example, there are 2 elements to the Source Script Example Code text:

Part#1
Replace inlist() with a user written StartsWithAny() function accepting a string and returning a boolean value.

Part #1 of the Source example is the English text explaining what we want the LLM to do when it sees the example code. This one single English text sentence makes the biggest difference in whether the LLM follows our instructions. We are basically telling the LLM to look for the FoxPro inlist() function and substitute the StartsWithAny() function or method and we are describing that we want to pass in a string and receive a boolean value to mimic the functionality of the original FoxPro built-in function.

Part#2
inlist()

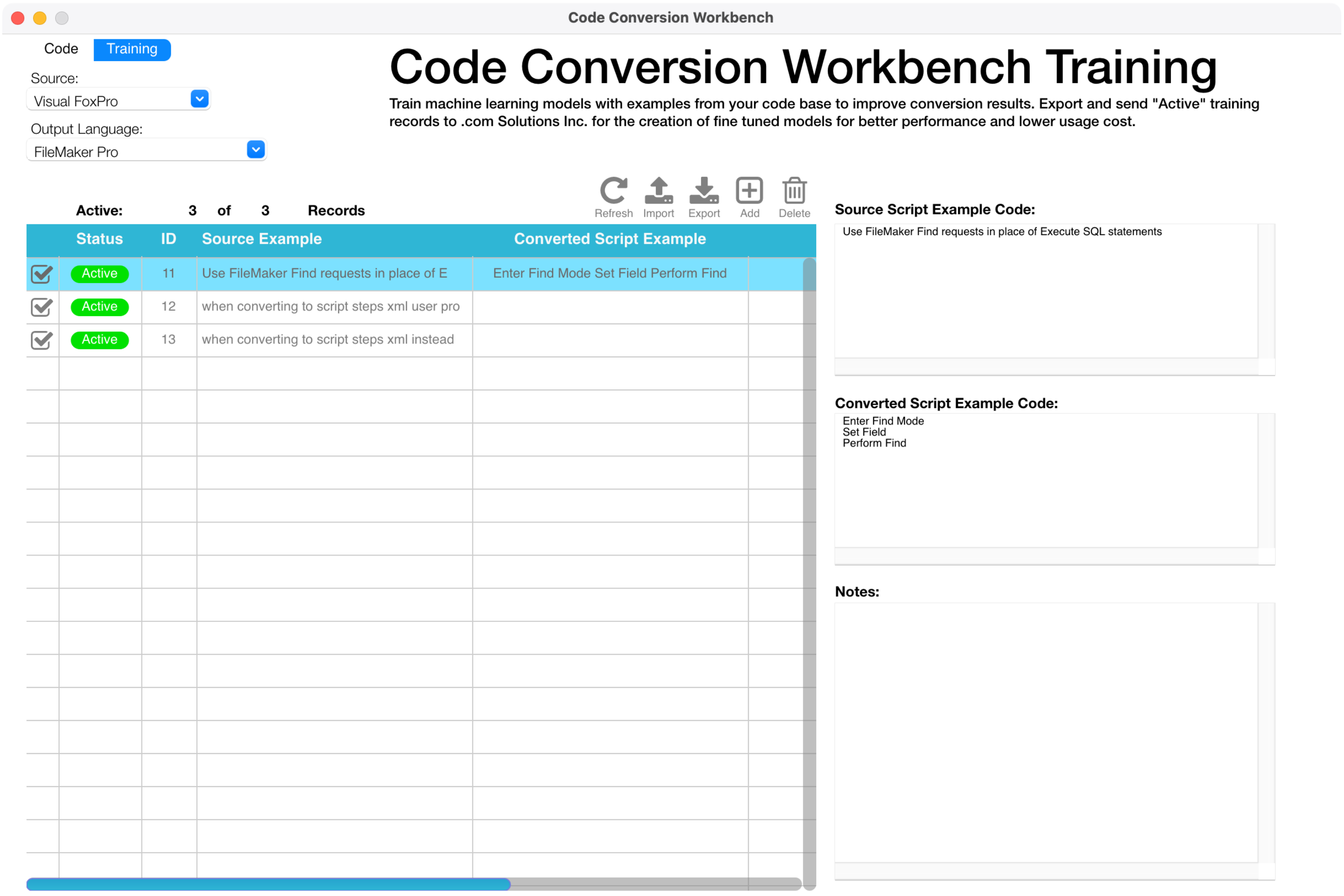
Part#2 of the Source Script Example follows on another line following the text, and provides an example of the FoxPro code.

There is only one part to the Converted Script Example Code text:

Part#1
string.StartsWithAny()

This is the output we want the LLM to create. And this is all we need. The LLMs have enough info about the Source and Output languages to know that the parameters need to be added within the parenthesis, and for each instruction converted these parameters will be added automatically where they are needed.

Training Tab - FoxPro to FileMaker Pro Examples



Code Conversion Workbench Training

Train machine learning models with examples from your code base to improve conversion results. Export and send "Active" training records to .com Solutions Inc. for the creation of fine tuned models for better performance and lower usage cost.

Source: Visual FoxPro

Output Language: FileMaker Pro

Active: 3 of 3 Records

	Status	ID	Source Example	Converted Script Example
✓	Active	11	Use FileMaker Find requests in place of E	Enter Find Mode Set Field Perform Find
✓	Active	12	when converting to script steps xml user pro	
✓	Active	13	when converting to script steps xml instead	

Source Script Example Code:

Use FileMaker Find requests in place of Execute SQL statements

Converted Script Example Code:

```
Enter Find Mode
Set Field
Perform Find
```

Notes:

These are 3 example records for FoxPro to FileMaker Pro conversions. All of the records are Active - which means that they will be used during the code conversion process.

Elements of a Good Training Record - FoxPro to FileMaker - Find Requests

In this example, there are 2 elements to the Source Script Example Code text:

Part#1

Use FileMaker Find requests in place of Execute SQL statements

We are telling the LLM to create FileMaker Find requests instead of using Execute SQL steps. If we don't provide this training record, the LLMs will generally use Execute SQL commands as a replacement for the SQL commands and queries found in the FoxPro code. Using Find Requests will result in a Found Set which will be immediately usable for looping through the records, sorting or direct display to the user via a Go to Layout step to a SubSummary layout. This is easier than putting the Execute SQL results into a variable and then working thru the results to write them into a temp database table for instance.

Part#2

Part#2 of the Source Script Example isn't needed for this training record.

There are several lines used for the Converted Script Example Code text:

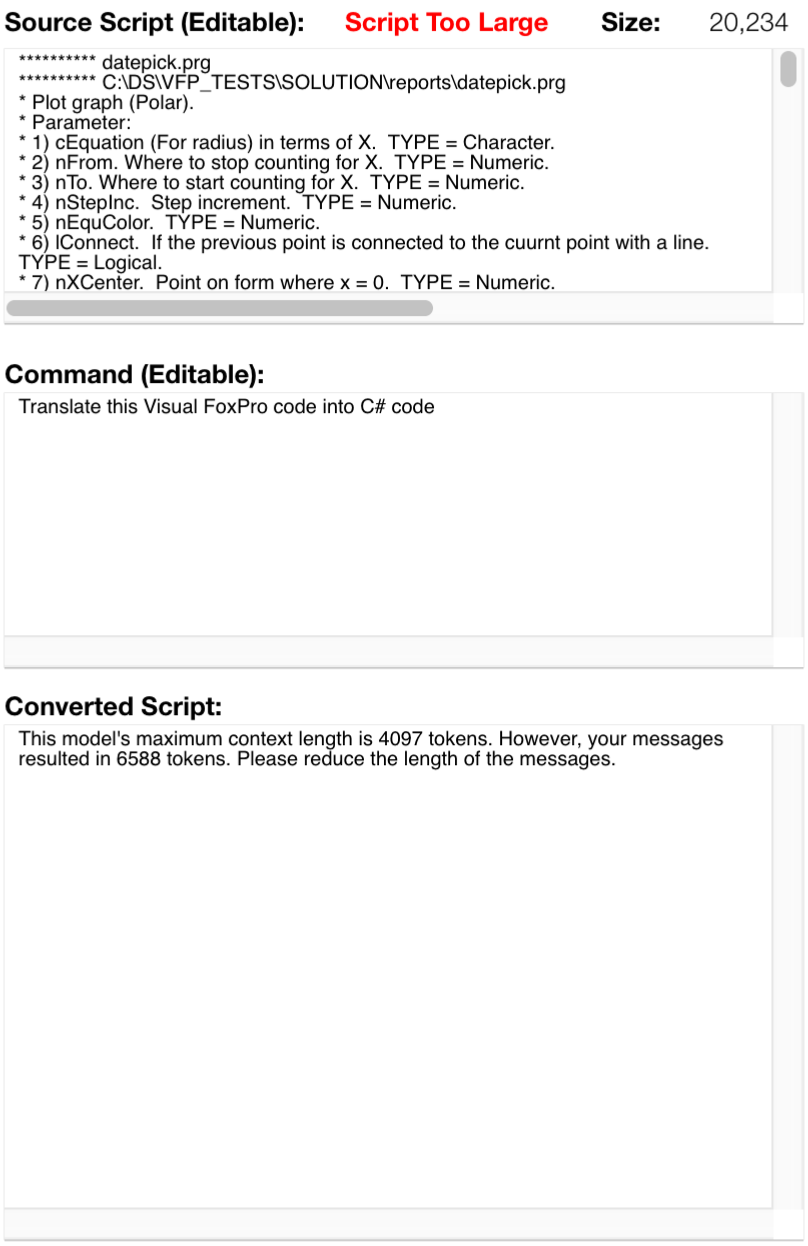
Part#1
Enter Find Mode
Set Field
Perform Find

This text provides an example of the FileMaker script steps required to implement a Find Request.

The result is that the LLMs generally do a very good job of creating Find Requests for FileMaker Pro queries as a replacement for SQL statements in the original FoxPro application.

Troubleshooting

Script Too Large - Error Returned From Model



```
********** datepick.prg
********** C:\DS\VFP_TESTS\SOLUTION\reports\datepick.prg
* Plot graph (Polar).
* Parameter:
* 1) cEquation (For radius) in terms of X.  TYPE = Character.
* 2) nFrom. Where to stop counting for X.  TYPE = Numeric.
* 3) nTo. Where to start counting for X.  TYPE = Numeric.
* 4) nStepInc.  Step increment.  TYPE = Numeric.
* 5) nEquColor.  TYPE = Numeric.
* 6) lConnect.  If the previous point is connected to the cuurnt point with a line.
TYPE = Logical.
* 7) nXCenter.  Point on form where x = 0.  TYPE = Numeric.
```

Translate this Visual FoxPro code into C# code

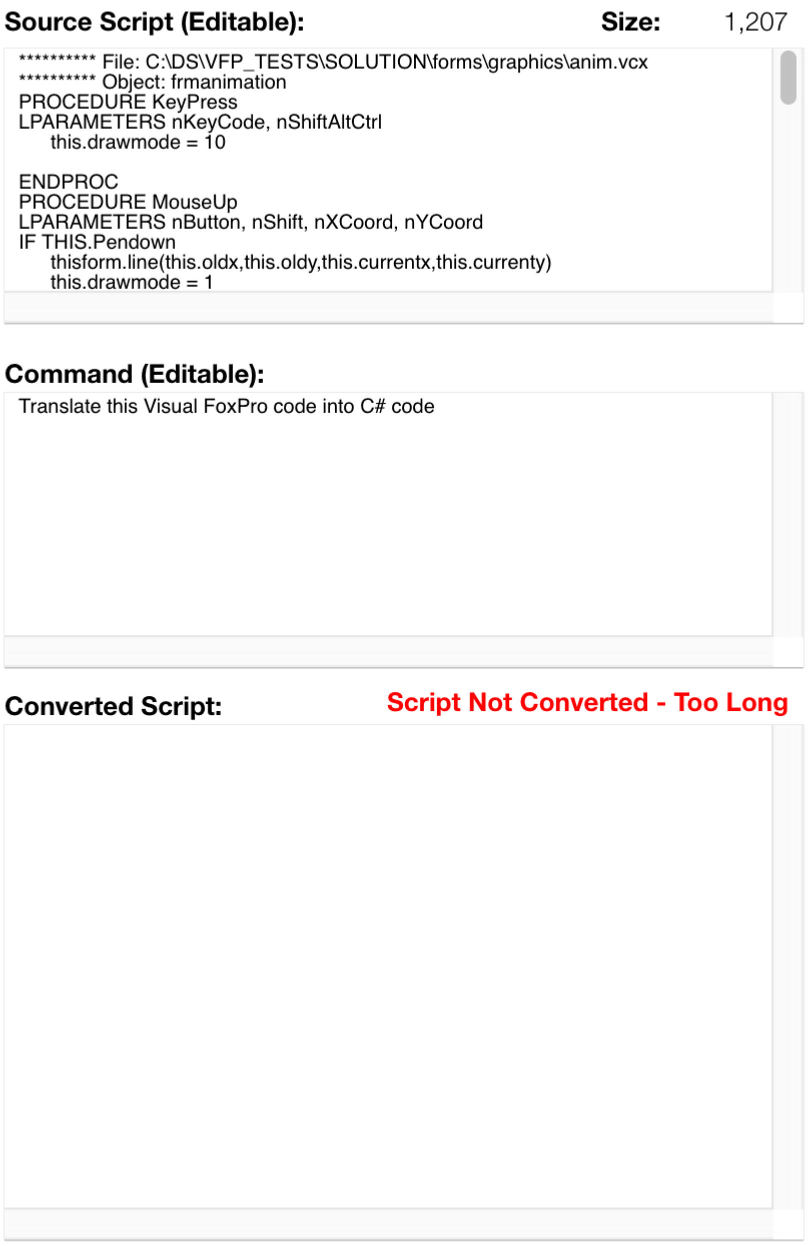
```
This model's maximum context length is 4097 tokens. However, your messages
resulted in 6588 tokens. Please reduce the length of the messages.
```

This screenshot shows an example of a large script which exceeds the 4097 tokens available with the selected AI model. This message was returned from the server and it shows that there are 6589 tokens contained in the 20,234 characters of text sent to the model. **A token represents approximately 1 to 3 characters of text.** Fortunately, there are models available with a capacity of up to 16,000 tokens.

1) One potential solution is to use one of the 16K token capacity models like gpt-3.5-turbo-16k.

2) Another option is to try breaking up the script into individual procedures/functions. This can be advantageous even when a 16K model is available because sometimes the larger capacity models get overloaded.

Script Not Converted - Too Long [Red Warning Text]



```
********** File: C:\DS\VFP_TESTS\SOLUTION\forms\graphics\anim.vcx
********** Object: frmanimation
PROCEDURE KeyPress
LPARAMETERS nKeyCode, nShiftAltCtrl
    this.drawmode = 10

ENDPROC
PROCEDURE MouseUp
LPARAMETERS nButton, nShift, nXCoord, nYCoord
IF THIS.Pendown
    thisform.line(this.oldx,this.oldy,this.currentx,this.currenty)
    this.drawmode = 1
```

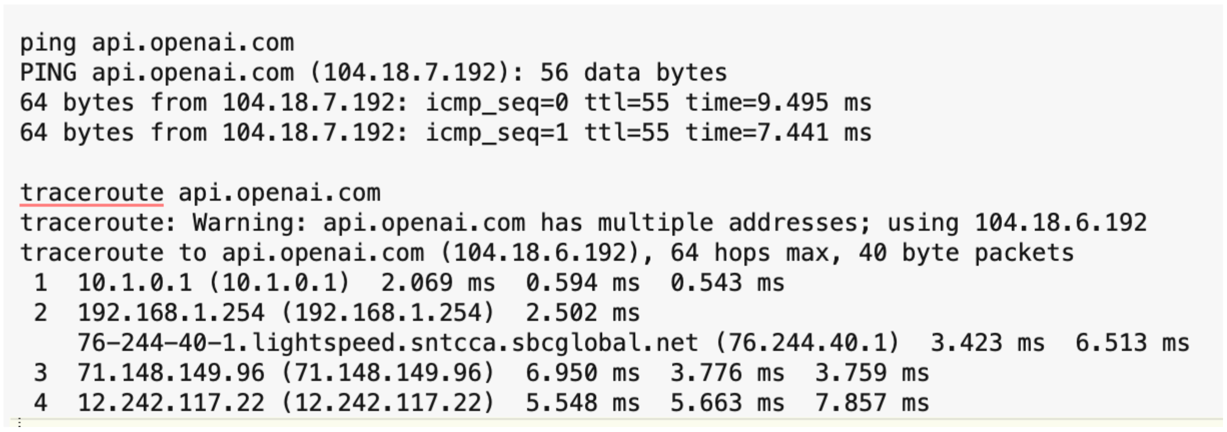
Translate this Visual FoxPro code into C# code

Script Not Converted - Too Long

This error is not returned from the server, it is displayed by the Code Conversion Workbench as a result of not receiving a response from the server. Therefore it is estimated by the software that the result could be due to the script being too long.
But it is also possible that the model was overloaded on the server. The gpt4 model was used for this test and experience has shown that the gpt4 model seems to get overloaded more often than the gpt-3.5-turbo model.

1) One possible solution is to break up the script into smaller amounts of text. Even though the script is reasonably sized, but since this script contains 4 individual procedures they could be converted separately - especially if you really require the additional code conversion quality of the gpt4 model.
2) Switching models from gpt4 to gpt-3.5-turbo is another option.
3) Trying again after a few minutes if the model is overloaded. Sometimes the models seem to work more quickly on weekends compared with during the work week.
4) Verify that your internet connection is functioning properly.

OpenAI API Endpoint Connectivity



```
ping api.openai.com
PING api.openai.com (104.18.7.192): 56 data bytes
64 bytes from 104.18.7.192: icmp_seq=0 ttl=55 time=9.495 ms
64 bytes from 104.18.7.192: icmp_seq=1 ttl=55 time=7.441 ms

traceroute api.openai.com
traceroute: Warning: api.openai.com has multiple addresses; using 104.18.6.192
traceroute to api.openai.com (104.18.6.192), 64 hops max, 40 byte packets
 1  10.1.0.1 (10.1.0.1)  2.069 ms  0.594 ms  0.543 ms
 2  192.168.1.254 (192.168.1.254)  2.502 ms
    76-244-40-1.lightspeed.sntcca.sbcglobal.net (76.244.40.1)  3.423 ms  6.513 ms
 3  71.148.149.96 (71.148.149.96)  6.950 ms  3.776 ms  3.759 ms
 4  12.242.117.22 (12.242.117.22)  5.548 ms  5.663 ms  7.857 ms
```

Problem Symptoms:

1) The models menu Refresh button clears the list of models.
2) The busy indicator spins and doesn't return results or any error message after clicking the Convert button.

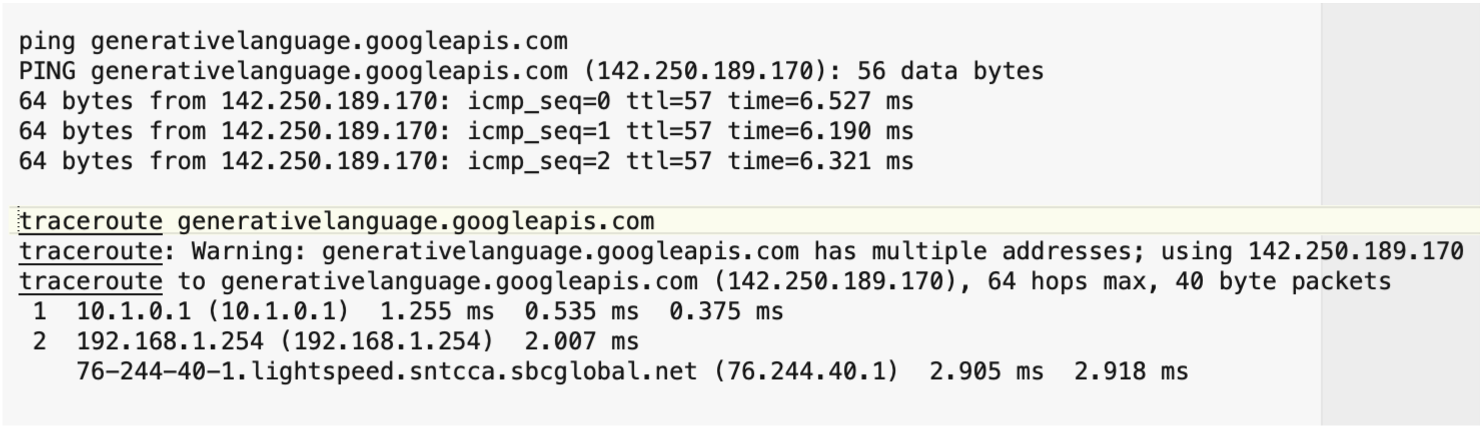
FmPro Migrator needs to have internet access to the OpenAI endpoint when using OpenAI machine learning models. This URL is:

api.openai.com

Troubleshooting:
Try performing a ping or traceroute command on the endpoint URL to insure that your computer can connect with the OpenAI API service.

If you cannot reach the endpoint URL, please check with your internet provider or corporate netoworking/security team for assistance.

Google Gemini API Endpoint Connectivity



```
ping generativelanguage.googleapis.com
PING generativelanguage.googleapis.com (142.250.189.170): 56 data bytes
64 bytes from 142.250.189.170: icmp_seq=0 ttl=57 time=6.527 ms
64 bytes from 142.250.189.170: icmp_seq=1 ttl=57 time=6.190 ms
64 bytes from 142.250.189.170: icmp_seq=2 ttl=57 time=6.321 ms

traceroute generativelanguage.googleapis.com
traceroute: Warning: generativelanguage.googleapis.com has multiple addresses; using 142.250.189.170
traceroute to generativelanguage.googleapis.com (142.250.189.170), 64 hops max, 40 byte packets
 1  10.1.0.1 (10.1.0.1)  1.255 ms  0.535 ms  0.375 ms
 2  192.168.1.254 (192.168.1.254)  2.007 ms
    76-244-40-1.lightspeed.sntcca.sbcglobal.net (76.244.40.1)  2.905 ms  2.918 ms
```

Problem Symptoms:

1) The models menu Refresh button clears the list of models.
2) The busy indicator spins and doesn't return results or any error message after clicking the Convert button.

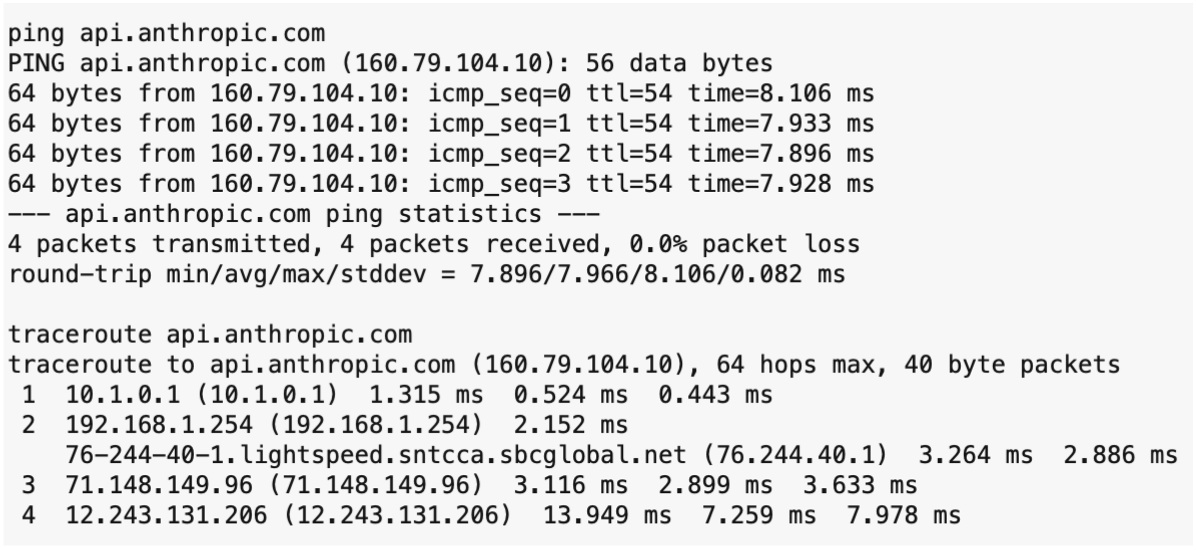
FmPro Migrator needs to have internet access to the Google Gemini API endpoint when using Google machine learning models. This URL is:

generativelanguage.googleapis.com

Troubleshooting:
Try performing a ping or traceroute command on the endpoint URL to insure that your computer can connect with the Google Gemini API service.

If you cannot reach the endpoint URL, please check with your internet provider or corporate netoworking/security team for assistance.

Anthropic API Endpoint Connectivity



```
ping api.anthropic.com
PING api.anthropic.com (160.79.104.10): 56 data bytes
64 bytes from 160.79.104.10: icmp_seq=0 ttl=54 time=8.106 ms
64 bytes from 160.79.104.10: icmp_seq=1 ttl=54 time=7.933 ms
64 bytes from 160.79.104.10: icmp_seq=2 ttl=54 time=7.896 ms
64 bytes from 160.79.104.10: icmp_seq=3 ttl=54 time=7.928 ms
--- api.anthropic.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 7.896/7.966/8.106/0.082 ms

traceroute api.anthropic.com
traceroute to api.anthropic.com (160.79.104.10), 64 hops max, 40 byte packets
 1  10.1.0.1 (10.1.0.1)  1.315 ms  0.524 ms  0.443 ms
 2  192.168.1.254 (192.168.1.254)  2.152 ms
    76-244-40-1.lightspeed.sntcca.sbcglobal.net (76.244.40.1)  3.264 ms  2.886 ms
 3  71.148.149.96 (71.148.149.96)  3.116 ms  2.899 ms  3.633 ms
 4  12.243.131.206 (12.243.131.206)  13.949 ms  7.259 ms  7.978 ms
```

Problem Symptoms:

1) The models menu Refresh button clears the list of models.
2) The busy indicator spins and doesn't return results or any error message after clicking the Convert button.

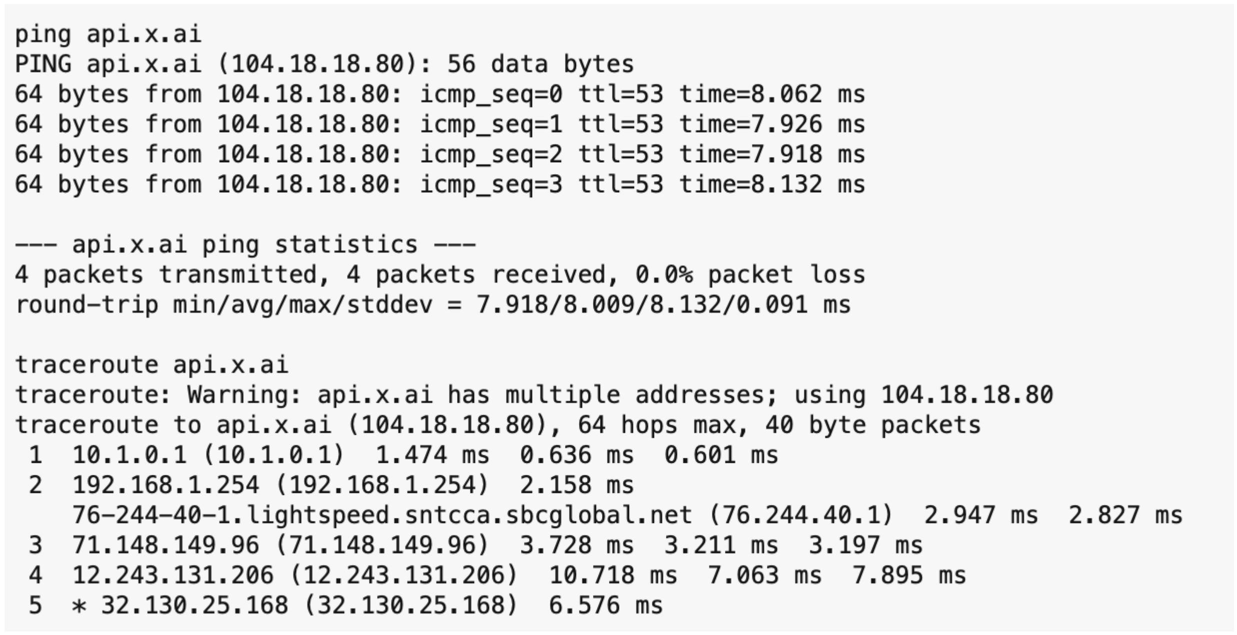
FmPro Migrator needs to have internet access to the Anthropic API endpoint when using Anthropic machine learning models. This URL is:

api.anthropic.com

Troubleshooting:
Try performing a ping or traceroute command on the endpoint URL to insure that your computer can connect with the Anthropic API service.

If you cannot reach the endpoint URL, please check with your internet provider or corporate netoworking/security team for assistance.

xAI API Endpoint Connectivity



```
ping api.x.ai
PING api.x.ai (104.18.18.80): 56 data bytes
64 bytes from 104.18.18.80: icmp_seq=0 ttl=53 time=8.062 ms
64 bytes from 104.18.18.80: icmp_seq=1 ttl=53 time=7.926 ms
64 bytes from 104.18.18.80: icmp_seq=2 ttl=53 time=7.918 ms
64 bytes from 104.18.18.80: icmp_seq=3 ttl=53 time=8.132 ms

--- api.x.ai ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 7.918/8.009/8.132/0.091 ms

traceroute api.x.ai
traceroute: Warning: api.x.ai has multiple addresses; using 104.18.18.80
traceroute to api.x.ai (104.18.18.80), 64 hops max, 40 byte packets
 1  10.1.0.1 (10.1.0.1)  1.474 ms  0.636 ms  0.601 ms
 2  192.168.1.254 (192.168.1.254)  2.158 ms
    76-244-40-1.lightspeed.sntcca.sbcglobal.net (76.244.40.1)  2.947 ms  2.827 ms
 3  71.148.149.96 (71.148.149.96)  3.728 ms  3.211 ms  3.197 ms
 4  12.243.131.206 (12.243.131.206)  10.718 ms  7.063 ms  7.895 ms
 5  * 32.130.25.168 (32.130.25.168)  6.576 ms
```

Problem Symptoms:

1) The models menu Refresh button clears the list of models.
2) The busy indicator spins and doesn't return results or any error message after clicking the Convert button.

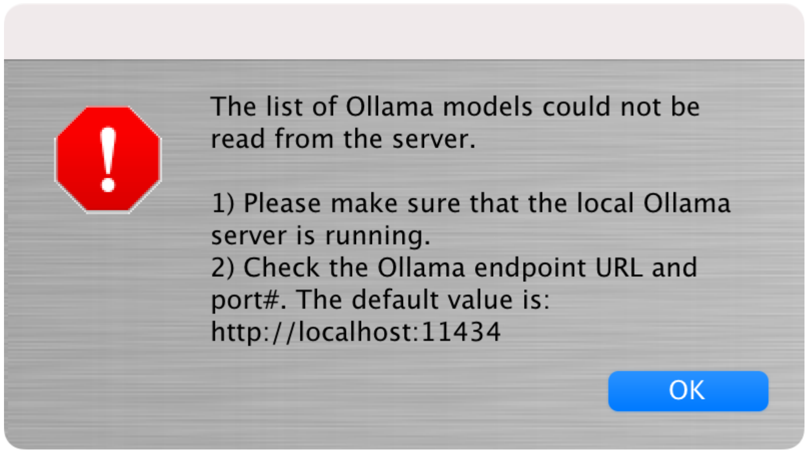
FmPro Migrator needs to have internet access to the xAI API endpoint when using xAI machine learning models. This URL is:

api.x.ai

Troubleshooting:
Try performing a ping or traceroute command on the endpoint URL to insure that your computer can connect with the xAI API service.

If you cannot reach the endpoint URL, please check with your internet provider or corporate netoworking/security team for assistance.

Ollama Localhost API Endpoint Connectivity



The list of Ollama models could not be read from the server.

1) Please make sure that the local Ollama server is running.
2) Check the Ollama endpoint URL and port#. The default value is:
http://localhost:11434

Problem Symptoms:

1) The models Refresh button displays the above error message because the Ollama server cannot be located.
2) The red error result text "Script Not Converted - Too Long?" will immediately be displayed above the Converted Script field as soon as the Convert button is clicked.

FmPro Migrator needs to have local internet access to the Ollama server endpoint when using Google machine learning models. The default URL for this local server is:

http://localhost:11434

Troubleshooting:
1) Make sure that the Ollama server is running. The server is started when launching the Ollama app, and a menu item will be displayed enabling control of the server. After the initial installation/configuration the app will quit as soon as it is launched - leaving the menu item.
2) You can go to the endpoint URL with a web browser and verify that you see the text:
Ollama is running

Note: When initially configured, "localhost" is the only address from which the Ollama server will respond.
Running the following command on macOS enables connections from other computers - but this command doesn't persist across restarts:
launchctl setenv OLLAMA_HOST "0.0.0.0"
The text of the next section can be placed into a file named ollama.plist which will persist these settings across restarts. The plist file gets installed in the path:
/Library/LaunchDaemons/ollama.plist

On Windows, the various parameters are configured with environment variables according to this

article:
https://docs.dify.ai/tutorials/model-configuration/ollama

In the plist info - the user name has been replaced with <USER> which needs changed to represent the username under which Ollama is installed.

The localhost:11434 URL doesn't need to be entered into the Endpoint URL field, if it is left blank the default value shown above will be used automatically.

Ollama plist for macOS - Enables Multi-User Connections

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
   <key>Label</key>
   <string>ollama</string>
   <key>StandardOutPath</key>
   <string>/Users/<USER>/.ollama/launchd.stdout.log</string>
   <key>StandardErrorPath</key>
   <string>/Users/<USER>/.ollama/launchd.stderr.log</string>
   <key>EnvironmentVariables</key>
   <dict>
      <key>OLLAMA_HOST</key>
      <string>0.0.0.0:11434</string>
   </dict>
   <key>ProgramArguments</key>
   <array>
    <string>/usr/local/bin/ollama</string>
    <string>serve</string>
   </array>
   <key>UserName</key>
   <string><USER></string>
   <key>GroupName</key>
   <string>staff</string>
   <key>ExitTimeOut</key>
   <integer>30</integer>
   <key>Disabled</key>
   <false />
   <key>KeepAlive</key>
```

```
    <true />
  </dict>
</plist>
```