# FmPro Migrator - BASIC to revTalk Conversion

.com Solutions Inc.

# 1 BASIC to revTalk Conversion

# BASIC to revTalk Conversion

# Introduction - BASIC to revTalk Conversion

This document provides an explanation of the steps required to convert BASIC scripts to revTalk scripts using FmPro Migrator Platinum Edition.

Revision 01
1/11/2010

## About the PHP to revTalk Conversion Process

The BASIC to revTalk conversion process is designed to convert all of the BASIC files within a source directory, including all subdirectories. FmPro Migrator is designed to read BASIC files having a variety of file extensions (including: bas, vba, vbs or txt) during the conversion process.

Each file within the source directory is read into memory and analyzed on a line by line basis. Keywords and operators are read and converted to the equivalent keywords and operators in revTalk.

## BASIC Code Processing Features

1) Traversal of the files and subdirectories of the selected source directory. Re-creation of the same file and directory structure within the selected destination directory. Quickly convert all of the .bas files within the source directory and subdirectories.
2) Processing of .bas, .vba, .vbs and .txt source files, including support for Visual Basic, PowerBasic, ZBASIC/FutureBasic, RealBasic, VBScript and VisualBasic for Applications code.
3) High performance processing. Process hundreds of files and hundreds of thousands of lines of code in seconds.
4) Code indenting is maintained for most situations. The exception is CASE statements where additional Default statements get added.
5) Line continuation characters are removed, as part of the code parsing process.
6) Compiler directives starting with "$" are commented out.
7) DIM/STATIC commands are converted into local/global variable definitions in revTalk, in which the "As <VariableType>" definitions are removed and the static memory quantity is also removed, since it isn't needed in revTalk. For instance the statement
DIM ParameterData2$(3750)
gets changed to
local ParameterData2
8) BASIC variable suffix characters (%, $) are removed from most instructions.
9) Each variable definition is checked to insure that it does't exactly match an existing revTalk keyword. If a conflict exists between a variable name and an existing keyword, then an underscore character is added to the variable name.
10) Variable assignments are converted into revTalk "put" commands. Colon or single quote comment operators following the assignment on the same line will be commented using the "--" revTalk operator

and placed to the right side of the assignment statement. Variable suffixes will be removed and the variable names checked against the revTalk keywords list on the left side of the assignment operator.
11) Private Sub definitions are converted into revTalk private command handlers with the "end sub" replaced with "end <command name>".
12) Public Sub definitions are converted into revTalk private command handlers with the "end sub" replaced with "end <command name>".
13) OPEN/INPUT/CLOSE commands using file numbers (#1, #2 etc.) are partially converted into revTalk open/read/close file commands. Manual changes will be required in order to specify
14) The BASIC For loop keyword is converted into a "repeat with" keyword in revTalk. The For loop variable is fully checked for revTalk keyword conflicts and BASIC variable suffix characters are removed. BASIC variable suffix characters are removed from the remaining variables.
15) BASIC code labels ending with a colon are commented out.
16) ON Error statements are commented out, along with GOTO commands. Error checking can be rewritten using Try/Catch statements in revTalk.
17) Passing parameters by reference using the ByVal keyword are converted into the @ reference passing character used in revTalk.
18) Functions having optional arguments specified with the Optional keyword will have this optional keyword removed. If default values are needed for the parameters, these should be defined manually within the function.
19) SELECT CASE statements are converted into revTalk SWITCH statements. Each CASE statement is closed with a break statement. CASE ELSE statements are converted into "default" statements.
20) The DO WHILE/DO UNTIL keywords are converted into a "repeat while/until" statements, with the closing LOOP keyword converted into an "end repeat" statement. The LOOP keyword is processed if it is the last word of an instruction containing other keywords and if it is on a line by itself.
21) The LET keyword is removed and the instruction is processed as a standard assignment statement.
22) The LONG IF keyword is processed the same as a regular IF keyword. (ZBASIC/FutureBasic).
23) SLEEP <interval> is converted into "wait <interval> milliseconds with messages".
24) Function return parameters specifying the function name will be replaced with the revTalk return keyword.

## BASIC Omitted Functions and Keywords

Some Basic functions are purposely omitted from the conversion processing because they require manual work:
1) Instr(variable1, ".") -> offset(".", variable1] - The string to find and string to search parameter positions need to be swapped manually when converting to revTalk.
2) Basic "+" operators can be used to represent either a mathematical operation or string concatenation.
3) bin$() -> binaryEncode() - The revTalk formatting needs to be manually applied to the data to be converted into binary. (ZBASIC/FutureBasic)

4) box, circle, button -> Create an object using the appropriate style of specified object. (ZBASIC/FutureBasic)
5) hex$() -> binaryDecode() - The revTalk formatting needs to be manually applied to the data to be converted into hex. (ZBASIC/FutureBasic)
6) kill path$ -> delete file <filepath> (ZBASIC/FutureBasic)
7) WAITKEY$ - There isn't an equivalent command to wait for keystrokes entered via the command line, which is how commands like INPUT$ and WAITKEY$ are used. revTalk is event driven, so messages such as entering keystrokes within fields, and tabbing between fields can be captured and used to trigger code to run at the appropriate time.

## Unsupported Features Requiring Manual Conversion

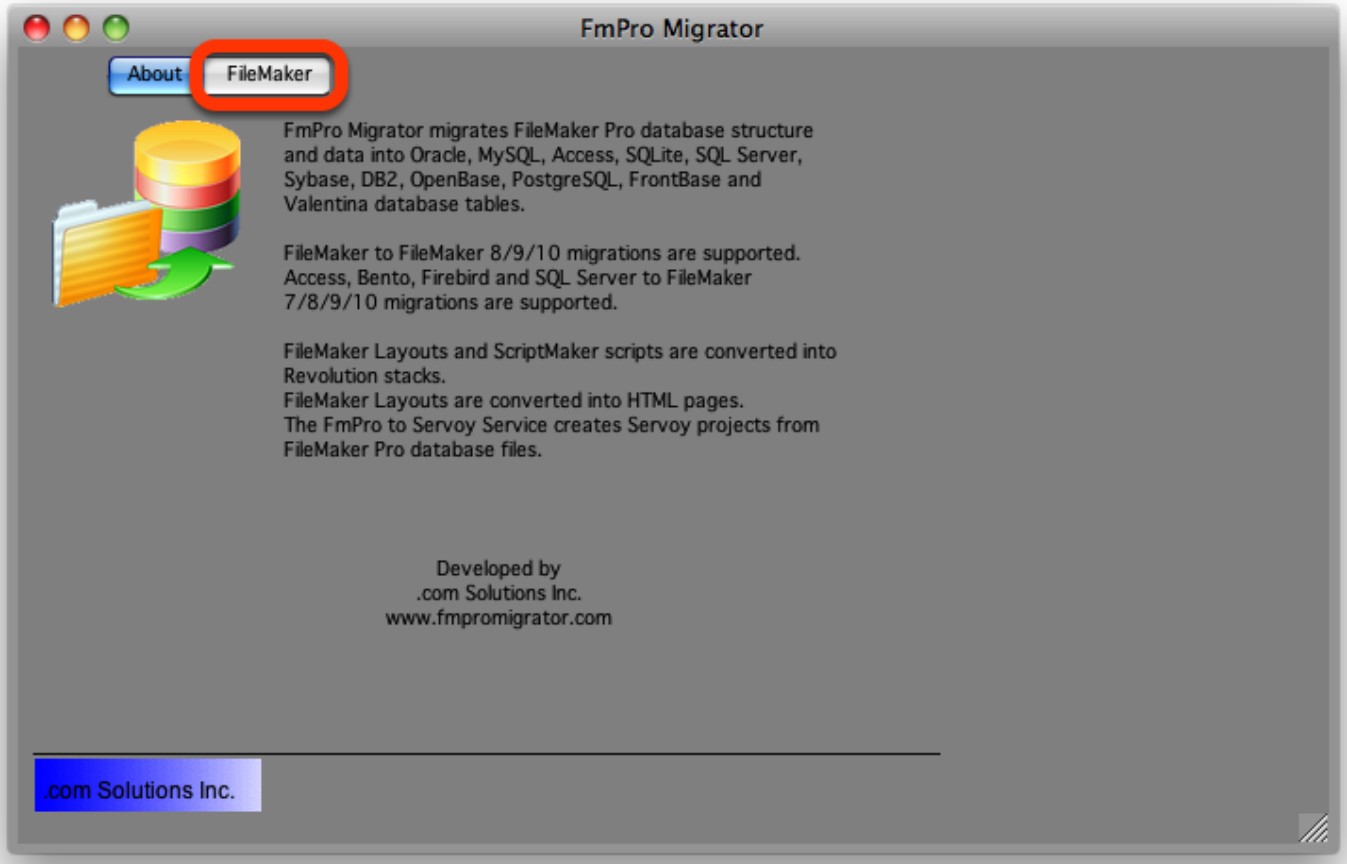
1) Poorly formed BASIC code will not get fully processed. For instance, keywords, operators and variables should generally be separated from each other by at least one space character.
2) One instruction per line can be parsed correctly. The additional instructions of BASIC code on the same line won't be completely converted. However commented code to the right of variable assignment statements will be commented appropriately.
3) Operating system specific or BASIC language specific functions won't be converted.
4) BASIC external libraries or calls to installed .dll files.
5) Syntax errors or other types of problems which prevent the BASIC code from executing won't be corrected.
6) BASIC object oriented code features won't be converted, but will remain in the converted code for reference purposes. For instance Button1.Caption won't be converted into "the name of button Button1". This is due to the difficulty of determining the type of object referenced in the original code (button, field, window etc.).
7) Multiple variable assignments within the same instruction. These statements need to be separated into two separate instructions.
8) Automated conversion of twips to pixel coordinates between Visual Basic and revTalk is not implemented. There are 1440 twips per inch and approximately 80 twips per pixel (depending upon screen resolution). revTalk uses pixel based coordinates when defining object locations.
9) BASIC On Error Goto ErrorHandler code remains unconverted.
10) The use of # characters in place of double quotes for enclosing date values being assigned to a variable.
11) Removal of Visual Basic object type names during the conversion of object variable DIM statements. This limitation also applies to user defined data types.
12) IsEmpty(), Null and Error value implementations. There is too much chance of error if doing this type of conversion using a simple text replacement algorithm. But if an assignment statement is found to contain the text "Nothing" it is replaced with "empty" for compatibility with revTalk.
13) ReDim statements are commented out, because they could either be used to clear an array or resize the array. The actual usage needs to be determined manually.
14) Visual Basic code using Collections.

15) Passing of named parameters to functions or handlers is not supported in revTalk, and needs to be changed manually in the source code.
16) ELSEIF statements should be manually converted into SWITCH/CASE statements if there are more than 2 conditions being checked. This change will also make the code easier to read, understand and troubleshoot.
17) The DO WHILE/DO UNTIL statements are converted if both keywords are on the same line with each other. If the DO keyword is separated from the WHILE/UNTIL keyword then it won't be converted.
18) Very few recorded VBA Macro statements will be directly converted due to the reliance upon application-specific objects, properties and methods. Most of these features will only available within the original Microsoft application.
19) Array references will be converted from parenthesis () to square brackets [] on the left hand side of the assignment operator. This change won't occur for array references on the right hand side.
20) WITH/END WITH keywords are not applicable in revTalk, so they are commented out.
21) The RealBasic CountFields/CountFieldsB() function should be manually replaced with setting the itemDelimiter to the field delimiter and then getting the count of the number of items in the container.
22) The ZBASIC/FutureBasic cursor cursorID is converted into "set the cursor to" but it will be necessary to change the constants to the appropriate values used in revTalk (watch, arrow etc).
23) The DELAY command is converted into "wait for". If you want to specify "with messages" then this text should be added manually along with a unit of time. (ZBASIC/FutureBasic).
24) PRINT and STDOUT keywords are converted into revTalk "put" statements, which will by default send the output to the message box during development. This is probably not what you want with a modern application. Further work should be done to analyze how this info should be presented to the user within the context of an event driven development process.
25) PowerBasic Embedded x86 assembler in-line code which starts with a "!" character will be commented out, since this is not applicable to revTalk development.

## Step 1 - Create FmPro Migrator Project File

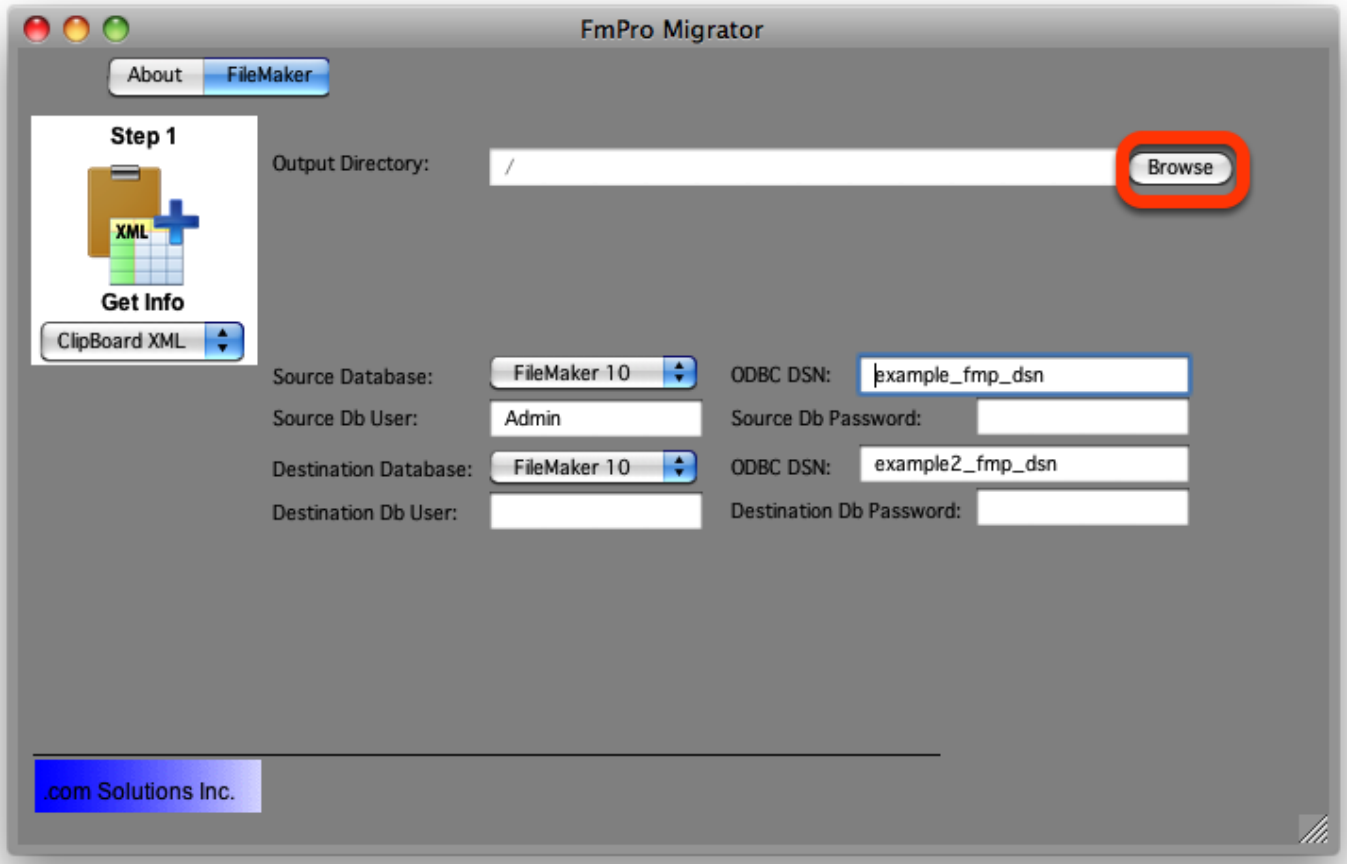
In order to perform a migration project, FmPro Migrator needs to create a MigrationProcess.db3 project file to store information about the migration project. Code conversion projects work a little differently than database conversion projects, so the Create Project File... menu is used to get the process started.

### Open FmPro Migrator



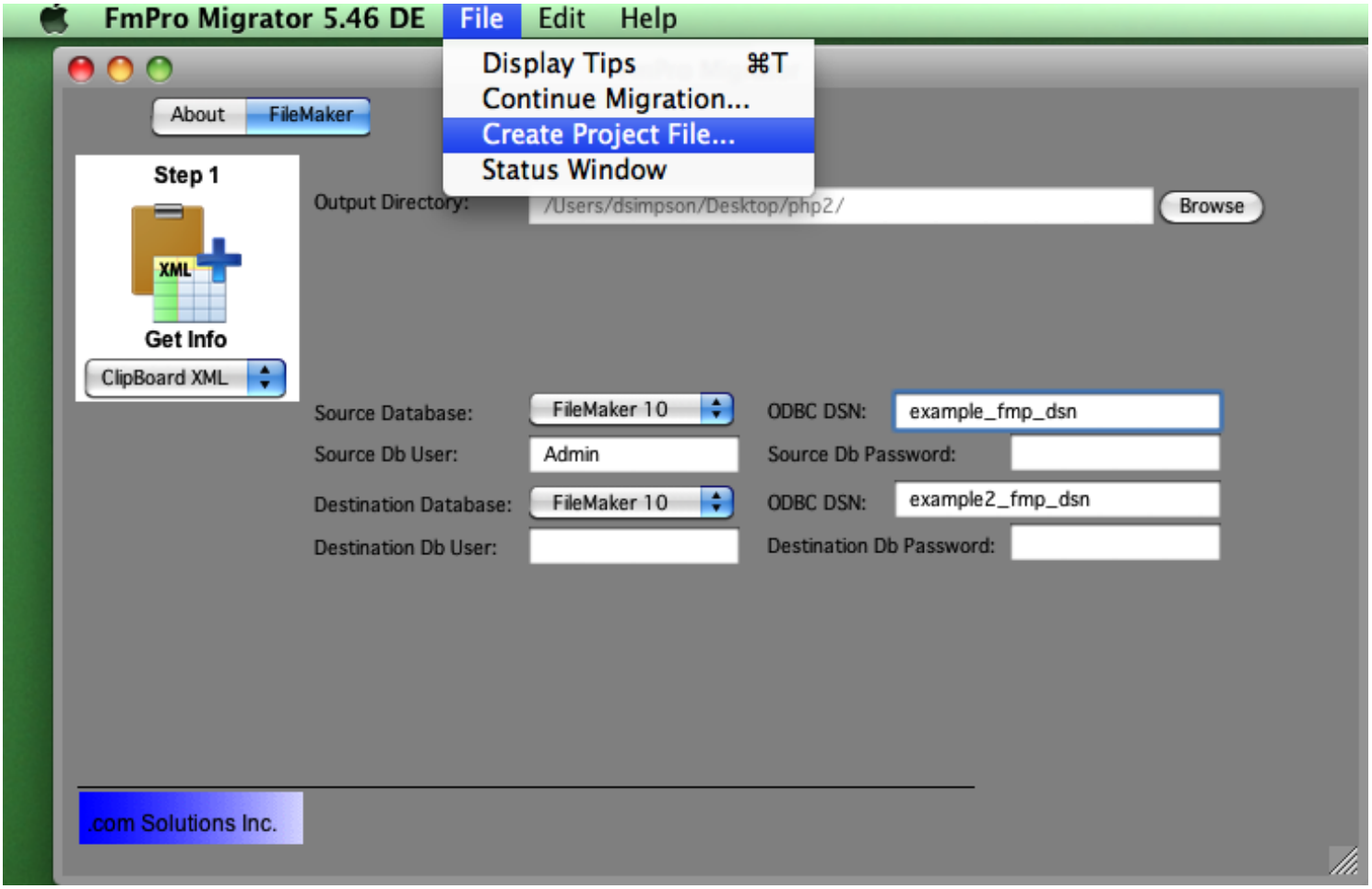

Click the FileMaker tab to select an output directory.

## Click FileMaker Tab



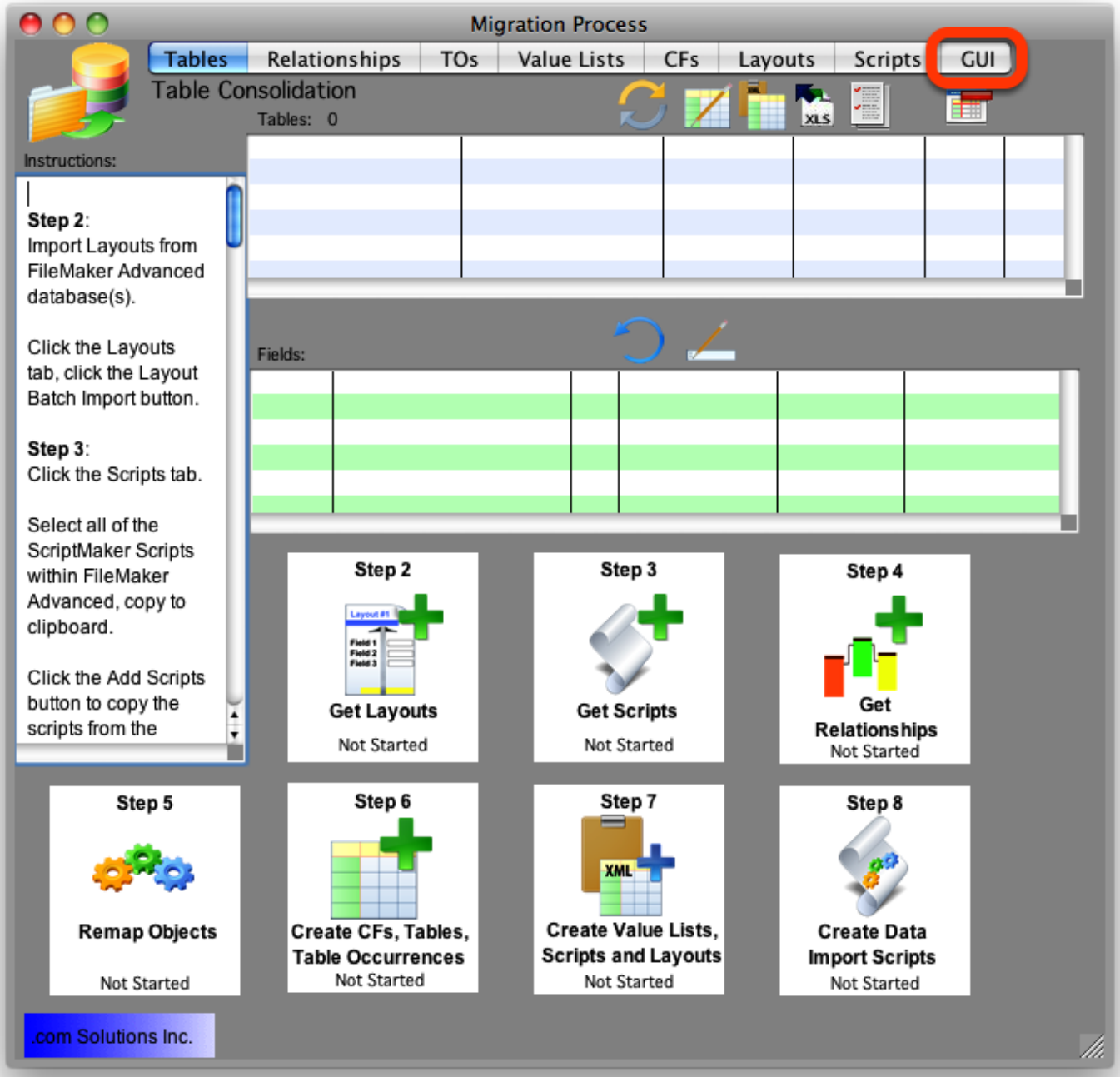

Click the Browse button to select the directory which will be used to store the FmPro Migrator project file. This directory can be the same output directory used for generating the converted scripts or stack file or it can be a different directory.

## Select Create Project File... Menu




Select the Create Project File... item from the File menu. As soon as the FmPro Migrator project file has been created, the Migration Process window will open.

## Click GUI Tab of Migration Process Window



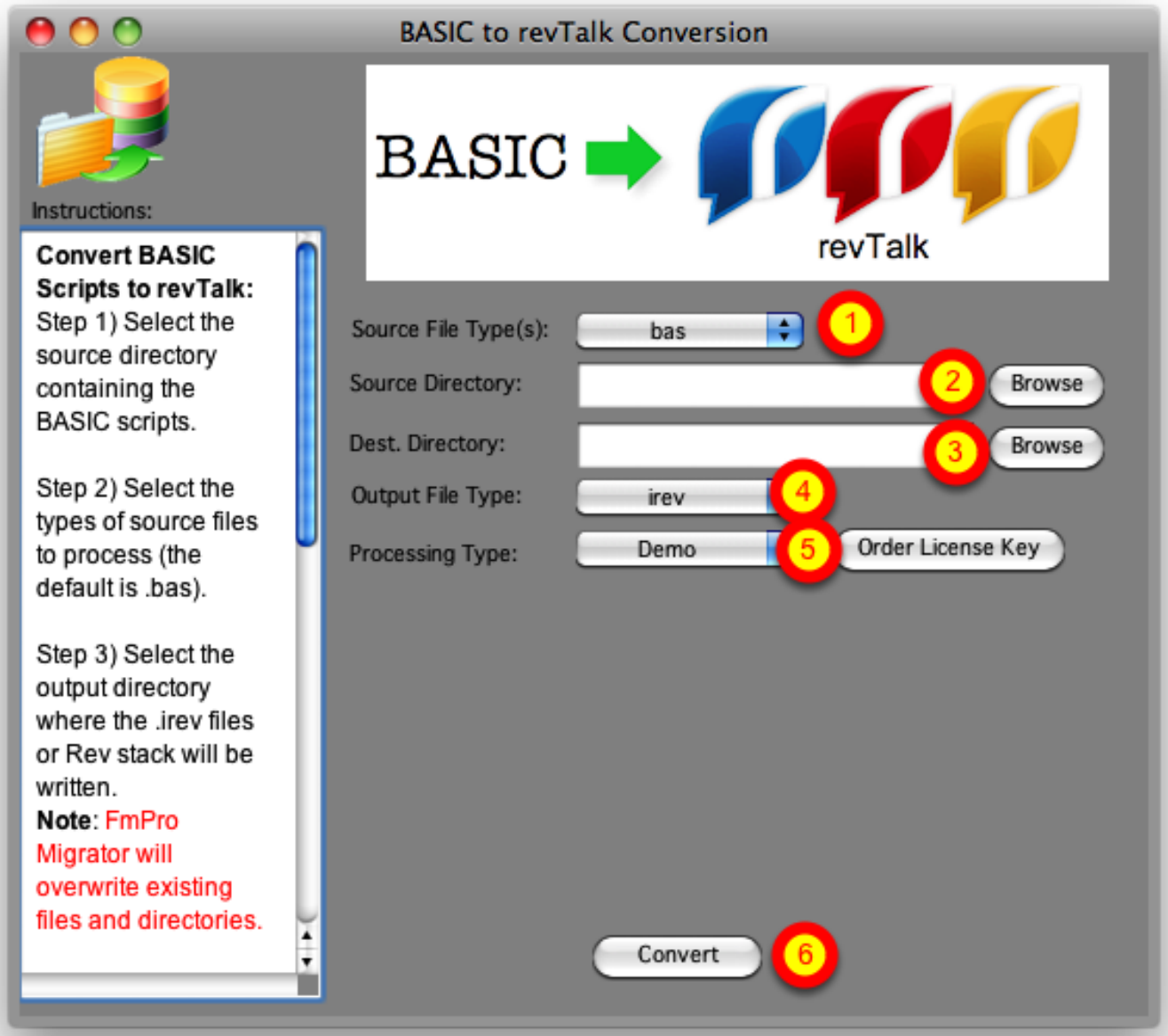

Since a database migration is not being done, ignore the contents of the various database migration features, and click on the GUI tab button.

## Step 2 - Select Conversion Options & Convert BASIC Files

### Click BASIC to revTalk Button

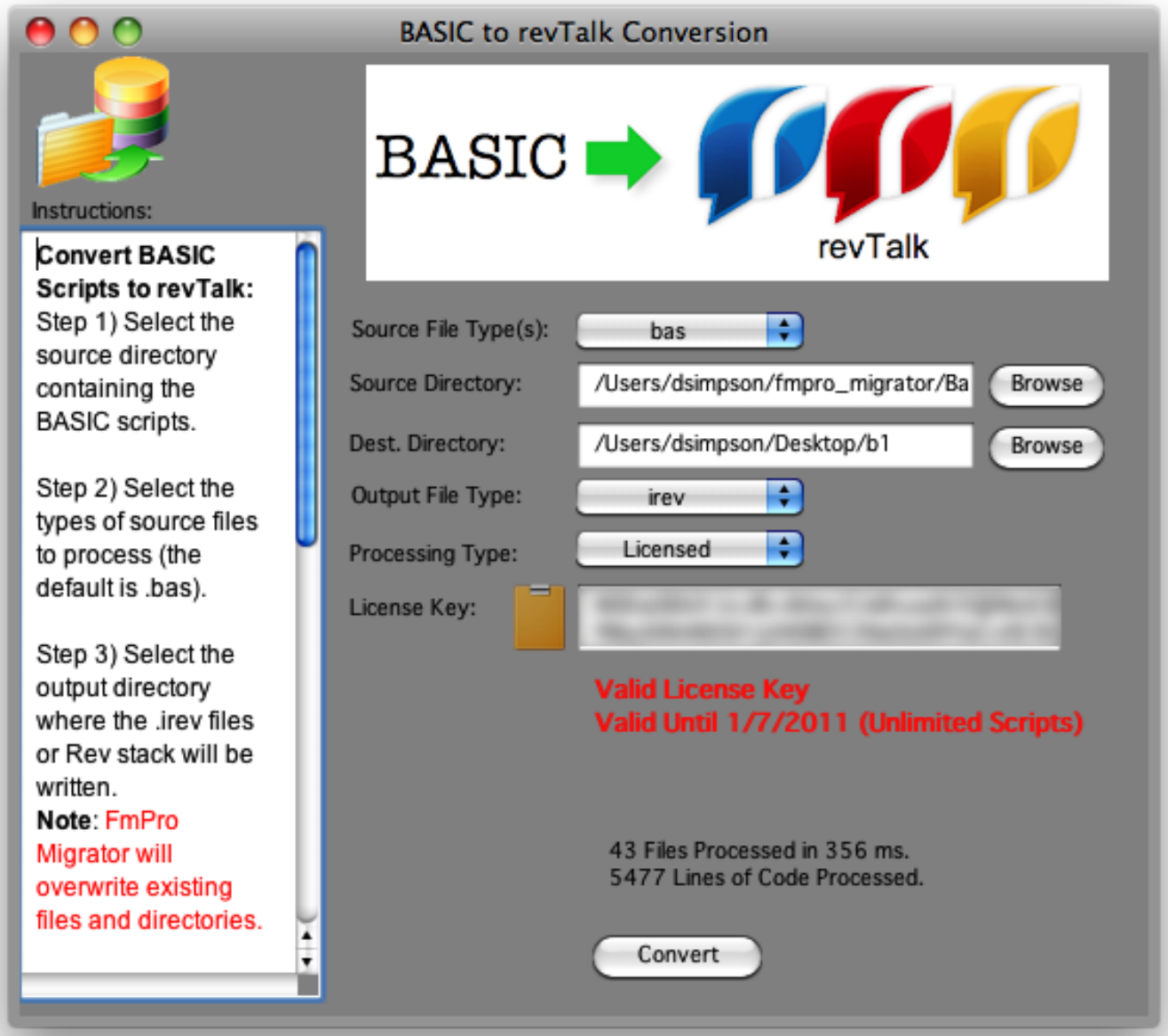

Click the BASIC to revTalk button to open the BASIC to revTalk window.

## BASIC to revTalk Options



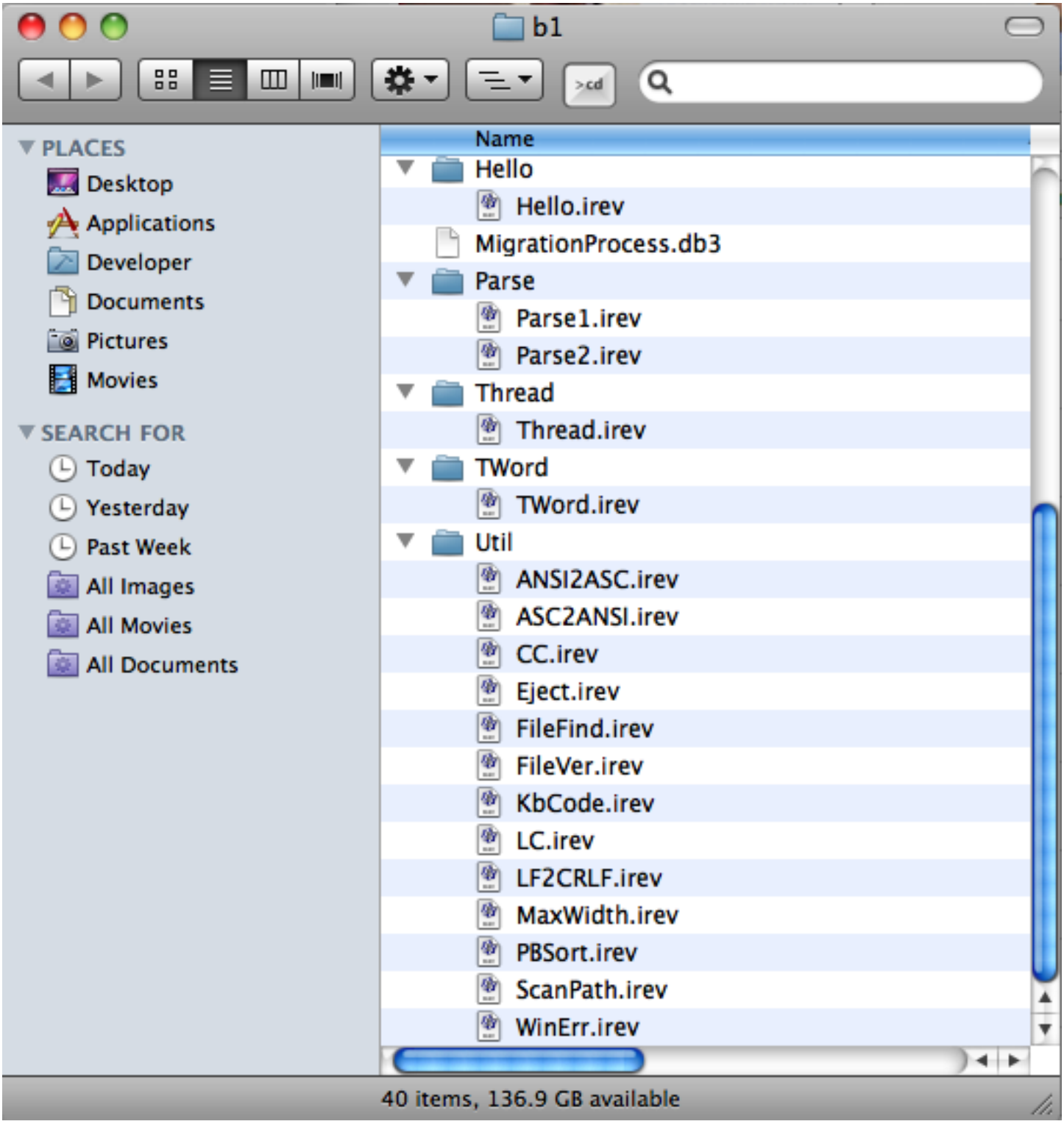

There are several options which need to be set prior to performing a code conversion project:
1) Source File Type(s): bas, vba, vbs or txt.
2) Source Directory - This is the top-level directory containing your BASIC scripts. All enclosed directories will be traversed and files within those directories will also be processed.
3) Destination Directory - This is the output directory where the converted files will be written.
4) Output File Type - The BASIC files can be converted into .irev files or a single Rev stack having a card representing each converted source file.
5) By default, the BASIC to revTalk process operates in Demo mode. In Demo mode, 5 files of unlimited length will be processed. Ordering a license key removes this limitation.
6) Click the Convert button to convert the BASIC files.

## Conversion Results



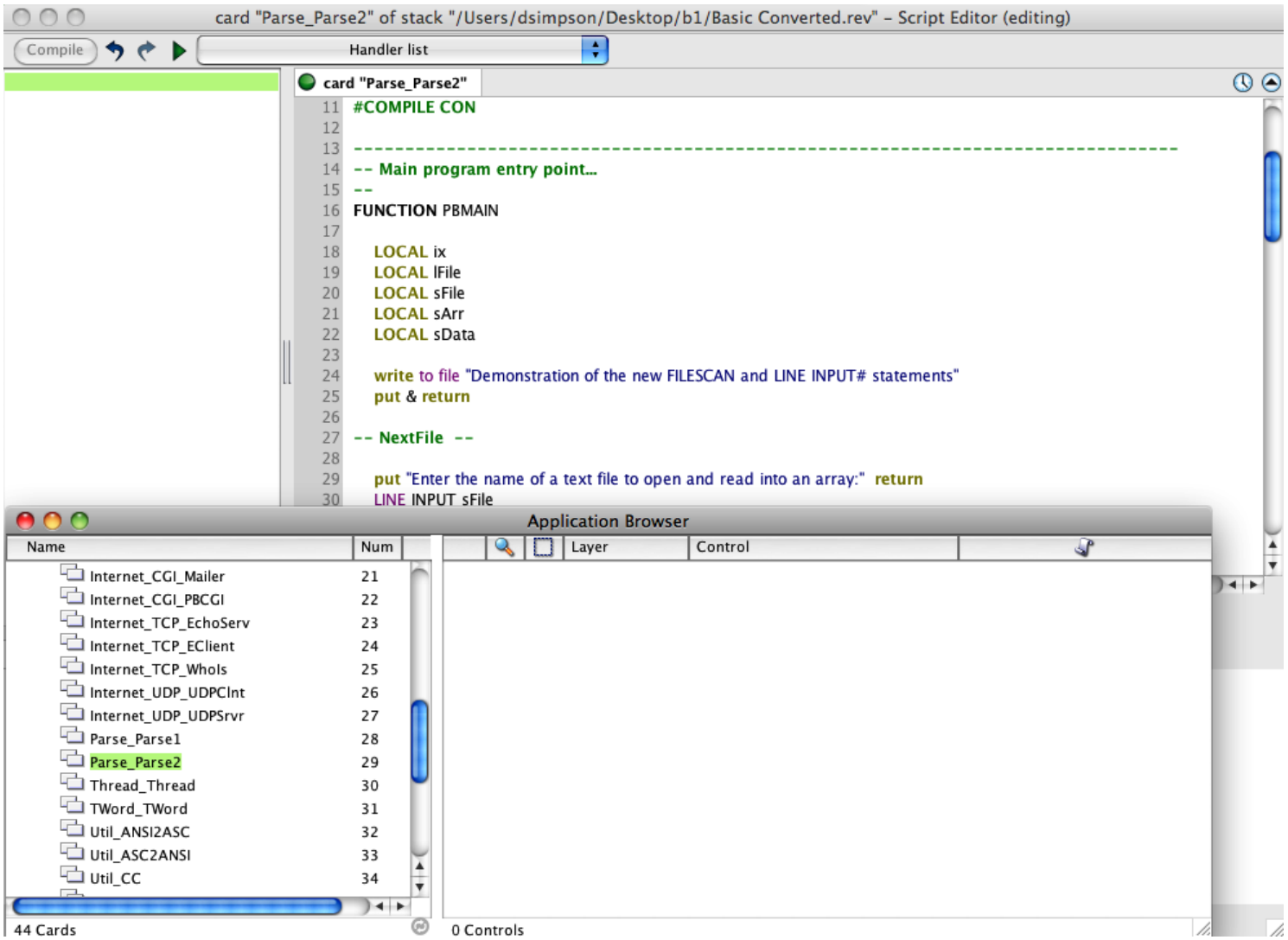

After clicking the Convert button, FmPro Migrator converts each of the files and displays the conversion results.

## .irev Converted Files




The generated .irev files shown in the output directory. When generating .irev files, the <?rev tags are added to each file since each line of the file is considered executable code.

## BASIC Converted.rev Stack with Converted Card Scripts




If the stack output file type is selected, a stack named BASIC Converted.rev will be created within the output directory. Each BASIC script is converted into a card having a name consisting of the subdirectory name and script name. Select any card in the Rev  Application Browser and right-click on the card to select the Edit Script contextual menu item. The script will be opened in the Rev editor.