Volker Krambrich
NORSULT, Platinum Partner from Finland
15. August 2018
This paper is meant to put forward some ideas on how to improve the Data Migration Tool. It is a first draft and not meant to be final!

Filemaker Data Migration Tool

Use Case and Improvement Ideas

I put together some thoughts reflecting all things written in this list and our experience with the tool. Last week in Dallas we sat together with Clay Maeckel, author of the tool, and my colleagues over a long lunch. We discussed various ideas, questions, problems and came up with many new ideas and possibilities for further improvements. One outcome was that I sit down and write this paper. It should help to guide the tool's developers to better understand our needs and wishes. I try to put up a coherent story around a use case that reflects the discussions and feed back on feature ideas from the community.

The roll of the FMDMT

The Datamigrationtool released with Version 17 of the FileMaker Platform is the first ever possibility provided by the manufacturer to securely and fast update a version of the FileMaker file with an enhanced or in any way further developed version of that file in respect of its features (vs. its data only). It allows for the first time to easily move a files' data

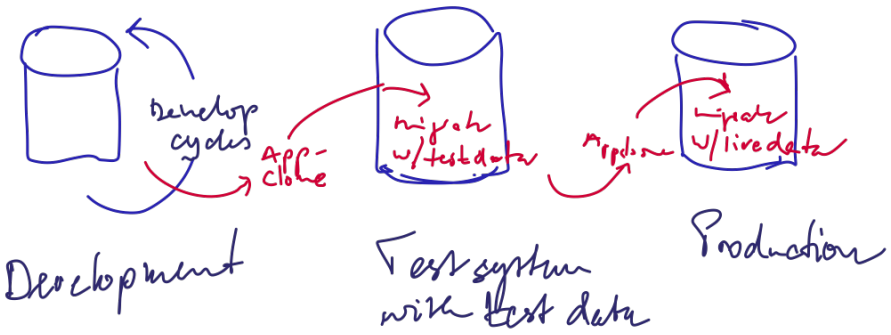

FIG. 1 PROCESS OVERVIEW

into to a newer or changed version. Its release coincides with the changed license model that allows users at least three server installations with any server license (DTAP model of development), or more generally spoken, it allows to keep a known state of a production untouched while development is made on a copy of that file without the overhead of

providing a sofisticated data import routine or any other handmade method to move the newer development into production.

Use Case

I use a common and sophisticated use case, not just a single file update. Most systems consist of two or even more files that in conjunction represent a FileMaker App. Updating the app then means to update all related files, understood as an update transaction. That is, either all files are migrated successfully or the state of the system shall be as before; if possible an error report shall be created automatically. In order to make best use of the Develop – Test – Approve & Production model of development (DTAP), the procedure to move an app from state to state should be both easy and easy to automat. Only then will it speed development and avoids the 'fast patches' on a live production system.

I take a look into the way how to achieve that goal logically and what parts are difficult or even missing with the tool(s) at hand. Another look shall then be taken on the elements that constitute 'data' in the tools' sense.

I do assume that we have an app on a FileMaker server, installed on a machine matching at least the minimum requirements for a server and for the tool[1].

I do consider even the practice to keep the system, the data and backup files on separate volumes. Further I want to support a possible separation of the role of one or several data managers apart from the role of the database manager[2].

It might thus be so that one data manager provides the database manager with the necessary update material from development (i.e. clone files, credentials, toolfiles). But the process of data migration is not executed by the data manager himself.

To execute a data migration the database (server) manager needs the tools and metadata involved. Specifically the safety and privacy of the managed data is concerned in this case. That includes obeying to license rules[3] as well as access credentials for the files. It should include everything for the process control and the reporting back to the data manager. In particular the material contains a command file, the update material (set of clone files, set of

---

[1] That explicitly requires to be running on a 64 Bit system version!

[2] In many cases I found that the app owner is in the role of the data manager, i.e. the person or team responsible for the content and logical consistency of the data in the app and the app development, while the server is operated and maintained by an IT department.

[3] FileMaker licenses the FMDMT and other command line tools to FBA and Community Subscribers only. The tools may thus not be stored on machines not owned by the licensee. They may be used anyway in processes controlled by the licensee exclusively.

additional data files, set of tool files as executables) where the command file references or calls the other files. The command file itself is executed from the fmmigration directory.

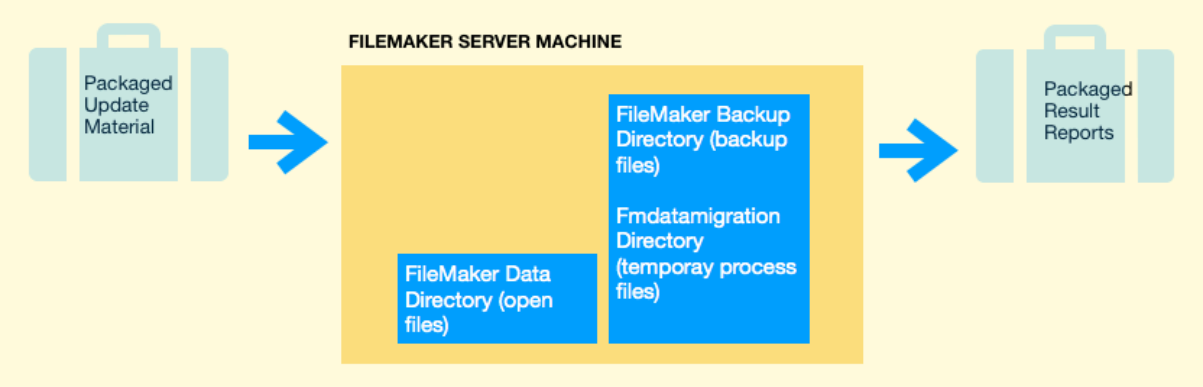

FIG. 1 PROCESS OVERVIEW

The abstract of the command file is expressed in the following meta code:

```
1. Set up and extract the update material
2. Check for the presence of every source file referenced
3. Check for the presence of a clone file for every source file referenced
4. Close every source file referenced on the FM server
5. Copy every source file referenced from Data to ProcessDirectory
6. Process every source file referenced securely
   6.1. create a process log
   6.2. create a result file (migrated source)
   6.3. perform additional operations on the result file
   6.4. provide a result code/information
7. Put the migrated version of every source file back to Server Data
8. Open every migrated file on the FM Server
```

Only the steps 6.1 and 6.2 of that command procedure are handled by the current FMDMT V. 17.0.2! It is understood that failure of one step leads to a roll back of the process, that means specifically to re-open eventually closed files on the FM Server after step 4.

We found several issues and ideas for the existing tool and also ideas for other command line tools to enhance the process for a smoother and more complete result. I will now look into more detail with the issues and ideas stepwise.

Issues, Ideas, Comments

Analyzer Mode

One of the most important issues and wishes with the current tool was the unpredictability of the time the migration might take until after the migration was run. The time depends on two main factors: the amount of raw data involved and the method the tool uses for a given data table (block mode vs. record mode). Chosen migration options might additionally influence the performance and choice of method. Thus the resulting feature idea

is a mode for the tool to just do the opening analysis where these choices are determined and just report back that result.

```
FMDataMigration -src_path <path> -clone_path <path> […] -analyze
```

The result would be a text output exactly as is, it can be piped into a text file, without the actual data migration taking place.

This mode would require the use of a [full access] account to the data files to not compromise any private data[4]. If used with the safe mode based on the extended privilege prefix 'fmmigration' a shortened output could be constructed like: "Migrating all/some/no tables in slower record mode".

Step 1. Preparations

How to setup and move files into directories and the syntax of the command files depends on the underlying operating system. System tool should be used to build the process environment for data migration. After these preparations we have at least a directory apart from FileMaker server Data with the following or similar structure and content

```
fmdatamigration
      <theCommandFileToBeExecuted.cmd>
      tools:
            <fmdatamigration_executable>
            <any_other_commandline_tool>
      sources:
            <sourcefilename>
      clones:
            <sourcefilename_clone>
      results:
      addondata:
            <fmdatamigration_datafile1>
            <fmdatamigration_script1.xml>
```

Steps 2. & 3. Check entry conditions

This is quite self explanatory. See if the required data sources are found on the system. In case of a running server they all should be in the reported files from the `fmsadmin` tool. Check if the respective clone files where exported to the <clones> directory.

Step 4. Close Files on Server

In order to make the migration transactional, no changes to the data files are allowed during the process. Taking a backup is thus not enough. The active files shall be closed to allow the copy. Use cli commands or the admin api to close the files and crosscheck the state after the commands.

---

[4] The disclosure even of a table name ("EmployeesToBeFired") could be such a break.

Step 5. Copy Sources

Use system syntax to copy the current source files from the FileMaker `<Data>` directory to `fmmigration/sources/` . Even large files copy relatively fast on system level to another volume. A copy is needed to not damage the active files accidentially. Furthermore the copies are needed for a roll-back, in case any errors occur after the migration and a return has to be made to the exact point (of file schema) before the data migration.

Step 6. Loop through Data Migration

This is where the current tool operates and where we need enhancements.

Let us start with the process log file. Depending on the operational mode, either using a [full access] account on both source and clone, or using the secure mode based on the fmmigration extended privilege. In the later case only a compacted log is written to ensure data privacy.

Suggestions:

Output an encrypted complete logfile when the `-verbose` option is chosen. The file encryption secret could be the same secret that is used with the fmmigration extended privilege. File format and encryption method need to be defined.

This would give the data manager the possibilty to check for detailed results when this file is put back in the Packaged Result Reports (see fig. 2.)

Signal Process start and end. It would be good to know if the process terminated abnormally, since no result code is issued from the tool, we discussed a method employing semaphore files. The first action of the fmdmt would be to create a file in the `results:` with the PID as part of the name. The last action after data migration and writing the output file would be to delete the PID file. As long as this file exists, the process is either running or terminated abnormally.

Perform operations in the file. If a comand line tool could launch a filemaker server in a protected single user mode without a UI, we could perform preprocessing or postprocessing by running scripts. These scripts could be in the file or given as a parameter path to the xml representation of the script to run. In the case of the xml script, the code would never be part of the resulting FileMaker file but only executed once. XML encoded scripts can access the data files in the `addondata:` directory. Compatible script steps should not require any user interface or user interaction; furthermore they can only operate on data stored locally in the file. This excludes also any reference to externally stored container contents. It could be used

to move data, re-calculate data, or even add data in new or existing table. Setting of Globally stored values is one more option.

Step 7. Move Migrated Versions to Server Data - or - RollBack

If any occurred during the processing so far, write back an error log and re-open the files that where closed in Step 4. only! This is one reason why those files should be kept intact and untouched during the process.

Otherwise copy and overwrite the new files from the results: directory to the original FileMaker Server Data directory.

Step 8. Open Migrated Files on Server

Finally use CLI commands or the admin api to open the migrated files files and crosscheck the state after the commands.

(**vk**)

Outlook on „Structural Data“ and other Implications on Data when using the FMDMT to come…