



Upgrading to FileMaker 8:

Migration Foundations
and Methodologies

About This Technology Brief

It is the intent of this technology brief to help the experienced FileMaker® developer better understand and prepare for migration to the FileMaker 8 product family. Reading this document will assist you in planning, preparing for, and executing a strategic approach to migration.

Originally compiled, edited and principally authored by Danny Mack, President of New Millennium Communications, Inc., a FileMaker Solutions Alliance Partner, this paper is part of a series of technology briefs written by developers for developers, to assist in migrating existing solutions to the new FileMaker 8 product family.

For additional technical materials, please refer to printed and electronic manuals and online help that ship with FileMaker Pro 8, FileMaker Pro 8 Advanced, FileMaker Server 8, and FileMaker Server 8 Advanced.

Please Note: This technical brief is relevant to both FileMaker 8 and FileMaker 7 products.

Table of Contents

I. Overview	5
II. Foundations	
• Leveraging the Value Proposition of FileMaker Pro 8.....	9
• The FileMaker 8 Relational Model.....	20
• File References in FileMaker Pro 8.....	41
• Scripting Issues Encountered When Migrating to FileMaker Pro 8.....	56
• Security and Access Privilege Issues.....	63
• “Record Ownership” in Converted Solutions: Opening and Committing Records	73
• Migration and Web Publishing.....	85
III. Methodologies	
• Conversion Basics	89
• Adding a New Interface File to an Existing Solution, Later Consolidating Tables	99
• Case Study: Migrating Using the Hub & Spoke Approach	107
• The Separation Model: A FileMaker Pro 8 Development Model	112
• Bridging .fp5 and .fp7	128
IV. Appendix	
• Conversion Issues & Resolutions	A1

Foreword

This document assumes that the reader is already familiar with the basic features of FileMaker Pro 8, and has read the Tech Brief entitled “Upgrading to FileMaker Pro 8 – Migrating Existing Solutions.”



The information in this compendium of articles is the result of a hard-working team of FileMaker, Inc. employees and independent developers who have designed, explored, and tested FileMaker Pro 8 and FileMaker Server 8. It represents an evolving body of knowledge, one that will mature over the next months and years as the application is mastered by professional FileMaker developers building solutions.

It has been a privilege to work with this team.

Danny Mack
Boulder, Colorado
March 2004

© 2005 FileMaker, Inc. All Rights Reserved. FileMaker is a trademark of FileMaker, Inc., registered in the U.S. and other countries, and the file folder logo is a trademark of FileMaker, Inc. All other trademarks are the property of their respective owners. Mention of third party products and companies is for informational purposes only and does not constitute an endorsement nor recommendation. Product specifications and availability are subject to change without notices. (DocV3)

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, AND FILEMAKER, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR THE WARRANTY OF NON-INFRINGEMENT. IN NO EVENT SHALL FILEMAKER, INC. OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS OF BUSINESS PROFITS, PUNITIVE OR SPECIAL DAMAGES, EVEN IF FILEMAKER, INC. OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY. FILEMAKER MAY MAKE CHANGES TO THIS DOCUMENT AT ANY TIME WITHOUT NOTICE. THIS DOCUMENT MAY BE OUT OF DATE AND FILEMAKER MAKES NO COMMITMENT TO UPDATE THIS INFORMATION.



Overview

Brief Synopses of the Sections of this Document

Overview

Foundations

Leveraging the Value Proposition of FileMaker Pro 8 – Michael Thompson

Michael discusses the advantages of FileMaker Pro 8 from the point of view of both developers and business owners, and addresses the strategic issues associated with migrating to this new technology, including how to recognize a return on the investment in migration.

The FileMaker 8 Relational Model – FileMaker, Inc. and Danny Mack

A fundamental article of the new relational model, including a comparison to earlier versions of FileMaker Pro, an introduction to the relationships graph, and an explanation of “context”. It illustrates the new features and the new rules with easy to follow examples.

File References in FileMaker Pro 8 – Corn Walker

File references are viewable and editable in FileMaker Pro 8 and the issues associated with them are fundamental to getting your converted solution up and running successfully.

Scripting Issues Encountered When Migrating to FileMaker Pro 8 – Darren Terry

In converted solutions, certain scripts may not function as they did previously. Darren reviews these issues in detail so that you understand how the behavior has changed, and what to do about it.

Security and Access Privilege Issues – Steven Blackwell

Passwords and groups are converted to accounts and privilege sets, but the rules have changed. Understanding the details of access privilege conversion is essential to replicating the original behavior or extending it.

“Record Ownership” in Converted Solutions: Opening and Committing Records – Ilyse Kazar

The script steps and other events that cause records to be opened (locked) and committed (unlocked) have changed significantly. This has major implications for the behavior of converted solutions as well as for new solution design.

Migration and Web Publishing – Bob Bowers

This is a primer on the realm of FileMaker 8 web publishing, including both Custom Web Publishing and the entirely new Instant Web Publishing.

Methodologies

Conversion Basics – Danny Mack

Danny presents a step-by-step orientation to the conversion process, including preparation, testing, and the necessary fundamental tasks for getting your solution “restored” to its original functionality.



Adding a New Interface File to an Existing Solution, Later Consolidating Tables – Todd Geist

How does one migrate a solution, most reliably and economically, to an optimal FileMaker Pro 8 architecture? Todd lays out a step-by-step migration strategy, including a rationale that may leave you thinking: “why would you do it any other way?”

Case Study: Migrating Using the Hub & Spoke Approach – Molly Connolly & Bob Bowers

In many solutions, the majority of the interface and logic is concentrated in one file, the “hub”, or in a few hubs. It may make sense to consolidate the “spoke” files into the “hub” files.

The Separation Model: A FileMaker Pro 8 Development Model – Colleen Hammersley & Wendy King

FileMaker Pro 8 is a dream come true for those who have long advocated the separation of data and interface. Colleen and Wendy present the new application model in all its glory.

Bridging .fp5 and .fp7 – Ernest Koe

There are many real world scenarios in that it will be necessary for FileMaker Pro 8 files to exchange data with FileMaker .fp5 files. There are several technologies that make this possible.

Appendix**Conversion Issues & Resolutions – Team**

This is the documentation of specific issues that can be encountered in solutions that have been converted to FileMaker Pro 8, and suggested resolutions (what you can do to replicate the original behavior). The issues are cross-referenced to the behavior changes in the application documented in the .pdf entitled “FM 7 Converting Databases”.



Overview

Learn FileMaker Pro 8 Before Converting a Complex Solution

As has been stated elsewhere, starting to learn FileMaker Pro 8 by converting a complex solution may be confusing. It is highly advisable to build some mastery of the new application by first building a new solution.

- Enjoy the ease of creating multiple tables in one file
- Explore the efficiency of the new scripting possibilities, especially script parameters
- Understand the relationships graph and the importance of managing the context from which your calculations and scripts will evaluate
- Discover the power of the new security model
- Learn your options for solution architecture – with your interface in the same file as your data tables, or in an entirely separate file with no tables

Once you have a reasonably good understanding of the new environment, then it becomes appropriate to consider the approach to take to migrating an existing solution.

A Strategic Approach to the Migration Process

As outlined in the Tech Brief, “Upgrading to FileMaker Pro 8: Migrating Existing Solutions”, there are several possible approaches to take to migration. Hybrid approaches are possible where one method is used for one part of a solution while another method is used for another part.

Here is a recap of the approaches, as discussed in the Tech Brief.

1. Convert and deploy, then modify if necessary
2. Modify, convert, test, & modify
3. Add a new interface file, and then later consolidate tables
4. Consolidate multiple files into a single file, starting with an existing file
5. Consolidate multiple files into a single file, starting from scratch
6. Create new files – an interface file and a data file, or multiple interface and data files
7. Co-existence of .fp5 and .fp7 files

Each of these approaches could be considered separately. Nevertheless, they can also occur as part of a sequential process. Some of the stages or aspects of the sequence may not apply to your situation, but there will be situations in which each of these approaches is relevant. Also, for some developers who support multiple solutions, one approach may apply to one solution while a different approach may apply to another.



1. Convert and Deploy, Modify if Necessary

If you have a relatively simple solution or a solution that is not business-critical, then you may be able to convert and simply deploy your solution without modification. If some features are broken or there are minor data problems, you can fix them as you go.

For more complex solutions, if you try to just convert and run, you may fairly quickly realize that it is not the ideal approach and start again with a more methodical approach. This document is intended to enable you to take a sequential approach to a successful migration of your solution.

2. Convert and Restore to Original Functionality (Modify, Convert, Test, & Modify)

Building on the information in the “Migrating Existing Solutions” Tech Brief, and in the “FM 7 Converting Databases” .pdf that comes with FileMaker Pro 8 and FileMaker Pro 8 Advanced, you can systematically prepare files for conversion and follow an iterative approach. Making innocuous (or even beneficial) changes to your solution in FileMaker Pro 6, and then converting, testing, modifying in .fp5 again, re-converting, and testing, until you are satisfied that you have done as much as is practical and efficient in the .fp5 format. You can then do the remaining tasks in the new .fp7 files, test, and deploy. See the section on “*Conversion Basics*” for more detailed information. The foundational information in the sections on *File References in FileMaker Pro 8*, *Scripting Issues Encountered When Migrating to FileMaker Pro 8*, *Security and Access Privilege Issues*, and “*Record Ownership*” in *Converted Solutions: Opening and Committing Records*, are all essential reading.

There is a finite set of issues that will need manual attention to enable converted solutions to operate similarly to the way they did in FileMaker Pro 6. Not all of these issues will affect all solutions.

Just as important as the process of manual modification, is the methodology used for testing your solution. Depending on the scale of a solution, it may or may not be feasible to test all permutations of the features of a solution. Scripts may work reliably if performed in a certain sequence, but not if performed in a different sequence. It will be most efficient and reliable if you, the developer, understand the issues that need to be addressed and then systematically and completely evaluate, and modify if necessary, all instances of an issue in your solution. Analysis tools will make it practical to do this.

Some tools are available now for automating the modification of files, and more tools will emerge.

3. Add a New Interface File and then Consolidate Tables

One of the key advantages of FileMaker Pro 8 is the ability to see and manage your database structure (or at least large sections of it at a time), in a graphical interface. Your multi-table business processes can be represented visually and, thanks to many enhancements to the available tools, managed efficiently and reliably.



A new interface file can be added to an existing solution, “pointing” at the data tables in the old solution, thereby providing a functional and appropriate environment for enhancing your solution, leveraging the power and optimal solution architecture of FileMaker Pro 8. You can later consolidate your data tables into one file (or a few files) and re-point your interface file. This method has the added benefit of not breaking the existing functionality of converted solutions while allowing the developer to explore the power of FileMaker Pro 8 in a separate file. See the section on “*Adding a New Interface File to an Existing Solution, Later Consolidating Tables*” for a detailed treatment of this method.

4. & 5. Consolidating Files

If you are absolutely sure that you will be consolidating the data tables and can afford to wait until you have revised both the table structure and the interface before using the new solution, then you may want to start by building a new file with multiple tables, but be warned that for a complex solution it may not be very efficient unless you have already mastered FileMaker Pro 8.

There are advantages to having fewer files, for simplicity and manageability. You may want to optimize your calculations or the way that you access related data. Depending on your solution, you may want to consolidate your data tables and fields into a single file or into a few files, either by starting from a single existing file, or by starting from scratch, bringing over just the data fields, and selectively bringing over other fields, then copying layouts and importing scripts, or writing new scripts.

There are a number of technical details that are necessary to understand to achieve this migration strategy successfully. There is good information about these issues in the sections on “*Adding a New Interface File to an Existing Solution, Later Consolidating Tables*” and in “*Case Study: Migrating Using the Hub & Spoke Approach*.”

6. Create a New Solution (Entirely Rewrite)

Starting with new files is most appropriate for a new business problem or an isolated information domain. It could be appropriate if you want to take full advantage of the new FileMaker Pro 8 possibilities and you can afford to wait to complete your redesigned solution before deploying. It is also appropriate if the features of the original solution are obsolete or poorly designed in the first place. The foremost requirement when designing a new solution is to determine your solution architecture. The articles on *The FileMaker 8 Relational Model* and *The Separation Model: A FileMaker Pro 8 Development Model* are both essential reading.

7. FileMaker Pro 8 Co-existing with Earlier Versions of FileMaker Pro

In organizations with a large number of FileMaker Pro files and multiple solutions (sets of files), it may be necessary to convert some files before others. If there are dependencies between solutions in different departments or business domains, it may be necessary to employ a technology that enables the exchange of data between FileMaker Pro 8 files and FileMaker Pro 5/5.5/6 files. See the document on “*Bridging .fp5 and .fp7*” for more information.



Database Design and Evolution

A primary idea expressed throughout this document is that it may be cost-effective and most practical to find multiple points in the migration process when your solution can be stable and functional. This will provide benefits to your organization if you are an in-house developer, and will allow you to more quickly take advantage of FileMaker Pro 8 if you are an independent custom or commercial solution developer.

When re-writing a solution from scratch, there will still be a conversion component to the process. At a minimum, the old solution will most likely be converted to FileMaker Pro 8 prior to importing data into the new solution. It is possible to export data from an .fp3 or .fp5 file to a text file and then to import it, but that would not handle the contents of container fields.

Most FileMaker Pro solutions have evolved over time and have been enhanced based on wisdom gained along the way – both from the insights of the developer and based on feedback from users. Developers' technical knowledge increases over time and that is often reflected in the incremental enhancements to the features of a solution.

The evolution of FileMaker Pro 8 solutions may be similar. In fact, the expertise and knowledge base which exists about FileMaker Pro 8 is less mature, and it will take a while for best practices and true expertise to develop, so an incremental approach may prove to be the most reliable way to minimize what will prove in the long run to be novice errors.

Compounding the advantages of taking this approach is that needs change over time, so that by the time you finish a grand design the priorities may have changed. True business value is often realized by an incremental strategy to developing new systems.

About the author

Danny Mack is the President of New Millennium Communications, Inc., a FileMaker Solutions Alliance Partner based in Boulder, Colorado. New Millennium specializes in FileMaker Pro consulting and solution development, and is the publisher of numerous plug-ins and tools that facilitate the work of FileMaker Pro developers, available at <http://www.newmillennium.com>.



Foundations

Leveraging the Value Proposition of FileMaker Pro 8

FileMaker® Pro 8 represents a new era for the FileMaker Pro family of products. With renewed power and flexibility, the product is poised to emerge as a market-leading rapid application development environment—applicable to all sizes and types of organizations. More than ever, FileMaker Pro 8 allows users with a broad range of expertise to contribute significantly to database solutions that produce tangible value. Whether the objective is to create a professional commercial software application or a desktop database that can be extended to the LAN, the WAN, and beyond, FileMaker Pro 8 certainly meets the need.

From a developer's perspective, this new version offers tremendous advantages over its predecessors and over other software development tools. It is important to remember, however, that such advantages are useful only to the extent that they create value in a particular context. Without a clear understanding of the value proposition, the FileMaker Pro 8 technical distinctions may be lost in the din of technology offerings, which too often over-promise and under-deliver. As FileMaker Pro is commonly used in the context of a business or professional organization, the developer and business manager are well advised to understand the product's distinct value proposition and learn how best to leverage that proposition within their particular environment.

What is it about FileMaker Pro 8 that compels its use by organizations' subject matter experts (SME's) and developers? While maintaining its historic ease of use, the FileMaker Pro 8 family of product offers an entirely new world of innovative and powerful features unlike any other database or software development environment. Organizations adopting the new tools will see immediate gains from increased file size limits, virtually unlimited number of tables, direct server access, server side processing, robust security, and faster wide area network (WAN) access. While protecting their intellectual property and helping enhance data confidentiality and integrity, developers can now provide end users or administrators with far more flexibility and many more options for managing critical business processes. Integrating solutions with existing IS/IT assets is much easier and more flexible. With powerful new features such as Custom Functions and Extended Privileges, developers can implement business logic and business rules across a solution simply by clicking a checkbox. And they can more easily collaborate with other developers and with subject matter experts without the inefficiencies associated with the interruption of workflow. FileMaker Pro 8, FileMaker Server 8 and FileMaker Server 8 Advanced together make for a compelling best-case business decision as a development and deployment environment.

While it is clear that the new product line has a lot to offer almost any organizational environment, its combination of ease-of-use, rapid application development, and enterprise level power must be used appropriately if its full potential is to be realized. The promise of new features and functions does not remove the fundamental requirements of the software development process nor does it insulate the organizational leadership, management and other stakeholders from being materially involved throughout. Focused too narrowly on the technology, it easy to lose sight of the broader organizational objectives. Indeed, if the needs of the organization and market do not lead the process, then much of the value and potential of the technology will be sacrificed.

Whether a subject matter expert turned database designer or a seasoned veteran designs a solution, it is up to the developer to manage the value delivery process and to create and maintain an environment that is conducive to realistic expectations. Furthermore, a solution for today's problem with little tolerance for adapting to tomorrow's needs, or one that is difficult to use or maintain, quickly loses much of its potential value. This means



that delivery of value must be considered throughout the lifecycle of a project, not simply at solution delivery and initial implementation. Fortunately, FileMaker Pro 8 is an exceptional development environment for providing value throughout the lifecycle of a project. Understanding how to get the most out of the technology at each stage of the process is essential if a solution is to realize its full potential. The intention and complexity of a solution, as well as the environment in which it will be used, will determine where to look to leverage the value of the new features of FileMaker Pro 8.

Requirements Gathering: Solving the Right Problem

Organizational constraints are often revealed through secondary symptoms, and in many cases root causes of problems are not immediately apparent. Furthermore, the symptoms, as well as their underlying problems, rarely exist in isolation. Invariably other systems can affect and are affected by newly implemented solutions. Finally, the problems we address today will almost certainly reveal further constraints and produce unintended consequences. This is why understanding the problems and their contexts is one of the first steps in creating solutions.

Traditionally, software developers have relied upon requirements gathering and needs assessment processes to define the objectives and to understand the often-complex relationships involved in designing an effective solution. Through interviews with end users and business managers, they produce a written specification that helps the developer and the stakeholders understand and communicate about the project.

In most cases, input from users is gathered through interviews and then translated into text. While this process can be relatively effective, it assumes that the users are capable of describing their needs in a manner that is both consistent with the real needs of the project and comprehensible (in all of its detail and nuance) to the interviewer. It also assumes that both the interviewer and user understand the environment in which the problem and solution exist well enough to provide an accurate and useful translation. This can be especially challenging when the developer is unfamiliar with the intricacies of the environment. While the use of drawings and mock-ups can narrow this communication gap, even with visual aids, the needs gathering process can involve several iterations of design and feedback in order to accurately assess the details.

FileMaker Pro 8 offers powerful capabilities to augment and assist in the requirements gathering and needs defining process. FileMaker Pro's ease of use allows even non-technical users the ability to participate more substantially in the defining stage of the process. By using FileMaker Pro as the environment for creating the visual aids and training users on the use of basic layout tools, those most familiar with the organizational processes can explore and experiment in real time. Adding and moving layout elements on workflow interfaces and looking at data in different ways can accelerate the process through which understanding of the significant issues is developed. More importantly, having the subject matter experts involved at this level of exploration often reveals unrecognized needs and related issues that would otherwise be revealed much later in the development cycle.

While much of this capability has existed in prior versions of FileMaker Pro, FileMaker Pro 8 promises much better performance over the wide area network. This means that subject matter experts and developers can



work together from different locations without the use of third-party remote access software. Changes made by the developer on a served set of files will appear to the SME immediately upon committing those changes to the server (upon returning to Browse mode from Layout mode) and vice versa. Additionally, many of the new features and capabilities of FileMaker Pro 8 provide improved support for collaboration. Defining new fields, for instance, no longer requires denying other users access to the database. This minimizes the disruption to other users as SME's and developers collaborate to understand and define the requirements of a solution. It also opens up the possibility for multiple teams to be working simultaneously with the same set of files. This real-time development environment offered by FileMaker Pro 8 can dramatically shorten the iterative cycles of communication involved in the requirements gathering process.

Multiple tables in a file and the relationships graph are two other new features available in FileMaker Pro 8 that can enhance the requirements defining process. Developers can explore and demonstrate data architecture without the time consuming hassle of creating individual related files. Multiple tables of data are quickly created in the same define fields session (no need to go back and forth from one file to another to add fields to different tables). Once rough table structures are in place, the relationships graph provides a very efficient means for defining relationships between tables. The fact that relationships are now bidirectional means that there is no need to create reverse relationships; further reducing the time spent in early conceptual stages.

The relationships graph also provides an effective means for communicating the data structure to users and other design team members. Let's face it; complex relational database design does not come naturally to everyone. Yet even database neophytes understand the need at some level for organizing data in a certain manner. A picture paints a thousand words, and providing surface level data requirements in a graphical format can quickly accelerate the understanding of more complex data structure. A well-organized relationships graph provides a visual overview of the architecture, which can then be used to explain it to technical and non-technical stakeholders alike. Expanding on the architecture in real time by creating new table occurrences and relating them to the existing structure is also a powerful form of communication. Remember too that what is created in the relationships graph is not simply a visual aid because the table structure it defines is viable in and of itself. Sample data can be entered or imported into the tables in order to more clearly demonstrate how the structure will actually work.

In prior versions of FileMaker Pro, designing data architecture was something best performed with a flow chart or entity-relationship diagram (ERD). While these will still be invaluable¹, the new features of FileMaker Pro 8 make exploring data architecture more accessible. Ultimately, the design of your solution may require several files, each containing a number of tables. In the early stages of requirements gathering and needs assessment, however, exploring data structure for the purposes of learning and communicating can be accomplished very efficiently in a single file.

Planning: The Art of Creating Expectations

One promise of FileMaker Pro 8 is shorter development cycles. There are many timesaving new features and new ways to design quick, effective solutions. Nonetheless, planning what it will take to successfully complete any software development project is tricky business, and the new and unique nature of FileMaker Pro 8 adds yet another variable to the planning equation.



In the initial use of FileMaker Pro 8, **not all assumptions derived from prior versions or other development environments may apply.** While FileMaker Pro 8 can be used in the same manner as prior versions, to do so would be to ignore its potential. Because FileMaker Pro is unique in many ways from other software development environments, different strategies and techniques will be used to produce intended results. **It is easy to think that the uses of the new FileMaker Pro 8 features are obvious, but effective design methodologies and best practices will emerge only through an experiential understanding of these features and how they relate to old features and other new features.** It will be necessary to explore new possibilities in data structure, navigational methodology, data integrity, transaction management, security, workflow control, and many other aspects of solution creation before acquiring a good handle on the factors that influence the planning process. While eventually FileMaker Pro 8 will become as familiar as earlier versions and other tools, it is best to approach its initial use with caution and respect when it comes to creating and managing expectations for early development projects.

One way to explore the new features and possibilities in FileMaker Pro 8 and learn how to better estimate time and resource requirements is to subdivide more complex solutions into smaller sub-projects. Take the quintessential database solution, the contact manager, for instance. Instead of jumping in with both feet (only to discover late in the game that there is a better way to do most of what is already done), begin with the basics. What does it take to manage a list of names? How will users enter new names and find existing names? What is the best way to manage the navigation between list views and entry views? Once you have a functional and effective name managing solution then add phone numbers. Will each name have a limited or unlimited number of phone numbers associated with it? What is the best data structure to manage this additional entity?

While this example may be overly simplistic and the approach may seem rudimentary and inefficient given the experience of many developers, it is surprising what can be gained through such a step-by-step methodology. In addition many of the new features in FileMaker Pro 8 make an incremental process more manageable. Being able to define multiple tables per file, bi-directional relationships, seeing a working relationship diagram in the relationships graph, the ability to pass parameters from buttons to scripts and from scripts to sub-scripts, all play a part in making an incremental, learn as you go, approach effective.

Another factor that must be addressed in the planning phase is that of development strategy. The FileMaker Pro developer of past versions was often programming most—if not all—of a solution him or herself. Thanks to some of its new features, FileMaker Pro 8 is a much better environment for collaborative development. Multiple developers or development teams can now access all aspects of a served file simultaneously with only minimal interference. There are also more effective methods for multiple developers or teams to work on separate sets of the same files, bringing them together at a later point in time. Finally, FileMaker Pro 8 is a better environment for modular design. One developer or team can be working on the contact management module while the other team works on a separate invoicing module. The two modules can be integrated when the independent modules are ready.

Planning and defining a development strategy is an essential step if appropriate expectations are to be created and maintained throughout the process. FileMaker Pro's power and flexibility, along with its ease of use, provide many timesaving new features that will eventually shorten development cycles. By providing tools



that make incremental and collaborative development more accessible, greater understanding of development methodologies can be acquired throughout the development process and a team or multi-developer approach can be employed to further shorten project completion times.

Design: Where No One Has Gone Before

While the specifics of design in FileMaker Pro 8 are discussed in greater detail elsewhere in this document and in other related materials, it is clear that one of the most significant changes in FileMaker Pro 8 is that of greater design options. Multiple tables per file, removal of the fifty-file limit, increased capacity of text and container fields, increased file size, multiple windows per table, a new security model, and a more flexible and powerful relational structure are just a few of the enhanced features that will change the way FileMaker Pro solutions are designed.

While the technical benefits of this new functionality may be immediately obvious, the nuances that this new power brings to the value equation are easily overlooked. With custom solutions, for instance, the increased granularity of access to the various layers of a FileMaker Pro 8 solution can allow the power-users and subject matter experts to stay connected throughout its lifecycle. Allowing super-users the ability to design entry layouts, reports, and scripted processes that meet their specific needs while restricting access to areas that may compromise the integrity of the solution is now more manageable.

With commercial solutions or solutions that will require 'upgrades' over time, the ability to separate data from structure (both interface and logic) is much easier in FileMaker Pro 8 than in prior versions. The developer needs to only replace the structure files as a part of an upgrade process. Separate structure files that continue to work with the customer's existing data files will eliminate the need for complex and unwieldy import routines.

FileMaker Pro 8 also lends itself to a modular approach to solution design. With this approach, a defined set of functions is contained in a file or set of files, constituting a module. Each module is designed to allow other modules to access its data while safeguarding its functionality. A contact management solution built by one developer, for instance, can be designed to allow an invoicing module developed by another developer to interact with its data. A sales reporting module might access data from both the contact management and invoicing systems. The value of this approach is certainly evident in commercial applications as many developers can leverage their specific expertise and solutions in combination with those of other developers. Furthermore, a developer can allow other developers access to needed information and functionality without allowing access to the entire solution, protecting both the stability of the module as well as the intellectual property.

Other development environments can also benefit from a modular approach. With a well-developed set of design principles, multiple SME's and developers within an organization can develop modules that deliver specific functionality, without compromising other development efforts or 'live' systems. A mix of commercial modules with custom modules or add-ons might be the best way to deliver value when meeting an organization's needs.

These design methodologies and others are covered more extensively elsewhere in this and other documents,



and new design methodologies may emerge as the use of FileMaker Pro 8 matures. Nonetheless, it is clear that one of the strongest values delivered by this new technology is that of versatile design. The way that solutions are designed in FileMaker Pro is forever changed, and commercial developers and organizations will both reap the rewards of the product as the technology and its use continue to evolve.

Development: It's All About Execution

Once the design strategy is in place, FileMaker Pro 8 continues to deliver value by providing a very flexible environment for development. Regardless of the rigor with which the requirements, specification and design phases are completed, unforeseen opportunities and constraints will emerge in the development process. The earlier these issues surface, the more efficiently and effectively they can be addressed. The collaborative, real-time development environment of FileMaker Pro 8 provides many opportunities for optimizing the development process.

Ultimately, complex solutions consist of components and sub-components that are integrated to work together as a seamless whole. In prior versions of FileMaker Pro it was often necessary to build complex collections of sub-components all at once such that it was almost impossible to determine where one component ended and the next began. FileMaker Pro 8 provides for greater modularity of design, which allows discrete (or semi-discrete) components to be constructed and tested for later integration with other components. A time clock module used to capture the time spent on a project, for instance, can be built and tested by both developers and users before combining it with a time sheets module being simultaneously constructed. Integrating the two modules can happen through a number of possible design scenarios, which may involve merging functionality in the same file, combining functionality between two files, or importing elements from one file into another. In any case, the new features of FileMaker Pro 8 allow the integration of modules to occur in a seamless and straightforward manner.

The ability to build and test small sub-components and modules represents a significant improvement in the way FileMaker Pro solutions are developed. It allows for unforeseen issues to emerge through the development process in a more contained and controlled manner. With adequate testing, the functionality, usability, and appropriateness of each module can be well vetted before the module becomes immersed in the overall solution. Catching the problems early on will help accelerate their resolution and keep the problems in one module from detrimentally affecting other modules.

FileMaker Pro has been a real-time development environment for years. The increased ability for multiple developers or development teams to access the full functionality of a solution without interfering with other developers, users, and testers places FileMaker Pro 8 in a new league. Certainly, more complex and mission critical development activities might best be orchestrated using separate sets of files, but many development projects can now peacefully co-exist with other development projects, with testing, and even with live use of the solution. In cases when a second set of development files is warranted, the integration of new functionality into the live file set is more easily handled due to advances in script importing, data importing, and copying layouts between files.

While implicit in the above illustrations, the value and leverage of collaborative development should not be



underestimated. Increasingly, there is a strong sentiment in programming communities that team oriented development produces significant advantages over the single developer approach. Combining developers with other developers or developers with SME's in the code writing process can enhance the quality of the development in many ways. Two participants focused on the same problem and bringing different perspectives are more likely to uncover issues early on and to see a wider variety of possibilities than a single developer. If one partner in a pair is also intimately familiar with the needs of the organization, the demands of the workflow, and the expectations of the users, so much the better.

Finally, the speed of development in FileMaker Pro 8 promises to be outstanding in comparison both to its predecessors and to other development environments. Many operations, which used to take several files, fields, relationships, and scripts, can now be managed with a small handful of these elements. And there are more robust tools available to simplify much of the repetitive and laborious programming with respect to navigation, interface control, and data management. Once the understanding of the new feature set is more mature, developing solutions in FileMaker Pro 8 could take less time than in previous versions of the product.

For the most part, developing software solutions is a process of discovery and adaptation. It is not until we put the pieces of our plans and strategy together to make them function as a harmonious and well-orchestrated whole that we understand how the design of a solution will come together. Integral to this process is feedback from SME's, users, business leaders, developers and the solution itself as each function is created, tested, revealed, and integrated into the larger context of other systems and the organizational environment—the longer the feedback cycle, the greater the possibility for wasted effort and for laying ill-fated foundations. Much of the new functionality available in FileMaker Pro 8 provides for increased collaboration involving the SME's and other stakeholders during the development process. Such involvement tends to shortcut feedback cycles, allowing the development team more insight into the opportunities and constraints that emerge from the solution. Ultimately, this saves valuable resources and produces a more effective end result.

Testing: The Often Ignored Ounce of Prevention

Let us face it; no one wants to test software. Inventing the testing procedures alone is challenging enough, and there are rarely adequate resources to complete the solution much less to test all of the variables. Furthermore, those who do take the time are often quickly disheartened by unexpected discrepancies in data and in user interactions, both of which invariably compromise otherwise well-conceived and well-written solutions. Testing commercial applications is even more demanding, as many customers in an inconceivable variety of environments will use them. Even large software companies use public beta testing cycles (at times unbeknownst to end users) to vet and refine their solutions.

Regardless of the resistance to and the lack of resources for its execution, there is little doubt that software testing pays off, and it is not enough to have the development team do the testing. Assuring the stability of the software requires testing at all stages of the process and by a wide variety of users. In cases where the time, resource and expertise is not available for a formal testing process, the FileMaker Pro real-time environment



can provide tremendous advantages by reducing feedback cycles and involving end users and subject matter experts throughout the testing process. Furthermore, as even extensive testing is vulnerable to unforeseen consequences, after initial beta testing is complete, issues that arise during early implementation can be easily addressed without interfering with the ongoing use of the solution.

One of the challenges in testing cycles is that feedback from tester to developer takes too long, and planned testing periods can be interrupted by show-stopping problems. In formal testing environments, professional testers are employed to run through a series of defined sequences in order to test each feature and function, reporting any issues back to the developer. With FileMaker Pro, developers working directly with users in early testing cycles can vet initial issues in real-time, identifying and repairing many on the spot in order to keep the testing process moving. In cases where more research is required to resolve a show-stopping issue, the user can move on to other aspects of testing or return to other work while the developer fixes the problem. The fact that testing and repairs can be done in the deployment environment eliminates the need for compiling cycles as well as issues that sometime arise from the compiling process.

Because a real-time development environment can dramatically shorten testing cycles, end users and subject matter experts who may not have time for long testing periods can be more involved. Thirty-minutes of testing in real-time with user and developer can accomplish quite a bit with respect to evaluating the usability and efficacy of implemented workflow. Furthermore, a user may identify issues that a professional tester is unable to see due to the user's familiarity with both workflow processes and expected data. In some situations, testing can be incorporated into a user's regular job function, especially after a solution is implemented, when refinements are addressed or new features are added.

For many developers and organizations, testing is considered a necessary evil and can be ignored all together much to the detriment of the implementation. The real-time nature of FileMaker Pro offers the ability to reframe the testing project, offering new opportunities to make the process more efficient and effective. FileMaker Pro 8 extends this capability by allowing access to design functions while others are accessing database functionality. This gives the developer more options and flexibility in the testing process as well as allowing users and SME's to be involved in a manner that further leverages precious resources.

Implementation: The Art of the Roll Out

Rolling out new software solutions in an organization is tricky business. Even when solutions are well designed and tested, it takes time to integrate changes into business processes and systems. In addition to the technical challenges, user buy-in and adoption can present significant obstacles. The more processes, systems and human beings affected by the software, the more challenging the implementation.

In order to minimize the negative impact and resistance to adoption, many organizations phase in new functionality incrementally and over time. Whenever possible, even modest software implementations are subdivided into discrete implementation units. Once a unit is in place and working, the next unit can be rolled out. This modular approach allows for multiple stable points throughout the implementation process that deliver the full value of the software in a timely manner.



FileMaker Pro 8 is an exceptional tool for an incremental roll out strategy. With a modular approach to design (discussed previously, and in more detail elsewhere), solutions can naturally be introduced module by module. Additionally, the ability of FileMaker Pro 8 to interact and exchange data with other software applications and solutions (via XML, ODBC, JDBC, etc.) allows for the development of less complex solutions by interacting with and leveraging pre-existing systems. Though this might at first appear to be more work, such a methodology provides an organization with the opportunity to focus at each stage on the functionality that is most immediately necessary and that will produce the greatest value.

Another advantage to an incremental approach (especially with a tool as new and powerful as FileMaker Pro 8) is that it offers a learn-as-you-grow environment. All aspects of the process, from requirements gathering to implementation, are bound to mature as each subsequent phase is successfully realized. The needs of the environment will also be transformed to some extent as each aspect of the software is introduced. Knowledge garnered from earlier phases in the process will provide better intelligence for use in later phases, ultimately producing superior results. Creating processes that allow new technical and design intelligence to 'fold' back into prior phases will assure that early work maintains its relevance and viability within the greater solution.

Certainly the timing of the roll out is a critical part of the equation. It is not uncommon for an organization to recognize the need for a solution late in a business cycle or to wait until the last minute to move forward on a critical project. Once the decision has been made, it is compelling to think that the solution is required as soon as possible. The specifics of the timing, however, do not fundamentally affect the fact that organizational processes are best rolled out incrementally and over time. The greater the complexity and number of variables, including technologies involved, related systems, and personnel, the greater the need for an incremental strategy. Seeking out multiple stable points along the implementation path provides the best chance for success when rolling out even modest software solutions.

Maintenance & Evolution: Leveraging the Investment

It has been said that software refinement is never really finished, but simply abandoned. While it is true that a solution can be engineered far beyond a point of viable returns, there is another side to this story. Solutions are created to remove constraints. As each constraint is removed, new constraints are revealed. When new constraints are significant enough to matter, new solutions are required. This cycle of constraint and solution is further influenced by the dynamic nature of both the marketplace and the organizational environments served by software applications. With few exceptions, software solutions that do not adapt to new constraints and changes in environment do not continue to produce value over time.

With the time and resources required to produce effective software solutions, it makes sense to use a design methodology and development environment that support the ongoing maintenance and evolution of the software system. Again, the FileMaker Pro flexible, real-time environment facilitates the introduction of new features and functionality as well as subtle changes and the maintenance of existing code. It is easy to underestimate the productivity enhancing power of adding or moving a data field on a layout or changing the field tab order. With FileMaker Pro 8, refinements like these are just a fraction of what is possible when it



comes to enhancing existing functionality and exploring new possibilities. Because refinements and alterations can be made very quickly, a wide variety of ideas can be easily tested for value and viability.

The new features in FileMaker Pro 8 allow the developer to provide access to some or all of the design features so that super-users and subject matter experts can experiment with new ideas in workflow, functionality and reporting. Because this access is very granular, end users can be given the ability to test their ideas themselves without compromising the balance of the solution or the data behind it. In some cases, users and subject matter experts will create new features and functions on their own, in other cases their efforts will produce prototypes for the developer to refine. Regardless, keeping the users involved in the exploration and design process can yield tremendous value over the lifecycle of a solution.

Solutions that do not adapt to the changing needs of their environment quickly lose their value. Those that can adapt quickly and effectively to the needs of an organization and its market will continue to produce value over time. While guidelines for adaptation and evolution must be followed in order to maintain the integrity of the solution, providing super-users and subject matter experts with the tools they need to experiment and invent can dramatically increase the value of a solution. FileMaker Pro 8 provides a flexible, real-time environment that allows developers to empower the user while protecting the core architecture and data.

Conclusion: Clarifying the Proposition

Due to the transparency of much of the new power in FileMaker Pro 8, many users may continue to design solutions as they have in prior versions. Others will go directly into the development of large-scale, enterprise solutions. In either case, and regardless of the outcome, the real value proposition of FileMaker Pro 8 may be missed.

It's true: FileMaker Pro 8 marks a new age for the tool and the solutions it inspires. The power and flexibility of this new environment, however, comes with a learning curve, increased levels of creativity and insight, and the requirement of greater diligence and rigor throughout the process. Indeed, learning how to develop elegant and effective solutions in FileMaker Pro 8 will happen incrementally and over time regardless of the way in which it is initially approached. Rather than looking at this process as a hurdle or barrier that must be crossed in order to get to greener pastures, it can be seen as a journey through which both understanding and value are received. Focusing only on the end results and ignoring or avoiding the steps along the way will diminish both the experience and the deeper value available.

FileMaker Pro 8 provides an outstanding environment for engaging the process while simultaneously creating value for businesses and organizations of all types and sizes. As the new features and functions are embraced for their technical value, it is important to embrace the entire value proposition. This includes setting and managing appropriate expectations and exercising proper restraint and consideration at each stage of the solution creation process.

Finally, what the FileMaker 8 family of products represents is a significant leap forward in the maturity and possibility of the software solutions it supports. Nowhere else do we find a tool so capable of seamlessly integrating the needs of the single-user desktop database and the enterprise-level solution. Being effective in



this new arena will require the development of a commensurate level of maturity in the way database solutions and services are delivered. Clearly this is a journey worthy of attention.

About the author

Michael Thompson is the director of Software Sales and Implementation at New Millennium Communications, Inc., a FileMaker Solutions Alliance Partner. With more than a decade's experience using FileMaker Pro, he has helped NMCI and its clients build solutions ranging from single file databases to full-scale CRM/MRP systems.

(Footnote)

¹ The relationships graph is neither a flow-charting nor ERD generation tool, and should not be substituted for one. There are limitations to using the relationships graph for this purpose, which are inherent to its primary function. See the document on the FileMaker Pro 8 Relational Model for a more detailed understanding of the relationships graph.



The FileMaker Pro 8 Relational Model

Summary

This document introduces the relational model of FileMaker® Pro 8 and the rationale behind it.

- **Pre-FileMaker Pro 3 -- Flat Files**
 - repeating fields – “sub-records” for simple one-to-many relations
 - lookup fields allow copying of data based on matching key values
 - no ability to display data directly from other files
- **FileMaker Pro 3-6 -- Relational**
 - repeating fields and lookups still supported
 - can define relationships from one file to another
 - can display and reference related records from a directly-related file
 - must create intermediate calculations to propagate data from more than one relation away
- **FileMaker Pro 8 - Relationships Graph**
 - repeating fields and lookups still supported
 - can define relations between all tables in a database in a relationships graph
 - can display and reference related records from any table in the relationships graph
 - relationships graph can span multiple files; each file has separate relationships graph
 - relational operators can be relative (“theta” joins); multiple predicates supported

Pre-FileMaker Pro 3 -- Flat File Model

Before FileMaker Pro 3, FileMaker was designed to be used as a flat file database with only very limited relational capabilities. A flat file is simply a file or table consisting of a set of records (or rows) and a set of fields (or columns). Each record can contain a value for any number of the fields defined on the table. For example, one might define an Invoices table where the fields are things like Customer Name, Customer Address, Invoice Date, Invoice Total, etc. One record in the Invoices table exists for each invoice.

These versions of FileMaker Pro had limited relational capabilities or relational substitutes: repeating fields and lookups. Repeating fields allow FileMaker Pro to store more than one value for each field and thus create sub-records on each record. For example, to create an Invoices table, it is necessary for each invoice record to hold a set of items ordered on that invoice. To accomplish this, one could create fields Quantity, Item Description, and Price and then define these fields to have twenty or so repetitions each. When these fields are laid on a form side-by-side with their repetitions running vertically, it would then be possible for users to enter up to twenty items on the invoice and a calculation could be defined to total up the items for the invoice.

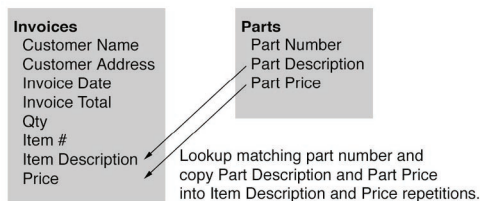
Qty	Item Description	Price



Lookups allow designers to make FileMaker Pro look up a value for a field from the same table or another table based on matching values. In this Invoices example, this would make it possible for the designer to have a separate Parts table and have FileMaker Pro lookup the price and description of the item and copy it over to the appropriate fields in the Invoices table when the user enters a part number in the invoice.

Qty	Item #	Item Description	Price

These fields are looked up from Parts table and copied into Invoices table based on Item#.



While repeating fields and lookups have proven invaluable in creating interesting database solutions in an otherwise flat file model, a relational model can provide a great deal more functionality.

FileMaker Pro 3-6 -- Relational Model

One of the more serious limitations with the flat file model and lookups is that it is difficult to share common data. As a result, there is a great deal of data duplication. The same data is needlessly stored in many different records. This makes the database larger and slower than necessary and this duplicated data is at risk to becoming out-of-date and inconsistent.

In the Invoices example above, the customer information (name and address) had to be entered for each invoice even if the same customer information was already entered for a previous invoice. In addition, the part description and price had to be looked up and copied over to the Invoices table for each item ordered. Certainly there are times when this might be advantageous -- for example, it is probably best that the part prices are copied when the invoice is created so that future price changes don't affect old invoices or that the customer name and address are looked up so that the invoice is a reliable historic record of the transaction. However, this is not always the desired behavior. What if you always want the most current data? To produce a customer statement (a list of unpaid invoices) to mail to a customer, it is important to use the customer's current name and address. It is possible to relookup the data periodically but this is time-consuming, does not guarantee that the data is always current, and requires that the records be modified.



FileMaker Pro 3-6 addresses these issues by introducing the ability to define relationships between files (tables). A relationship describes how records in one table are related to records in another table. In FileMaker, these relationships are based on simple single field matches -- the value of one field in a given row is matched against the values of another field across all rows of a different table (or the same table in the case of a self-relationship). In other words, for each “parent” record, FileMaker finds the set of matching “child” records where a given field has the same value as the parent record. The set of matching records could be empty, could be a single record, or could be a small or large set of records.

Invoice Record

Invoice #	Customer ID	Name	
		Address	

Qty	Item #	Item Description	Price

Invoice Total

portal to related Line Item records

In the Invoices example, it would be useful to take advantage of a relational model to create a separate table for all the customer data. In the Customers table, each record would correspond to a different customer and be identified with a unique Customer ID. In each record in the Invoices table, the customer’s current name and address can be displayed by relationship. This Customer ID can be used to define a relationship between the two tables so that we can display current data from the Customers table without having to copy it over to the Invoices table.

Customer Statement

Customer ID	Name	
	Address	

Invoice #	Date	Description	Amount

Balance Due

from Customers

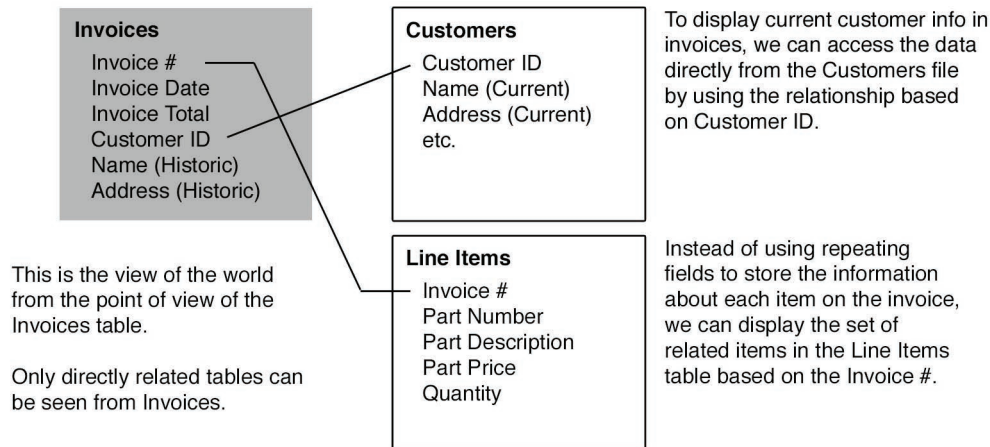
header or leading summary part

body (in list view)

footer or trailing summary part

Further, we can do away with the use of repeating fields to hold the individual invoice items on each invoice. We can create a Line Items table to contain a record for each item by defining a relationship based on the invoice number between the Invoices table and the Line Items table. When we display a record in the Invoices table, we can use the FileMaker Pro relational capability to display the set of Line Items records which have the same invoice number as this invoice. By doing this, we eliminate the limitation of having a fixed number of repetitions for the item data and we can generate more complex reports and queries involving the set of all line items for all invoices.

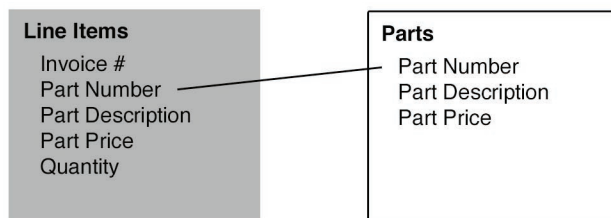




The FileMaker Pro 3-6's relational model is more flexible and more powerful than its original flat file model. However, it still has limitations. It is not easy to get a big picture view of the database and all of the relations between the tables since the designer must define every table's relations to each of its related tables separately from all of the other tables. This model makes it difficult to do queries involving other tables in the database. It requires extra work for database designers who want to view, work with, and perform queries across all of the tables in the database rather than only those directly accessible from each table.

For example, it may be of interest to generate a list of all customers who have purchased a particular product. To achieve this, it is necessary to do some fairly complex scripting using multiple Go to Related Record steps in a series of subscripts, building multi-keys in global fields with utility relationships at each intermediate table.

It is also a limitation of this model that you cannot directly access fields in related tables that are more than one table away. It is often necessary to create calculated fields or looked-up fields in order to display data. Intermediate fields are necessary for every field that the designer wishes to access from more than one table away.



This is the view of the world from the point of view of the Line Items table.

Only those relationships defined in Line Items are visible.

Each table has its own partial view of the database defined separately from the other tables.

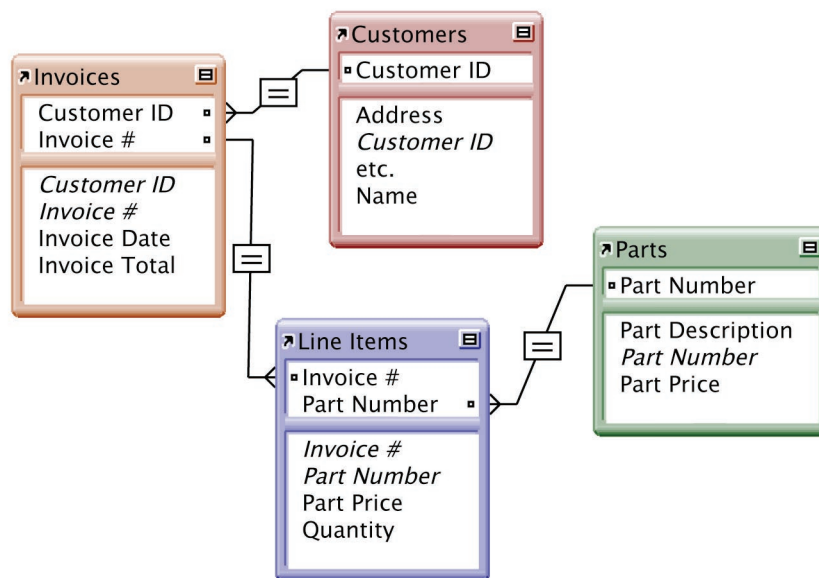


The view of the database is different for each table in the database. Each table is only aware of those tables that it is directly related to and each table is NOT aware of tables that are directly related to it. For example, the Line Items table is not aware of the relation from Invoices to Line Items. The relations in this model are one-way.

FileMaker Pro 8 -- Relationships Graph Relational Model

The FileMaker 8 product family introduces a more complete relational model to FileMaker Pro. This model is based on the interface of a “relationships graph” where the tables and relations between the tables form a visual graph of nodes and connections. This graph allows the designer to see all the tables and all of the relations at once if desired. For complex databases, the designer may choose to break up the graph into more manageable pieces.

Representations of tables in the database are shown in the graph and relations are defined between the tables and shown as connections between the tables. The connections are drawn between the match fields that define the relation. By requiring that every table occurrence in the graph have a unique name, it becomes possible to describe very complex relationships involving many intermediate tables with very simple qualified names. As a simple example, to refer to and display the related part description for a particular line item in the invoice, it would only be necessary to reference “Parts::Part Description”. If we’re asking for the data from the Invoices table, the database engine is able to determine the set of relations leading from Invoices through Line Items to Parts, perform the required join and return the value of the appropriate row. By simply referencing Parts::Part Description, we’ll get the value of the field from the first matching related record. Using portals we can access the entire set of related records. This relationships graph model and the field name qualification syntax (“table name” :: “field name”) is comparable to how many other database systems and languages (including SQL) work with tables and fields in a database. By adopting this model and syntax, FileMaker has become more accessible to developers accustomed to other database systems and traditional relational databases.



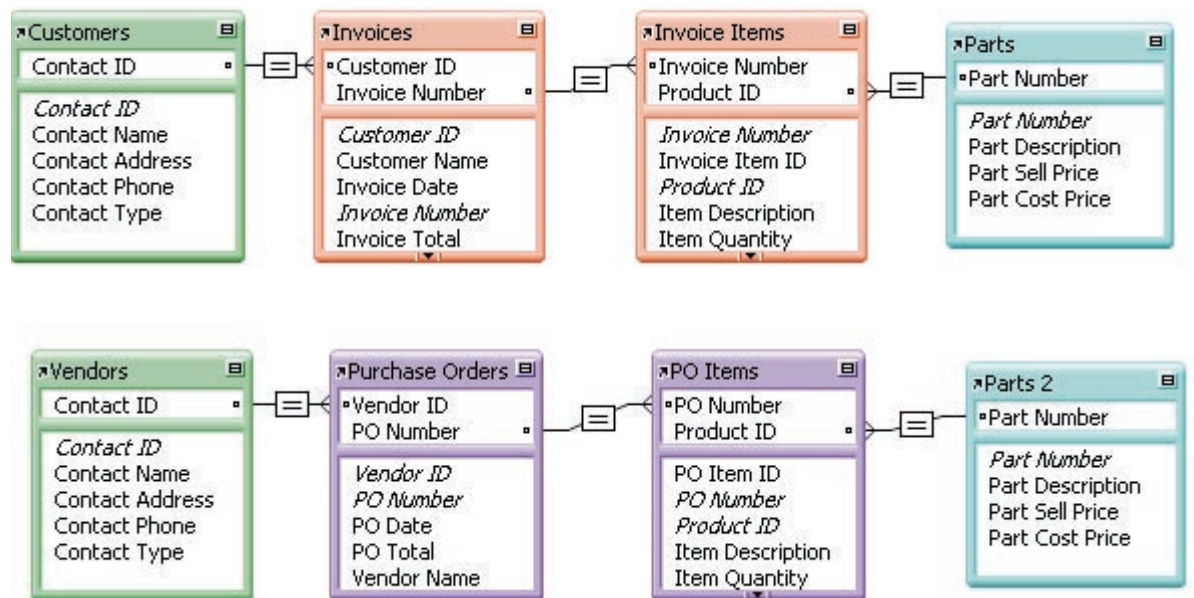
Note that this model does not have the limitation of the previous model where it was not possible to directly access fields in tables that are more than one table away. Therefore, it is no longer necessary to create intermediate calculated or lookup fields to copy the data from one table to the next. Each table now only needs fields defined for the data that belongs to that table. Furthermore, this model allows much more complex queries to be performed from any point in the graph involving any number of tables in the graph no matter how far apart as long as they are somehow related.

Tables on the Graph

One constraint of the relationships graph is that it must not include any cycles. Another way to say this is that there must only be one path between any two tables. If more than one path were available from one table to another, then it would not be possible to determine the desired path by simply referencing the desired table.

If an additional relationship is desired from one table to another, it is necessary to add an additional table occurrence to the graph, based on the same table.

Each occurrence of the table in the graph must have a unique name. Names can be generated by appending sequential numbers (i.e., Invoices 2, Invoices 3, etc.) or the developer can provide unique names. Note in this example that there are two occurrences of the Contacts table, named “Customers” and “Vendors”, and also two occurrences of the Parts table, named “Parts” and “Parts 2”.



Note that there is not any requirement that the tables in the graph have to be connected to each other. Tables can be present in the graph that is not connected to any other table in the graph. There can also be sets of tables in the graph that are connected to each other but not to other tables in the graph.

“Tables” and “Table Occurrences”

We refer to tables in the graph as “table occurrences” and to the underlying tables as “base tables”. Nevertheless, in common usage, when referring to table occurrences on the graph we often refer to them simply as “tables”. At first, this can be confusing, because the concepts of the table as well as the table occurrence are new. It is similar, however, to the common usage of the word “field” to refer to a field on a layout. It isn’t generally necessary to refer to a “field occurrence”, since it is implicit based on usage when one is referring to the field itself versus referring to the particular occurrence of the field on a layout.

Relationships Are Not Directed

The relationships in the relationships graph are not directed. A relationship defined between any two tables works in both directions (not in just one direction as in FileMaker Pro 3.0-6.0). For example, using the same graph, one can show the related customer data on an Invoices layout and show the related invoices data on a Customers layout.

However, many options defined on the relation are directed. When you define or edit a relationship between two tables, you have several additional criteria that can be specified including an optional sort order, whether to allow creation of related records, and whether to delete the related records in one table if a record is deleted in the other table. These options (and others) must be specified for one direction or the other (or both). For example, you may wish to specify that if a record is deleted in Customers, that all of the related records in Invoices are also deleted. However, you probably do not want to delete the related record in Customers when a record in Invoices is deleted.

External Tables

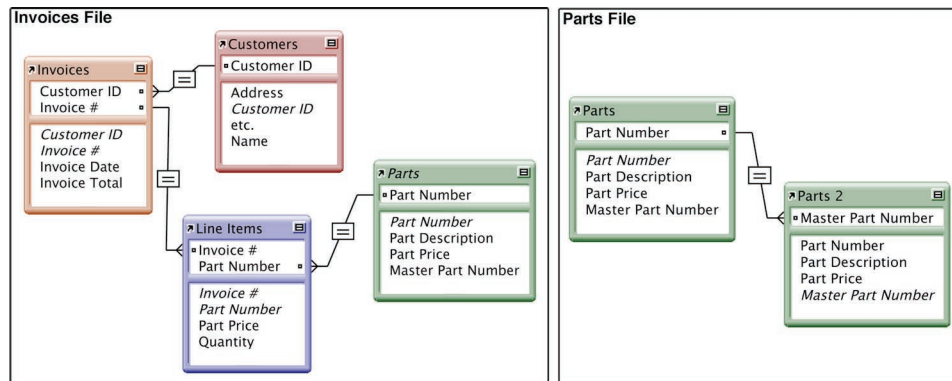
An important feature of this data model is that the tables in the relationships graph do not need to exist in the same FileMaker Pro 8 file as the graph. A table that appears in the graph may actually exist in a different FileMaker file. This means that layouts and scripts can be created in the same file as the graph to work with the tables in the graph while the data is pulled from an entirely different FileMaker Pro 8 file.

Multiple Tables, Multiple Files, and Multiple Graphs

Before FileMaker Pro 8, each file contained a single table. Therefore a database of multiple tables was necessarily made up of multiple files. However, any particular file could be used as part of multiple databases. For example, a Zip Code (Postal Code) table might be shared and referenced by several different databases.



The data model provides for the definition of many tables in a single file. Therefore, an entire database of many tables can be wholly contained in a single file on disk or can span many files, some of which have multiple tables. FileMaker Pro 8 allows the user to create and use files with various numbers of tables. A file could have only a single table and appear much like files from previous versions of FileMaker Pro, or it could combine multiple tables into one file and still access tables in other files.



The important thing to note here is that, just like in previous versions of FileMaker Pro, each file has its own view of the relations of the database. This means that each file has its own relationships graph that may or may not correspond to the relationships graphs of other files in the database. For example, the relationships graph given earlier of Invoices, Line Items, Customers, and Parts being all related together might be the relationships graph of just the Invoices file. The Parts table might exist in a separate Parts file and have its own independent relationships graph involving a self relation to show parts made up of subparts. The design and organization of the database is up to the designer.

The first reason that each file has its own relationships graph is to provide backward compatibility with databases created in previous versions of FileMaker Pro. In the relational model of FileMaker Pro 3-6 each file contains the set of relations between its table and the tables in the other files. There was no requirement that a given table had to define the same set of relations to other files that were defined to it. This is also true with the relationships graph model. Not only does this ability make conversion of databases to FileMaker Pro 8 possible, it also is very reasonable. There is no reason for a file containing a Zip Code table (or Employee list, etc.) to be aware of and define reciprocal relations to every other table that has a relation to it.

The second reason that each file has its own relationships graph is that FileMaker Pro has always allowed each FileMaker Pro file to be opened and modified independently of all other FileMaker Pro files. FileMaker Pro 8 does not require that all files referenced or all files that reference a given file be present in order to use or modify that file (though it will look for referenced files which are missing, the developer can cancel and proceed). FileMaker Pro 8 also does not require that one have design access to all files involved in the database in order to make design changes to one of the files. This is really important when modifying a file that is accessed by many other databases. It would be unreasonable to require all the other files to be updated to accommodate schema changes. Therefore, each FileMaker Pro 8 file still has its own set of relations (in the form of the relationships graph). The relationships graph in each file can be modified independent of whether any



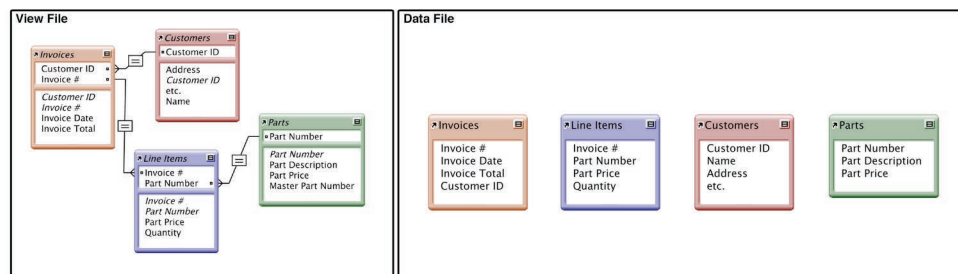
other files reference this file.

It is worth noting again that a relationships graph need not be fully connected. There can exist in the graph groups of tables that are connected to each other but not to other tables in the graph. Therefore, a single file can act as if it has multiple relationships graphs. Each disconnected sub-graph is, in a way, its own independent graph.

So, to summarize, a database can be built on one or more files. Each file can contain zero or more data tables. Each file has its own relationships graph that can contain disconnected graphs. Scripts and layouts defined in the same file can all take advantage of that file's relationships graph, as can calculations in tables that are in the same file as the graph.

Separate View and Data Files

A relationships graph can include external tables (tables in other files) and it's not necessary to have any tables defined in a file at all (you can delete the initial default table). This means that it is possible to create a FileMaker Pro 8 file that contains no data tables at all but has a relationships graph that relates a number of tables defined in external files. This file can still have a complete set of layouts and scripts that make use of the data tables in the graph. The result is that all the data tables could be in one file and all the layouts and scripts could be in another file.



This allows a developer to easily replace the “layouts and scripts file” without affecting the data files. However, there is a limitation here. The field definitions are part of the definitions of the data tables and are thus part of the data file. This means that changes to the data table fields must still be made in the data table files themselves.

Using the Relationships Graph

One implication of this relationships graph model is that it is always necessary to provide the context from which references in the relationships graph should be evaluated. For example, each layout in a database file must be defined to display records from a particular table or a particular occurrence of a table in the graph. Once this starting point has been established, it is then possible to reference any field in any table in the graph as long as it is related (even very indirectly) to the starting point.



This is true for all field references in the database regardless of whether that field is on a layout, in a script, via a calculation, part of an export order, etc. There must always be some way to establish the context (a particular table in the graph) from which all references to other fields in the graph are to be interpreted.

For layouts this requirement is straightforward in that part of the definition of the layout is the table that the layout is displaying data from. (This is necessary anyway to support multiple tables per file.) Afterwards, any reference on the layout to any field from any table in the graph is interpreted relative to that layout's defined table. This definition can be changed after the layout is created as well since FileMaker Pro 8 will still be able to interpret all the fields on the layout from the new table as well. If, at any time, a field is referenced which is not related to the layout's table, then an "<Table Missing>" message is displayed in place of the field's data, much like the messages in FileMaker Pro 3-6 to report that a field is missing, a file is missing, or the user does not have access.

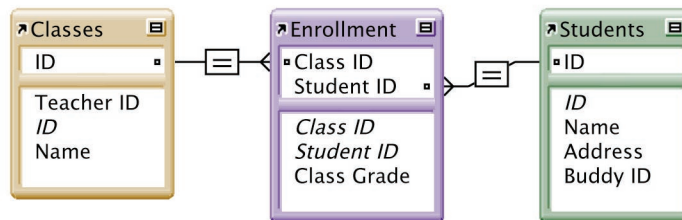
For calculations, it is necessary to specify where in the relationships graph the calculation's field references should be evaluated from. In the case of calculations defined as part of a field's definition, the base table is already known -- it is the table that the field is defined in. However, if the table appears in the graph more than once, it is still necessary to specify which occurrence of the table in the graph should be used to evaluate the calculation.



A Complex Example

The following is a more complex example that can be used to demonstrate some of the power and some of the intricacies of using a relationships graph. This database is intended to manage classes, students, teachers, assignments, and grades.

We can start with the basic tables: Classes and Students. Since each class will have many students enrolled in it and each student will be enrolled in multiple classes, we need an intermediate table, Enrollment, to provide the pairings in this many-to-many relationship.

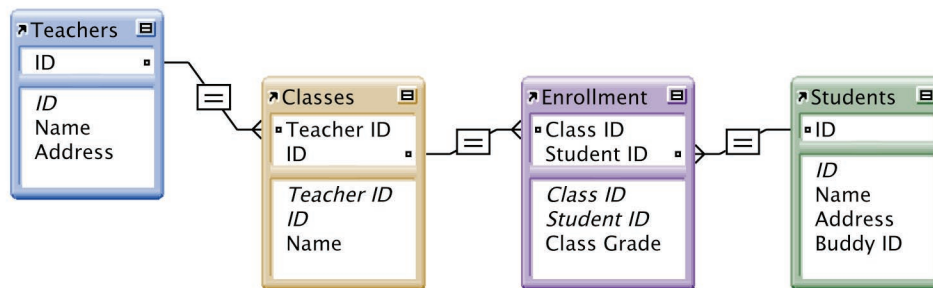


Now, it's worth noting that someone who does not have a lot of experience with relational database design could stop here with these tables and have a perfectly reasonable solution for tracking enrollment of students in classes. This person might also create additional tables for teachers and assignments and not even try to relate them all together in some meaningful fashion.



The graph we have so far allows us to create a layout based on the **Classes** table where we show the set of students enrolled for each class. This would be accomplished by specifying the **Classes** table when creating the layout. (The table assignment could be later modified in the Layout Setup dialog.) Then, a portal would be created on the layout and this portal would be defined to draw its records from the table **Students**. Note that it is never necessary to specify any of the intermediate tables. By specifying a table in the relationships graph, FileMaker Pro 8 can determine the set of relations between the starting table and the requested table to find the set of related records. In this case, the **Enrollment** table is the only intermediate table in the set of relations between **Classes** and **Students**. To complete the creation of this layout, a set of fields from **Classes** (like the class name) and a set of fields from **Students** (like the student name) would be placed on the layout as desired. In this same manner, we can create a layout based on the **Students** table that shows the set of classes each student is taking.

This example can be taken further by adding in a table for **Teachers** that also benefit from a one-to-many relationship. Each teacher can teach many classes but each class has only one teacher.



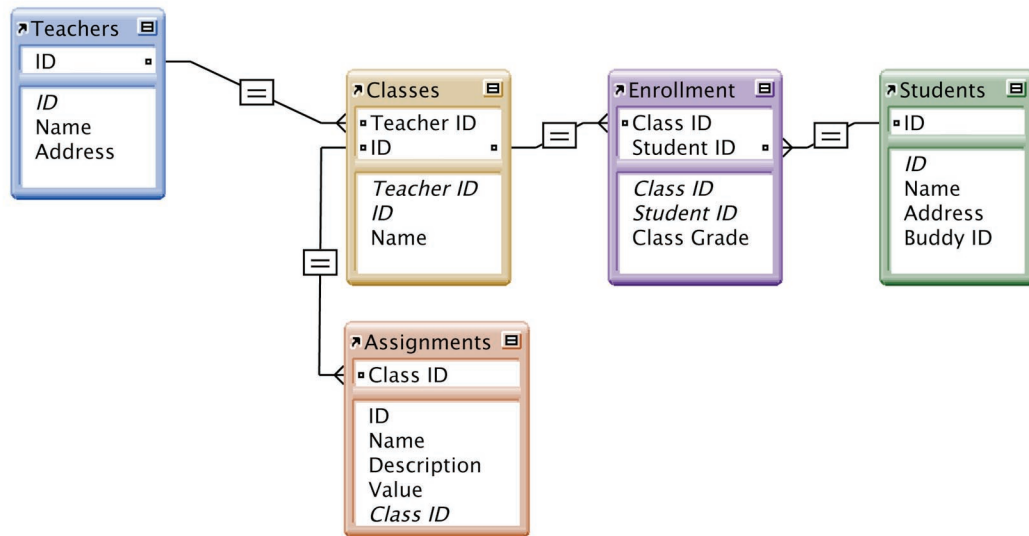
Now this allows us to modify our layout based on the **Classes** table to include the name of the teacher assigned to this class by merely placing the **Name** field from the **Teachers** table (**Teachers::Name**) on the layout. Since the relationships graph specifies the set of relations between **Classes** (which the layout is based on) and **Teachers**, FileMaker Pro 8 can perform the necessary operations to come up with the set of matching records. If **Teachers::Name** is placed directly on the layout (not in a portal), FileMaker Pro 8 will display the data from the first matching related record. The field **Teachers::Name** could also be added to the **Students** layout just as easily to display the teacher's name for each class that the student is taking.

Student Name Blake Wren		ID 1008
Class ID	Class Name	Teacher Name
BIO1251	Biology	Francis Crick
ENG1003	English	Edgar Allen Poe
HIS1001	History	Herodotus
MAT1003	Math	Archimedes
MAT1320	Differential Equations	Albert Einstein
SCI1001	Science	Albert Einstein
SST1002	Social Studies	David Riesman



Layout based on Students displaying portal to Classes, including field from Teachers

Now add a table of assignments and relate these to the Classes table with a one-to-many relationship.



Now we can create another layout based on the Students table (or modify the existing one) to show all the assignments for each student. Note the complexity of this relationship between Students and Assignments but that it is not necessary to deal with this explicitly once the relationships graph is provided. Just as in the previous examples, we can create a portal of assignments in the Students layout and add fields from Assignments directly. In the previous FileMaker Pro relational model, we would have to create relations in each table's file and create a lot of intermediate fields in each of the tables between Students and Assignments in order to propagate the data all the way over from the Assignments table. With a relationships graph, we can just reference the desired field directly.

Context for Fields Displayed in a Portal

However, displaying the Class Names for each of the Assignments in the same portal cannot be accomplished simply by placing the Name field from the Classes table in the portal. Field values from intermediate tables in the graph are evaluated based on the layout's table, not the portal's table. If the class name is placed in the portal, then every row will display the same value, that is, the value from the first related class, which is not what is intended.

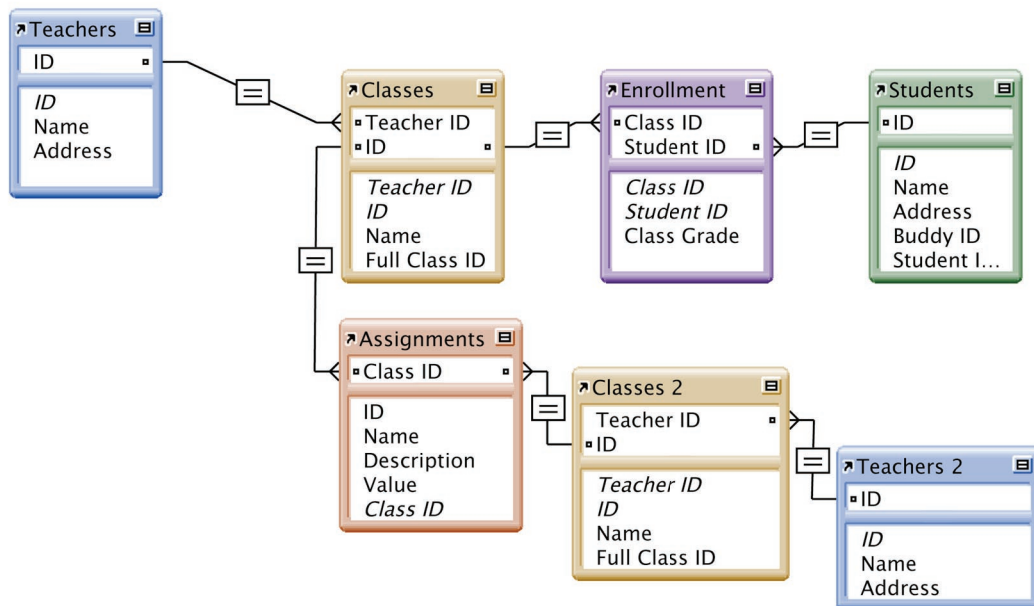


Student Name Blake Wren		ID 1008
Class Name (from Classes)	Teacher Name (from Teachers)	Assignment Name (from Assignments)
Science	Albert Einstein	Who was the raven, anyway?
Science	Albert Einstein	Expound on "Others Directed".
Science	Albert Einstein	Describe significance of DNA methylation.
Science	Albert Einstein	Explain the Lorenz equations.
Science	Albert Einstein	Explain significance of pyramidal stones.
Science	Albert Einstein	Demonstrate Archimedes' Principle.
Science	Albert Einstein	Build a cold fusion system.
Science	Albert Einstein	Devise a unified field theory.

Layout based on Students displaying portal to Assignments, including field from Teachers

Note the incorrect value in all rows in the Class Name and Teacher Name columns.

To display fields in a portal correctly, the field must be in the portal's table or in a table *beyond* the portal's table in the graph ("beyond" from the perspective of the layout's table). To display the class name that is related to the Assignment, it is necessary to create another table occurrence of the Classes table, and to relate that table occurrence to the Assignments table.



Student Name Blake Wren		ID 1008
Class Name (from Classes 2)	Teacher Name (from Teachers 2)	Assignment Name (from Assignments)
English	Edgar Allen Poe	Who was the raven, anyway?
Social Studies	David Riesman	Expound on "Others Directed".
Biology	Francis Crick	Describe significance of DNA methylation.
Differential Equations	Albert Einstein	Explain the Lorenz equations.
History	Herodotus	Explain significance of pyramidal stones.
Math	Archimedes	Demonstrate Archimedes' Principle.
Science	Albert Einstein	Build a cold fusion system.
Science	Albert Einstein	Devise a unified field theory.

Layout based on Students displaying portal to Assignments, including field from Teachers

The following rule is used to determine what record (and thus table) to use as the starting point for referencing fields in the context of a portal:

If the first relationship in the path used to get from the portal record's table to the field's table is the same as the one used to get to the layout's table, the window's current record (and thus the layout's table) will be used as the starting point for evaluating that field's references. Otherwise, the portal row's record (and thus the portal's table) will be used to evaluate that field's references.

Another way to say this is that the path to the displayed field must include the same full path as that to the portal's table, or else only the first related value will be displayed.

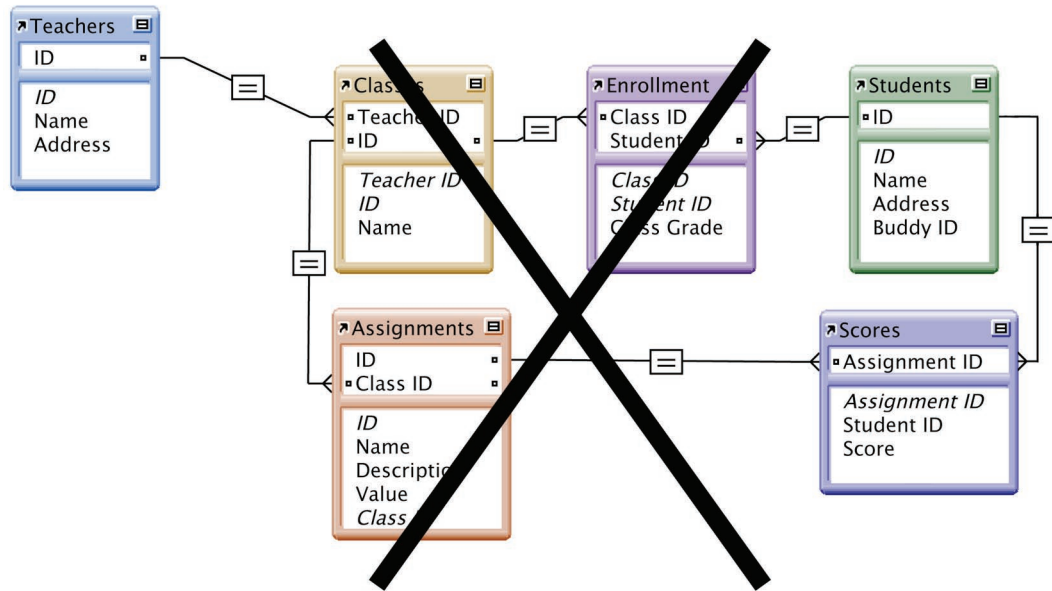
Note that this issue does not apply to other uses of the graph, such as displaying individual related fields (not in a portal), evaluating related fields in calculations, or using related fields in a find request, since in those instances there is only one starting point and one ending point (one full path) that is involved. It is an issue unique to portals because of the ability to display fields from a table occurrence other than the one that it represents.

Context in the Graph

Now we want to track each student's scores on each of the assignments. This is going to require another table (Scores) that is going to identify both the assignment and the student (by their respective IDs) for each score value. This is where it gets a bit more complex. If we connect the Scores table to the Assignments table based on the Assignment ID and we connect the Scores table to the Students table via the Student ID, we're going to create a cycle in the graph.

A cycle cannot be allowed in a relationships graph since it would mean that there is more than one path between any two tables in the graph.

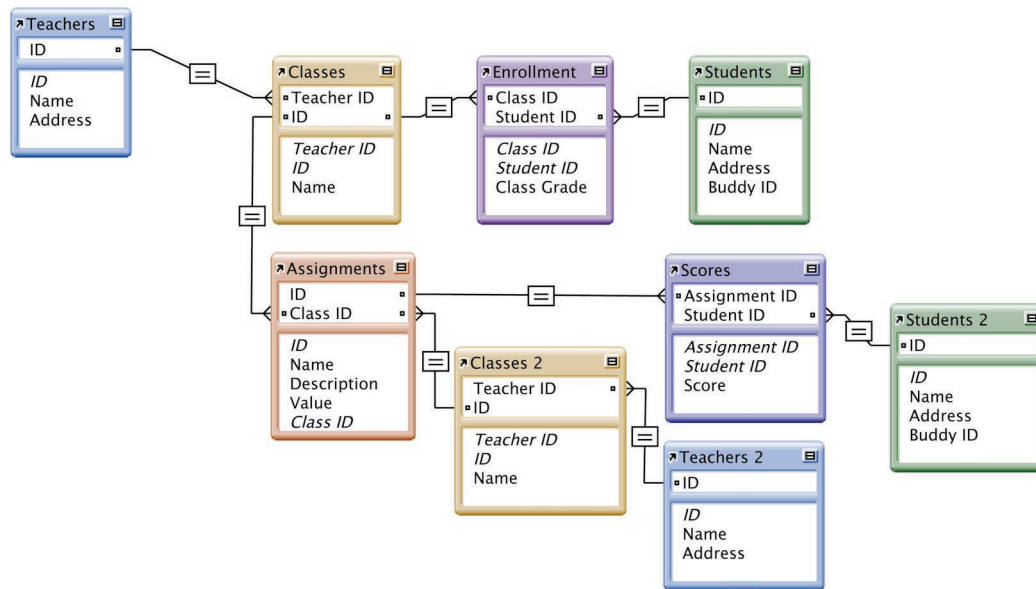




For example, if this graph were allowed it would no longer be possible to show the set of students enrolled in a class by merely defining a layout to be based on Classes (the starting point) and defining a portal to be based on Students (the end point). There would be more than one way to get from the starting point to the ending point and, depending on the data and the relations between the tables, each path may result in a different set of records.

Since there can be only one path between any two tables in the graph, the attempt to relate one table to another table more than once must result in the creation of another occurrence of the table in the graph.





In this picture of the graph, we have made the break in the cycle at the Students table so that there are now two occurrences of the Students table in the graph. There is still only one underlying Students table, but the Students table appears in the graph more than once and a layout or calculation can make use of either occurrence by referring to that occurrence by name.

There is no reason that the break in the cycle could not have been made elsewhere in the graph. It would be just as easy to have the Scores table appear more than once so that one version was related to the single Students table occurrence and the other was related to the Assignments table. Other organizations are possible as well. The designer is free to choose whichever makes the most sense for their solution or how the tables are accessed. In fact, if there is some reason to do so, the designer could break the cycle in several places or add additional occurrences of several tables. For example, both Students and Scores could be added to the graph a second time if desired so that Scores and Students 2 would appear in the graph as in the diagram above and Scores 2 could be added to the graph in a relation with Students. This would allow the designer to take advantage of the relation between Students and Scores 2 or that between Scores and Assignments as appropriate in different places in their solution.

With the Scores table integrated in the graph, it is possible to see once again how data from any table in the graph can be used anywhere else in the graph. For example, a layout based on the Students 2 table can show all scores that the student has received, and, in the same portal, display the assignment names, class names, and teacher names from their respective tables.



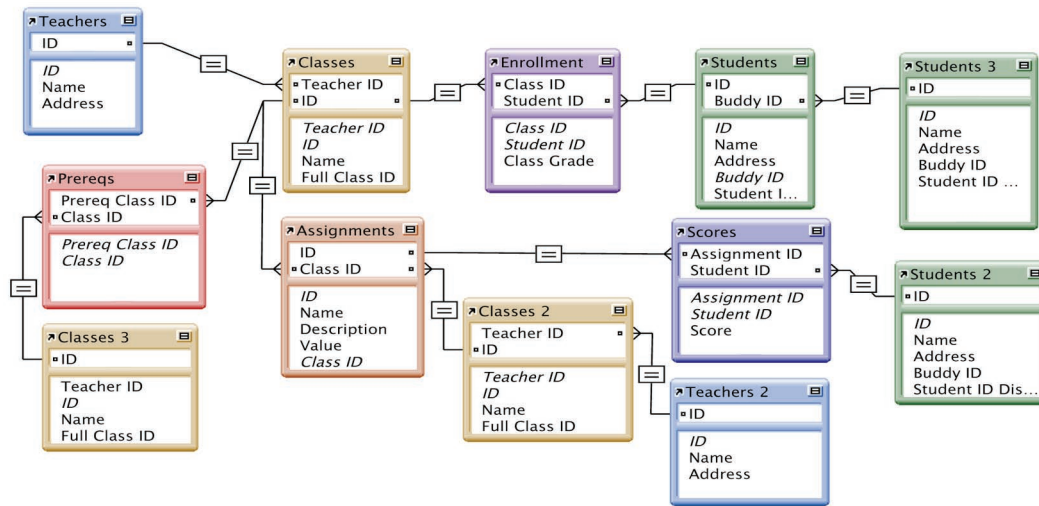
Student Name		ID			
Erika Fletcher		1248			
Class ID	Class Name	Teacher	Asmt ID	Assignment Name	Score
SCI1001	Science	Albert Einstein	5243	Build a cold fusion system.	95
MAT1003	Math	Archimedes	4953	Demonstrate Archimedes' Principle.	93
SCI1001	Science	Albert Einstein	5722	Devise a unified field theory.	82
ENG1003	English	Edgar Allen Poe	1176	Who was the raven, anyway?	88
SST1002	Social Studies	David Riesman	1763	Expound on "Others Directed".	84
BIO1251	Biology	Francis Crick	2621	Describe significance of DNA methylation.	92
MAT1320	Differential Equations	Albert Einstein	2726	Explain the Lorenz equations.	85
HIS1001	History	Herodotus	3773	Explain significance of pyramidal stones.	78

Layout based on Students 2 displaying portal to Scores.

Now that a table is present in the graph more than once, notice how the choice of the starting point matters. To display data for a student, we can choose to base the layout on either Students or Students 2. If you created a layout on Students and then created a portal showing records from Classes you would get the set of classes that the student is enrolled in since that is what the relationships between Students, Enrollment, and Classes defines. Now if you were to change the layout to be based on Students 2 and leave the portal displaying Classes you will get something different. You will get the set of classes for assignments for which this student has a score. (Students 2 --> Scores --> Assignments --> Classes) Since this student may not have completed all of the assignments for all of the classes or not all of the classes even have assignments, this set of classes may not be the entire set of classes that the student is enrolled in. (If you ask a different question, you may get a different answer.)

Now let us introduce a list of prerequisite classes for each class and a one-on-one buddy system for the students. For the prerequisite classes, we need yet another many-to-many relationship but this time between Classes and itself which introduces another occurrence of the Classes table. For the one-on-one buddy system, we need a one-to-one relationship between Students and itself -- a one-to-one relationship does not require an intermediate table.





The user interface of a portal makes it possible to display the power of the relational model very directly, but note that these same relationships can be leveraged in scripts and calculations without ever displaying the results.

Using Relationships For Queries

The relationships graph model lends itself well to queries. As described earlier, any number of tables in the relationships graph can be referenced directly. By simply referencing a field in a table on the relationships graph, FileMaker Pro 8 can evaluate the relationship between the tables to produce the appropriate query results as requested.

The context of a find operation is dependent on the current layout at the time the find is performed. This is true in a scripted Perform Find step as well as in a manual find. A “Go to Layout” step may be necessary, even if the layout is not displayed, to set the correct context for a scripted find.

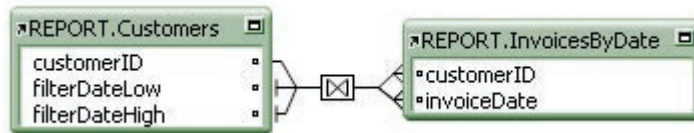
The relational model also supports the ability to define relationships that are based on matching rows on a relational operator other than equality (i.e., greater than, greater than or equal, less than, less than or equal, or not equal). FileMaker Pro 8 also supports relationships with multiple match fields. Relationships based on multiple fields, or multiple “predicates”, along with relative operators, add extraordinary power to the relationships graph.

For example, you can place a portal on a layout based on a table occurrence titled REPORT.Customers (which is based on the Contacts table), displaying all invoices to that customer for a certain date range. In FileMaker 3-6, this type of relationship would require complex calculated keys and/or large indexed multi-keys. However, in FileMaker Pro 8, it can be done with one relationship, two global “filter” fields in the Contacts table, and no additional indexed fields.



	REPORT.Customers		REPORT.InvoicesByDate
	customerID	=	customerID
AND	filterDateLow	=	invoiceDate
AND	filterDateHigh	=	invoiceDate

The relationship will appear in the graph like this:



Ad-Hoc Queries

The relationships graph described so far represents the set of relations that the designer of the database defines as part of the construction of the database. These relationships are invoked when references are made from one table in the graph to another. However, there can be times when it may be desirable to perform a query which defines an entirely different set of relationships -- a set of relationships not represented in the relationships graph. This is an ad-hoc query and the relationships graph model supports it. Such a query can be specified using a query language like SQL. The query includes the specification of the relations between various tables, as in this simple example:

```

SELECT Customers.Name, Invoices.InvoiceNum FROM Customers, Invoices
WHERE Customers.CustomerID = Invoices.CustomerID;
  
```

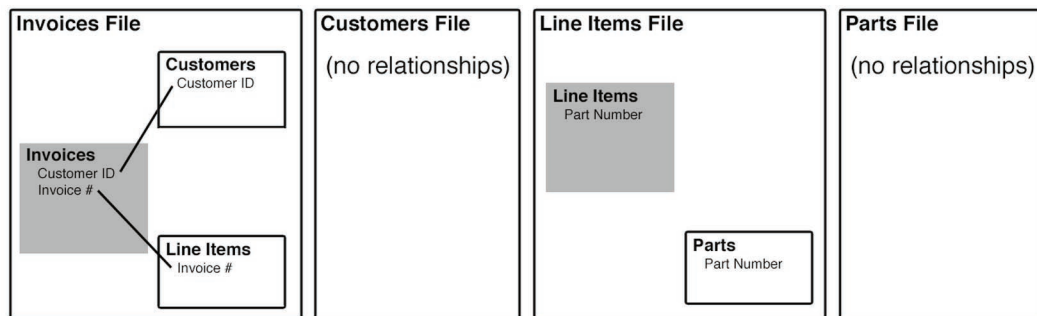
One method by which ad hoc queries can be posed to FileMaker Pro 8 is through the FileMaker Pro 8 ODBC interface.

It should be noted that the database's relationships graph can still play a part in ad hoc queries. Unlike most traditional databases, FileMaker Pro 8 supports the definition of calculated fields as part of the table. These calculated fields can reference fields in related tables as defined by the relationships graph. Therefore, an ad hoc query that defines its own relations between tables makes use of the database's relationships graph inadvertently if the query references a calculated field that references a related field. However, this dependency on the relationships graph is entirely transparent and is merely part of the definition of the calculated fields.

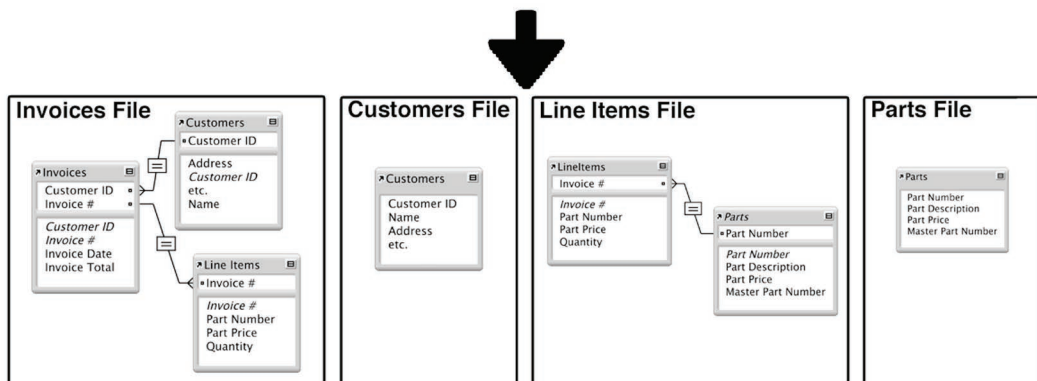


File Conversion

The automated conversion of FileMaker Pro 3-6 files that have relationships defined is possible since each relationship in the old file corresponds to a relation in the relationships graph of the new file. The names of the relationships in the old files simply become the names of the table occurrences in the relationships graph.



Prior to conversion, the four FileMaker Pro 3-6 files make up an Invoices database. The Invoices file is related to the Customers and Line Items files and the Line Items file is related to the Parts file.



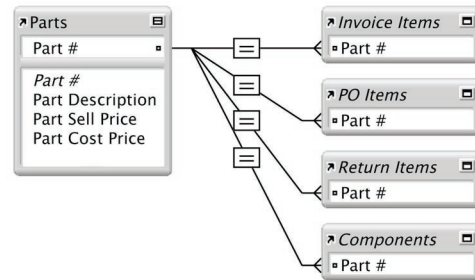
The result of automatic conversion of four files in a FileMaker Pro 3-6 database to the relationships graph: each file still has one table and each file still has a set of relationships to the tables in the other files, only now the relations appear in the relationships graph. In the files that did not have any relationships defined, only each file's own table appears in the graph.

In this example, each of the four FileMaker 3-6 files were automatically converted and a relationships graph was generated for each file that corresponds to the set of relationships in the original files. The conversion process does not try to eliminate all the intermediate calculations that the designer created in the previous relational model and it doesn't try to coalesce the relationships graphs of each of the files. The conversion process just brings the files forward and they continue to function as before. Each file still has only its view of its directly related tables. Those files that had no relationships to the other files in the database are left with graphs containing merely the one table in that file.



After conversion, the designer may decide to rework their database to take advantage of the new relational model. For example, they may wish to add the Parts table to the relationships graph of the Invoices file so that they can show part information on the invoice without the need for the intermediate calculations in the Line Items table. They may also choose to add a new interface file to better leverage the power of the new model, or to consolidate data files to eliminate other files and take advantage of the ability to have many tables in a single file.

Since the relational model in FileMaker Pro 3-6 only permits direct relationships to be defined, the relationships graph that results from the conversion process is always a very simple graph with the file's table appearing with a set of direct relationships to it (including self-relationships). Again, the names of the old relationships become the names of the table occurrences in the graph. After conversion, the designer can immediately take advantage of the relational model and create layouts based on the other tables to show data from different perspectives. For example, by creating a layout based on the Customers table, one can easily create a report showing all the invoices for a given customer or all the parts a given customer has ordered. This is made possible by the relationships graph because there is no fixed perspective. Layouts and calculations can be created from any point in the graph.



(Note that it may not be advisable to create layouts based on the other table occurrences in a converted file until one has a good understanding of how to manage context in scripts.)



File References in FileMaker Pro 8

Introduction

What are “File references”?

A file reference is the mechanism FileMaker® Pro 6 and FileMaker Pro 8 use to store the locations of external files, including both FileMaker Pro and in some cases other files.

When a solution refers to an external file, FileMaker Pro stores a “reference” to that file. The reference consists of an internal ID number and one or more places (paths) to use to look for the file when it is needed again. External files might be located on your local hard drive, on a shared network drive (although this is not recommended), hosted via FileMaker Pro peer to peer hosting, or hosted using FileMaker Server.

In FileMaker Pro 6 the developer never specifies the file reference directly. Instead the developer locates files using the OS level file browser or the FileMaker Pro “Hosts” network browser. FileMaker Pro then stores one or more paths to the file depending on the configuration of the client machine, the location of the files, and whether the “Save relative path only” box has been selected.

When selecting a file in one of these dialogs FileMaker Pro makes a determination as to whether this constitutes a new file reference or whether it can reuse an existing file reference. This determination is based on the file name and the path to the file.

In FileMaker Pro 6 there are four common path types stored within a file reference: Relative, Absolute, Windows Network, and FileMaker Network. A single file reference can store multiple different paths to a file, each storage location corresponding to the four different types of file references stored (although in practice there is likely a maximum of three stored).

What has changed?

One area of change involves how FileMaker Pro 8 identifies and locates files necessary in the operation of the database solution. FileMaker Pro 8 reveals the concept of a “file reference” to the developer and converts the previously hidden file references of .fp5 files into now-visible file paths for .fp7 files on conversion. A thorough understanding of the mechanisms involved in the conversion of these file references will aid the developer in ensuring their solutions find the correct files and are operating efficiently.

In FileMaker Pro 8 the four basic path types remain, however FileMaker Pro 8 introduces a new method by which to specify each path type. The limits in FileMaker Pro 6 are lifted and the developer is able to store multiple file paths within a single reference, including more than one path of the same type. Moreover, the developer is able to specify the search order within the File Reference for locating files. This enables the developer to ensure correct operation of the solution in single-user and multi-user settings as well as in development and deployment scenarios.



Testing conditions

Except when specified, the term “FileMaker Pro 6” will be used to refer to the behavior of versions 5.5 and 6 of the product. The extension “.fp5” will be used to refer to files created with FileMaker Pro 5.5 or 6.

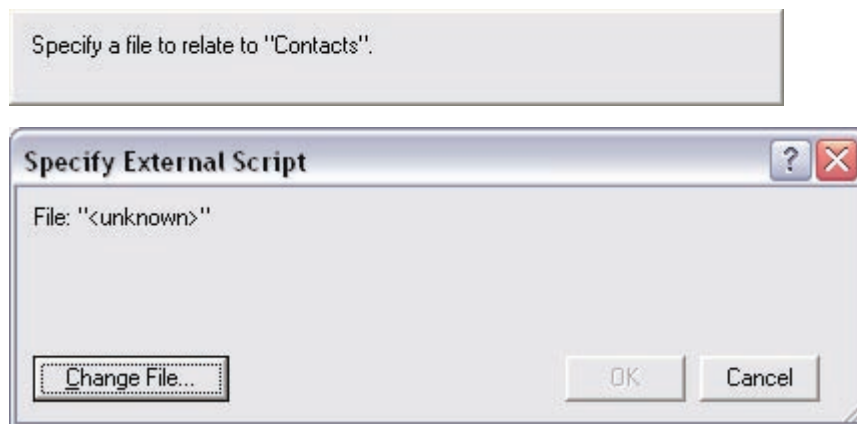
Topics not covered

There is another type of file reference that is not covered in this document. FileMaker Pro may also store references to images, OLE objects, and other files in records (e.g. an image in a container field) or on layouts (an Excel table on a report). These file references are stored in a different location and in a different manner from those used by the application for relationships and scripts and therefore are not covered by this document. A number of plug-in vendors have solutions for extracting the file paths stored in these references. Consult the FileMaker website, or <http://www.filemakerplugins.com>, for a list of plug-in vendors and solutions.

File References Overview

When are file references stored?

File References are stored for three primary uses. The most common times are when creating relationships to other files and when creating scripts that reference external files (e.g. Perform Script [External], Open File, Close File, Import Records). They are also used when defining value lists that are derived from another file. Here are the FileMaker Pro 6 dialogs that appear before a file reference is created when creating a relationship or an external script call on Windows.



It is important to note the distinction, however, between manual actions and scripted steps; the former does not use a file reference while the latter does (e.g. selecting “Import Records” from the File menu compared with the “Import Records” script step).

Each of these operations has the potential to store a new file reference or use an existing one depending on the situation in which they were created. For example, if a relationship and value list are created referencing the same file in all likelihood they will share a file reference. If, however, a relationship is defined to a second file,



the files are moved from a local development environment to a hosted deployment environment, and then a value list is created that references that second file then a new file reference may be created depending on the relative location of the files, the sharing status of the files, and whether the “Save relative path only” checkbox is checked or not.

What types of files do file references refer to?

File references are primarily used for FileMaker Pro files. A file must first be converted to FileMaker Pro format before you are able to create a relationship to it.

Similarly, a Value List cannot be defined to retrieve its values from a non-FileMaker Pro file. Likewise the Open[] and Close[] script steps only apply to FileMaker Pro files.

In some cases File References can also store information about other types of files that are available. Unlike the Open and Close script steps, the Import Records and Send Message script steps allow the specification of other files. These file references cannot be used by items requiring a reference to a FileMaker Pro file.

This is not foolproof however. Because of the extensive file location mechanism it is possible for FileMaker Pro to confuse files and file references. One of the mechanisms it uses when failing to locate a file is to search for a file of the same name but with no extension.

Why might I have multiple references to the same file?

When FileMaker Pro is asked to use an external file it tries to determine whether a suitable file reference exists already. Usually, if the files are in the same place as when a prior file reference was initially created, it is able to reuse the existing file reference.

However there are a number of cases where FileMaker Pro is not able to determine that it can use an existing file reference and instead creates a new one. Because this happens behind the scenes it generally does not cause problems. However it is not uncommon during development to move the solution around on the local machine or on the server. Although not recommended, sometimes during active development a developer might have multiple versions of a solution on a hard drive or a copy on the local machine and a copy on the server. With multiple copies of files located in various places it is easy for FileMaker Pro to become confused about which is the “correct” copy of the file. In a typical scenario files are first developed locally and then hosted with FileMaker Server with further development occurring remotely. References to the files were initially created on the local computer so when looking for a particular file it is possible that FileMaker Pro will find the “local” copy rather than the “hosted” copy. This generally manifests itself by the file being listed twice in the Window menu, file changes “disappearing”, or the perennial complaint from end users that the data they diligently entered has gone missing.

It is also possible, during development, to embed the DNS name or IP address of the development machine in a file reference. During development FileMaker Pro is able to locate the files. However, during deployment these files may no longer be available and performance may suffer while FileMaker Pro attempts to locate the file across the network. Worse yet is the situation where FileMaker Pro is actually able to find the file on the network rather than the locally hosted file. When “correcting” this behavior a new file reference may or may not be created storing the true location of the file.



File references in FileMaker Pro 5/5.5/6

Path Types

Before talking about how FileMaker Pro actually searches for files it is important to note the types of paths that might be stored in a file and how they differ from one another. The four common path types stored in .fp5 files are relative, absolute, Windows networking and FileMaker networking. On Mac OS, a file “alias record” was also stored in some cases.

Relative

Relative paths indicate where a file is located with respect to the current file. If two files STUDENTS.fp5 and TEACHERS.fp5 are located in the same folder then the path of the STUDENTS file “relative” to the TEACHERS file includes only the file’s name (e.g. “STUDENTS.fp5”). If the STUDENTS file was located in a subfolder of the folder containing the TEACHERS file then the relative path would include the subfolder name as well as the file name (e.g. “.:Subfolder:STUDENTS.fp5”, “Subfolder\STUDENTS.fp5”). A relationship from STUDENTS to TEACHERS under this structure would indicate a similar path format (e.g. “.:TEACHERS.fp5”, “..\TEACHERS.fp5”). Relative paths apply to hosted environments as well. If the files STUDENTS and TEACHERS are both hosted by the same FileMaker Server machine they will have a similar relative path as if they were both located in the same folder. This is independent of the actual folder structure on FileMaker Server itself.

Full (sometimes called absolute path)

The Full path to a file indicates where the file was located on a local workstation. Unlike the relative path, the full path stores the entire location of the file including the volume name or drive letter on which the file is located (e.g. “Macintosh HD:Users:Admin:Desktop:Keystone:STUDENTS.fp5”). A full path may be stored in addition to the relative path unless the ‘Store relative path only’ checkbox is checked, in which case it is not stored unless the target file is located on a different volume/drive than the current file. It may also be stored in conjunction with a network path.

If a full path is stored in a file reference it is platform specific. FileMaker Pro 6 creates a new file reference when targeting the same file on a different platform than the one on which the original file reference was created.

Windows Networking

The Windows path is slightly different from the Full path. The Windows path contains syntax that is specific to Windows operating systems (e.g. “C:\Documents and Settings\User\Keystone\STUDENTS.fp5”). As can be expected, Windows paths are not valid on Mac OS and file references that contain only a Windows Network path will not resolve on Mac OS. If the file cannot be found using another stored path, FileMaker Pro 6 will prompt the user to locate the file unless error messages are being suppressed.

FileMaker Networking

Network paths are stored in file references under two situations. The first occurs when the file being referenced is located on the network and is selected through the “Hosts” dialog. The second occurs when the referencing file has its sharing options set to multi-user and the “Save relative path only” checkbox is not checked.



There are two types of network file reference. The first type is the “local network” path and is indicated by an asterisk (*). The other is the full network address that is either the DNS name of the workstation or the TCP/IP address. A full network address is commonly visible in the “Perform Script: External” script step where the name of the file is followed by the network path.

Note: While FileMaker Server on Mac OS 9 supports AppleTalk networking and FileMaker Server on Windows NT supports IPX/SPX networks the scope of this conversion document refers only to TCP/IP networking since the others are no longer supported in FileMaker Pro 8.

The local network path indicates that the file was available in the “Hosts” dialog with the “Local Hosts” option chosen. The full network path indicates the file was located by choosing the “Specify Host...” option when locating the file or that the file had its sharing status set to multi-user when the file reference was stored. While the local network path is the more flexible of the two, a full network path leads to the best performance and is required in certain scenarios including multi-server partitioning of the database, or host and client in different subnets.

Alias

On Mac OS, an alias record was also stored under some conditions. The alias record could be used to have the operating system locate the file. In typical use, this would have similar behavior to using the relative and full paths, however Mac OS also has ways of resolving aliases when a file has been renamed but not moved, or moved but not renamed.

FileMaker Pro 5 “searching” behavior

FileMaker Pro 5 file references sometimes would locate files other than the ones the developer was expecting. One of the chief reasons for the “wrong file” being chosen is user error. It is possible that a developer, with multiple copies of the solution on their machine or the network, specified the wrong file when creating the file reference. In this scenario it is possible some operations will reference the correct files while others reference the incorrect file. The best way to avoid such a scenario is to always zip or stuff (or otherwise make inaccessible) duplicate copies of a solution. Files should only be available on the network from one host and hosted files should never be available through OS level file sharing or on a local machine.

Another reason for the “wrong file” behavior in FileMaker Pro 5 was that when locating a file, FileMaker Pro assumed the file reference it used was incorrect. That is, if a file being referenced was said to be located at “C:\My Documents\Keystone\STUDENTS.fp5” and it was not found at that location, FileMaker Pro 5 would check a number of other locations for the file along with adding and removing the extension from the specified file name. However failing to locate the target file in the location specified by the file reference FileMaker Pro 5 may still find files by looking in various other locations including the “default” folders and the local network (the “default” folders include the location in the “Open File” and “Save File” file pickers as well as the folder containing the current file and the Application folder). If unable to locate the file in the location specified by the file reference FileMaker Pro 5 would check the default folders for the file. This behavior is slightly different between the Mac OS and Windows. In a network situation FileMaker Pro 5 may find the target file on the local volume or drive or in one of the default folders if it is unable to locate the file on the same server as the current file or the server specified in the file reference.



On the Mac OS, it was possible for FileMaker Pro to find the file even after it had been moved to the Trash. This was due to the alias record for the file which enabled FileMaker Pro to locate files that had been moved from their original locations. FileMaker Pro 8 no longer stores an alias to a record when specifying the file on Mac OS X.

FileMaker Pro 5.5/6 Save Relative Path Only behavior

Beginning with FileMaker Pro 5.5 a new option was introduced called Save Relative Path Only (SRPO). This checkbox was added to the “Open File” dialogs to instruct FileMaker Pro 5.5, when creating and storing paths in file references, to only save the relative path.

This option is the default behavior for all new file references; in most cases to store the absolute and network paths one must manually un-check the SRPO checkbox when specifying the file. This option is generally useful because under most development and deployment situations all of the requisite files are located on the same host computer. By only storing the relative path it instructs FileMaker Pro 5.5 and FileMaker Pro 6 to only look for target files in locations relative to the current file. While the use of the relative path is applicable to FileMaker Pro 5 (assuming the file reference was created with FileMaker Pro 5.5 or FileMaker Pro 6 and the SRPO checkbox was not unchecked) the searching behavior does not behave as in FileMaker Pro 5.5 and FileMaker Pro 6 if the file is not located in its relative location.

The actual path stored when the SRPO checkbox is checked varies by platform. Under Mac OS X FileMaker Pro 6 was observed to only store the relative path if the SRPO checkbox was checked (as is the default). Under Windows 2000 FileMaker Pro 6 was observed to store both the Relative path and the Windows path when the SRPO box was checked. However, when searching for a missing file the Windows path was ignored and the SRPO checkbox (and modified search behaviors) were preserved. Note that when creating a file reference the SRPO checkbox is ignored when choosing a file that located on a host that is not on the local network. In this case only the Network Path will be saved in the File Reference. The SRPO checkbox is also ignored when choosing a file located on a different volume or drive than that of the current file. In this case the Full or Windows path will be saved in the file reference and, if the current file is set to multi-user, the network path to the host.

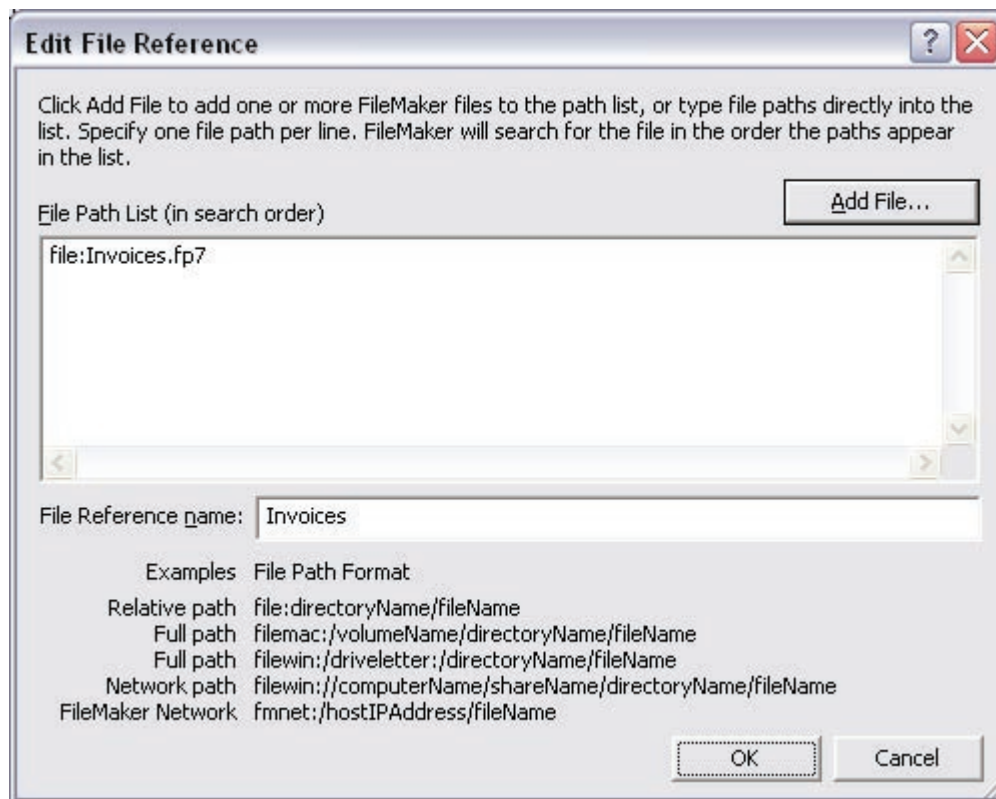
The SRPO checkbox not only affects what paths are stored but also how FileMaker Pro 5.5 and FileMaker Pro 6 will search for files when resolving file references. Unlike FileMaker Pro 5, when a file is not found in the location specified by the file reference and the SRPO flag is set, FileMaker Pro 5.5 and FileMaker Pro 6 will not look elsewhere for the file. In this case the “searching” behavior is short-circuited and the target file is presumed missing.



File references in FileMaker Pro 8

Defining file references

In FileMaker Pro 8, some file references are no longer hidden behind the curtain but instead are exposed to the developer for viewing and editing. A new central area, accessed through the Define File References interface, is where most references for FileMaker Pro 8 files are stored.



The file references located in this area are referred to as “named” file references. Unlike FileMaker Pro 6 – where an existing file reference may be used by script steps, value lists, and relationships through consolidation – in FileMaker Pro 8 file references for script steps are stored separately from file references used by the relationship graphs and value lists (with three exceptions). There are several entry points for this dialog and it is possible to add named file references without ever having visited the “Define File References...” dialog.

Upon conversion of a solution most of the file references you will be dealing with will be located in this central area. The unnamed file references will be stored with the script steps and can be located using the Database Design Report or some of the forthcoming 3rd party tools.



Path Types

Relative path

The relative path in FileMaker Pro 8 is very similar to that in FileMaker Pro 6 with one difference. In FileMaker Pro 6 the relative path to the target file is always “relative” to the current file. If the target file is on a different volume, drive, or server than the current file then the relative path is not stored.

FileMaker Pro 8 behaves in a similar fashion. If the target file is located on the same volume, server, or hard drive then the reference to that file will be stored as a relative path. If the file is on a different volume, server, or hard drive then the appropriate non-relative path is stored.

However, unlike FileMaker Pro 6 the developer has the ability to edit the “relative” path and specify a “full” path using the “relative” path syntax (e.g. “file:/Macintosh HD/Users/Karin/Desktop/filename.fp7” or “file:/C:/Documents and Settings/Karin/Desktop/filename.fp7”). While this path will resolve correctly for the platform specified in the path it is not a “cross-platform” full path. While you could specify both “relative cum full” paths FileMaker Pro 8 will attempt to resolve such a path regardless of the platform that could cause an opening delay. It is best to use the platform-specific full path rather than editing a relative path to make it behave like a full path.

All converted file references consist of at least a relative file path.

Full path (absolute)

The full path in FileMaker Pro 8 is analogous to the full and Windows paths in FileMaker Pro 6; the full path for a file reference takes two different forms depending on whether the target file is on Mac OS X or Windows.

When transferring a file from Windows to Mac OS X or vice versa the full path is not automatically updated, this must be performed manually. Likewise, if a file is to be distributed to both Mac OS X and Windows clients it may be necessary to include both the Mac OS X and Windows path variants to locate a file. While this sounds like more work on the surface it’s actually a good thing. In FileMaker Pro 6 deploying solutions with files located on a central server was difficult to do in a cross-platform environment. With FileMaker Pro 8 it is now possible to specify both the Mac OS X and Windows paths separately and they will resolve correctly on each platform.

Windows Network

The Windows Network path is a variant on the full path. Generally it is inadvisable to locate files on network shares, however FileMaker Pro 8 provides a mechanism for storing those paths. The syntax is similar to the windows full path and is specified as “filewin://ComputerName/Share/directory/fileName.fp7”.

Note that there is no “Mac OS X Network” path type as shared volumes are browsed the same as local volumes; in this scenario you would use the Mac OS X full path.

FileMaker Network

The FileMaker Network path is analogous to the Network path in FileMaker Pro 6. They differ primarily in two ways, the first inconsequential and the other with a larger impact. The first difference, as with the other file reference types, is in the syntax. FileMaker Network paths must start with “fmnet:” and include either the TCP/IP address of the host or the asterisk indicating FileMaker Pro 8 is to search the local network for the file. However the more important difference is that unlike FileMaker Pro 6, in FileMaker Pro 8 you can define more



than one network path to a file. This includes combinations of paths to check a specified server and then the local network as in the following example.

```
fmnet://fmserver.company.net/TEACHERS.fp7  
fmnet:/*/TEACHERS.fp7
```

The flexibility this affords the developer should make deployment of solutions much easier than was previously possible with FileMaker Pro 6 or only available through third-party tools.

Using file references

Named file references

Most file references of interest are now edited in a central location and accessible through the “Define File References...” function of FileMaker Pro 8. These file references are referred to as the “named” file references to denote how their storage and access differs from the “unnamed” file references. With the exception of a few script steps, all file references that can only point to FileMaker Pro 8 files are stored as named references. These include those used by the relationships graph, value lists, and the “Open File,” “Close File,” and “Perform Script” script steps.

Named file references can be defined and modified centrally using the “Define File References...” This presents a challenge to the developer wishing to consolidate and remove file references, as they must manually ferret out each usage of a particular file reference. At this time the only method of doing so post-conversion is by cross-referencing the file references with the output of the FileMaker Pro 8 Advanced Database Design Report or through manual inspection of each table occurrence on the graph, each value list, and each script containing one of the three script steps mentioned above.

Script step references

Aside from the “Open,” “Close,” and “Perform Script” script steps, file references are stored with the individual script step and are referred to as unnamed file references. In general, the file references for script steps that might target a file that is not a FileMaker Pro 8 file are only modifiable by editing the applicable script/script step. Those script steps include:

- Import Records
- Export Records
- Convert File
- Save a Copy as
- Recover File
- Send Mail
- Send Event

Where named and unnamed references target the same file it is incumbent upon the developer to keep them in sync manually when developing and deploying solutions. This includes file references that were consolidated pre-conversion by File Reference Fixer, as they will be split into named and unnamed file references upon conversion to FileMaker Pro 8.



Generally the best option will be to store these as relative file paths as this will allow portability of the solution without need to revisit scripts to redefine references. Until tools are created to help centrally manage these unnamed file references the developer will need to manually inspect scripts and cross reference with the Database Design Report produced by FileMaker Pro 8 Advanced.

Other File references

With FileMaker Pro 8 container fields can now store any type of file, including other FileMaker Pro files. There are three script steps that use File References that are specific to records rather than the file itself. These include the following:

- Insert Picture
- Insert QuickTime
- Insert File

The syntax for the file reference for “Insert File” is similar to that of the named file references, the exception being that there is no “FileMaker Network” syntax. The syntax for the “Insert Picture” and “Insert QuickTime” script steps differs in that they use the words ‘image’ and ‘movie’ respectively in place of ‘file’ for each path type (e.g. “moviemac:/Macintosh HD/Desktop Folder/1984.mov”).

The option to only save a reference to the file rather than embedding the file in the database is specific to a particular field within a particular record and is therefore not editable within the file-specific areas that references are stored. As such, this type of file reference is beyond the scope of this paper.

Search order

The search order of the paths within a file reference is very straightforward and user-definable. Each path for a particular file reference is separated with carriage returns. An example might be:

```
file:TEACHERS.fp7
fmnet:/192.168.0.32/TEACHERS.fp7
fmnet:*/TEACHERS.fp7
filemac:/Macintosh HD/Users/Admin/Desktop/MySchool/TEACHERS.fp7
```

The search order for these file references is simply top to bottom. In the example, FileMaker Pro 8 will first look for the target file in the same folder or on the same host as the current file. Failing to find it with the relative path, FileMaker Pro 8 then searched the network for a FileMaker Server 8 or Host at the TCP/IP address 192.168.0.32 with the file. Failing to find it there, FileMaker Pro 8 then looks for any FileMaker Server 8 or Host on the local network for the file. Only after having failed to find it there will FileMaker Pro 8 look for the file on the volume named “Macintosh HD” but only if the system is Mac OS X. If that hard drive is not available or the file is not at the specified location, FileMaker Pro 8 will stop searching for the file and, unless error messages are being suppressed, will display a dialog asking the user to locate the file. However, the file reference will not be automatically updated with the new location when specified by the user in this manner.



Broken File references

File references that are broken are not automatically fixed in FileMaker Pro 8 when you are prompted to locate the missing file. The developer, or a user with appropriate privileges, will be required to fix the broken file references either by editing the script or by editing the named file reference.

File References on Conversion

Pre-conversion steps

Before conversion it is sometimes appropriate to create “relative only” file references for your scripts, relationships, and value lists. However, this process can be time consuming and generally creates a new file reference, leaving the developer with the task of cleaning them up if the FileMaker Pro 8 conversion utility does not.

One option a developer has of consolidating and removing file references before conversion is through the use of the third-party tools such as the File Reference Fixer utility that is a part of the MetadataMagic solution from New Millennium Communications <<http://www.newmillennium.com/>>. This tool will allow you to alter, fix, consolidate, and remove file references from your solution before running it through the FileMaker Pro 8 conversion engine. Consult the documentation available with the product for more information.

File References in Converted solutions

Upon conversion, FileMaker Pro 8 converts the file references in a solution, naming them with the target file name plus a serial number that increments based on how many references target a similarly named file. File references that are known to not be in use are removed, however duplicates that are in use are not consolidated. The results of this process are written to the conversion log file.

An examination of the conversion log will yield a number of file references that have been converted and others that have been deleted. The first mention of file references is a count of the number that may have been converted. Later in the log you will find references to specific file references that have been deleted. These items will be listed as, “Deleting unneeded file reference: Filename.fp7.” Among this list one might find file references that appear to be for files that are not and purportedly have never been part of the solution. A reasonable question might be, “where did these file references come from?”

During the life cycle of a solution, a file may reference many different files. Often times when creating a new solution a developer will choose to “reuse” files from an existing solution to leverage work already performed. FileMaker Pro 6 does not delete file references when they are no longer in use and, because they are hidden from the developer’s view, this generally does not pose a problem. Other file references, which may still be in use in some obscure and never visited script step, will also be present in the file although never accessed in the day to day use of the system.

Upon conversion, FileMaker Pro 8 deletes file references that it considers to no longer be in use. This means a script, relationship, value list, or other object is not accessing the file reference. If a file reference is still in use by a script step it may still exist in the converted FileMaker Pro 8 file although unless it is one of the “Open File,”



“Close File,” or “Perform Script” steps those references will not be listed with the named file references and will instead be stored with the relevant script step.

It is possible, if a file has been closed improperly or otherwise shows signs of corruption that FileMaker Pro 8 will inadvertently delete a file reference that is still in use or not delete a file reference that is no longer in use. FileMaker Pro 8 does a very good job of converting databases, even those with some structural corruption. The resultant FileMaker Pro 8 files should be corruption free however as with most issues of file corruption, the conversion engine may not satisfactorily convert unhealthy files. In this situation it is likely best that the files be rebuilt in FileMaker Pro 8 if a known good backup is otherwise unavailable.

File paths converted in FileMaker Pro 8 prefer the relative path over the full path (which is platform specific). When FileMaker Pro 8 encounters a full path in an .fp5 file it converts the path using the syntax for the relative path rather than the platform specific syntax for the full path. FileMaker Pro 8 adds relative paths to converted files/file references. This is generally performed when the path to the file is stored as a network path but the file is located in the same folder as other files being converted.

File conversion does not consolidate file references. Because a file can reference two different files with exactly the same name, FileMaker Pro 8 has no way of knowing whether two file references that are both in use actually refer to the same file. There is no harm in having duplicate file references upon conversion although for clarity and performance it is better to edit and consolidate them beforehand.

On conversion, extensions are removed from file names in the file references, yet in newly created .fp7 files the file references, by default, include the file extension.

Converted references search algorithm:

- 1) Reference will look for file without extension.
- 2) Reference will look for file with extension.
- 3) Reference will prompt for file.

Conversion and Runtime files

If .fp5 files converted to .fp7 are then bound to a Runtime application with a custom extension (e.g. “.USR”) the binding engine does NOT add that extension to the file references which have no extensions on conversion. Custom extensions are mitigated.

When file references in newly minted .fp7 files are created they, by default, contain the file extension. When binding these files, FileMaker Pro 8 Advanced forces a change of the extension of the files from .fp7 to a user-specified value (e.g. “.USR”), removes the “.fp7” extension from the file reference paths, and does not add the developer-specified extension.

Why should I edit my file references?

You may find, upon opening a converted solution in FileMaker Pro 8, that there are some performance drawbacks, especially when working with related files and layouts. Much of this potential performance degradation can be attributed to multiple file references and, in a network environment, using the wild-card (*) network address to specify a hosted file.



The first time FileMaker Pro 8 requests a file, it needs to resolve the file reference to locate the file. For each subsequent request for the file through that reference, as long as the file remains open, FileMaker Pro 8 will not need to resolve the file reference again and will be able to access the file much faster. Optimizing your file references therefore has a direct impact on solution performance. By consolidating your file references you ensure that FileMaker Pro 8 will only need to resolve the reference once. However, there are further performance implications to file references. In FileMaker Pro 6 it is common, when creating references to network files, to allow FileMaker Pro to use the “local network” reference for the file (as indicated by the asterisk). This method is quantifiably slower than directly referencing the server IP address or DNS name and adds time to the resolution phase. For unconsolidated file references, this creates a delay several times over. Search order is also important. If you have multiple paths in your search order placing the full paths before the relative and network paths may cause performance degradation for files which are deployed on FileMaker Server 8 or otherwise are accessible via the relative path.

Given the nature of file references in FileMaker Pro 6 and FileMaker Pro 8 the developer must be concerned about their state in both deployment and development scenarios. Where FileMaker Pro 8 improves on this is that the developer now has explicit control over the search order used when locating files. Consolidating file references makes the process of modifying these references easier. To avoid the problems of FileMaker Pro 8 not being able to locate files or locating the wrong copies of files it is preferable that the developer explicitly specifies where to locate the necessary files. This can be in the form of setting only the relative path (in cases where the files will be deployed together in one folder or on one server) or an absolute path in cases where the files will be deployed across multiple servers or where there may be both local and hosted files as part of the solution.

The choice of file path to use is based on the likely development and deployment scenarios for the files. Upon conversion you may find that you want to modify your file paths for consistent operation and better performance. There are a few guidelines for modifying file references that should help you choose the correct path. For solutions where all of the files are located in the same folder or on the same server using only the relative path will provide you with the most consistent, accurate behavior and the best performance. For solutions that have some local files and some hosted files there will need to be more ‘nuanced’ references. Where the hosted files reference each other those references should be relative paths. References from the local files to the hosted files should contain the network address if possible, otherwise use the wild-card network. References from the hosted files to the local files will need platform specific full paths. This latter scenario is the most difficult to specify as there is no guarantee what the local path will be for the files. There is no mechanism to specify a “dynamic” path programmatically. When specifying a full path it is preferable to use the “filemac:” and “filewin:” syntax rather than the “file:” syntax to avoid performance degradation as FileMaker Pro 8 attempts to resolve a path that is not valid for the current platform.

How can I consolidate my file references?

Because FileMaker Pro 8 now exposes the file references to the developer, it is possible to consolidate most of the file references within a solution manually.

Depending on the size, complexity, and number of files in the solution this can be a daunting task. The “Define File References...” dialog does not indicate where a particular file reference is being used. Instead the

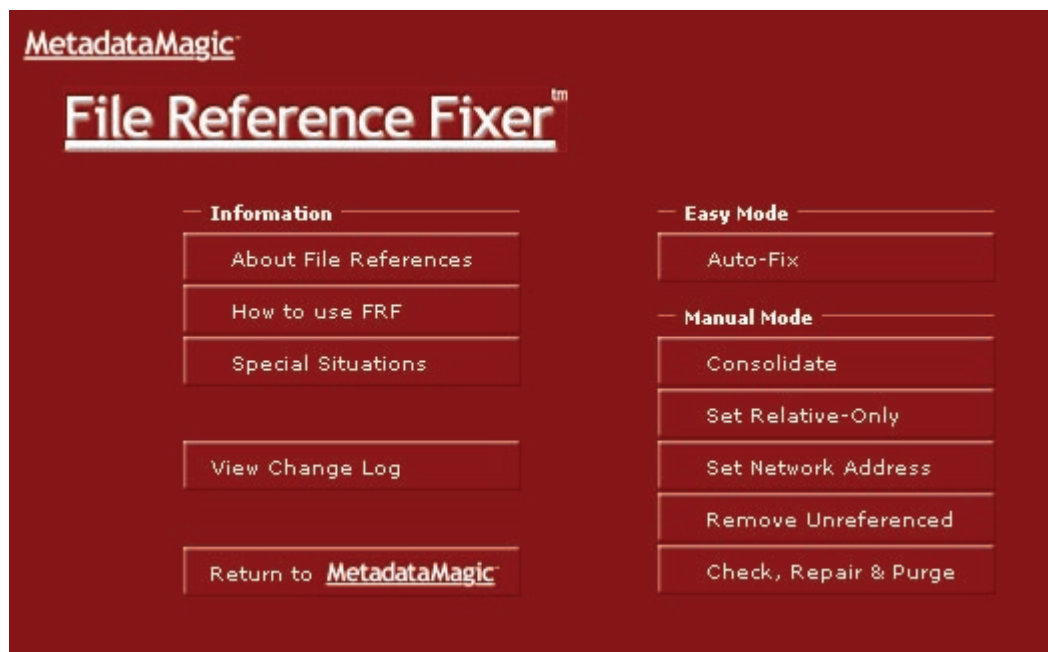


developer will need to create a Database Design Report of the solution and manually examine each area for file references. This can be aided by renaming obsolete file references with a unique name (e.g. “DELETEME1”) and searching the output of the DDR for occurrences of the file reference name. The developer will then need to modify the sections of the solution that make reference to the obsolete file reference and re-point it to the “correct” file reference. Note, however, that unnamed file references (e.g. Import Records) cannot be consolidated with named file references.

One suggestion is to first define a “master” file reference and then have all the modified/edited references point to the new reference.

MetadataMagic, from New Millennium Communications, provides as part of its functionality, a utility called File Reference Fixer. This tool exposes the existing file references in an .fp5 solution along with where the file references are being used.

File Reference Fixer includes features to consolidate redundant file references, edit network addresses, and set file references to relative-only, along with a single-click “Easy Mode” which enables consolidating all file references and setting them to relative-only for use when all files in a solution reside in the same folder or on the same server.



Consolidating file references before conversion takes a lot of the guesswork and busywork out of the process of ensuring file references are correct throughout a solution. In addition, the documentation included with File Reference Fixer explains some of the special scenarios regarding editing and consolidating file references that are beyond the scope of this document.



Conversion after a solution has been processed with File Reference Fixer will lead to a lot fewer problem areas and, with most solutions, will obviate the need to manually re-specify file references after the fact entirely.

Moving forward

As you modify your converted solution it will become increasingly easier to maintain your file references in a consistent manner. Named file references allow the developer full control over where and how to locate files in a solution. They also make it more difficult for the developer to duplicate file references as the developer makes the choice whether to reuse an existing file reference or create a new one. The file reference system in FileMaker Pro 8 also makes it easier to deploy your solutions in various scenarios. Partitioning a database across multiple servers is now accomplished by modifying one or two file references in each file and generally does not require that they be modified while deployed as is necessary in FileMaker Pro 6. This adds up to create solutions that are more consistent, reliable, and are easier to document for the developer and end user.

About the author

CORN WALKER works at inRESONANCE, Inc., a FileMaker Solutions Alliance Partner based in Northampton, Massachusetts. inRESONANCE is a strategy and technology consulting firm serving schools and non-profit organizations. iR provides customizable FileMaker Pro solutions, FileMaker Pro training, and web-design and integration services to clients worldwide. Visit <http://www.inresonance.com> for more information.



Scripting Issues Encountered When Migrating to FileMaker Pro 8

This paper explores some of the scripting issues involved with converting database solutions from FileMaker Pro 6 to FileMaker Pro 8. It discusses areas that will require particular attention from the developer to help ensure a smooth migration, including the issues of “focus” and “context”, and the behavior of Startup Scripts and Closing Scripts.

Focus and Context

For our purposes, context is defined as the starting point for any given operation. It includes the file, window, layout and found set that are current at the moment when a script (or other operation) begins. Focus is a similar concept, and refers to the currently active window/file/layout/found set, etc. Thus, as a script begins, it will be affecting the current context, but the focus may well shift as the script progresses (say, if Go To Related Record, Perform Script [External], or Select Window are called).

FileMaker Pro 8 introduces many new features that FileMaker developers have not encountered before, and these changes will likely pose a learning challenge. In particular, FileMaker Pro 8 allows for multiple tables per file, and for multiple windows to be displayed per file. These two basic concepts change the established developer’s understanding of FileMaker Pro behavior with regard to focus/context and script behavior, because FileMaker developers have never had to put much thought into the current context or focus in previous versions of FileMaker Pro.

In previous versions of FileMaker Pro, the focus or context of a given operation was implicit in the design, since FileMaker Pro has supported only one table per file and only one window per file. These facts, coupled with the FileMaker Pro single-threaded architecture, ensured that the developer was always aware of the precise context of where a script was functioning. A script would only operate on one window of a given file because that was the only window that could exist. The script would also operate on the currently displayed layout within the active window/file, on the current Found Set in that file. Thus, a developer always knew what the context was based on the only window of a given file.

FileMaker Pro 8 offers a significantly different development environment to the developer by allowing for multiple tables per file, and for multiple windows to display for the same file. Scripts do not belong to a specific table, but to the file as a whole. Thus, if a script in File A needs to operate upon several different tables within File A, there is no need to perform external script calls to other files. **The same script can operate on several tables, since many tables may be contained within the same file.** In addition, since a given file can have Table Occurrences for tables that exist in external files, a single script can affect the data residing in separate files without performing any external script calls. Developers may no longer think of a file as a table; these are two very distinct concepts in FileMaker Pro 8.

FileMaker Pro 8 allows multiple windows to be open for the same file. A single script can change the focus between several windows for the same file without calling any other scripts. Merely bringing another window to the foreground does not imply a change in focus to another file or another table, unlike earlier versions



of FileMaker Pro. Thus, it does not imply a need to call another script. Furthermore, previous versions of FileMaker Pro would perform an implicit Refresh Window when a scripted operation was completed (that is, when the last script in a chain of scripts was finished running). This would always bring the last file in the chain to the front. **Scripts no longer perform an implicit Refresh Window when they complete in FileMaker Pro 8.**

Implications

As a consequence of these changes, the former behavior of several key script steps are not maintained in FileMaker Pro 8. Script steps that would automatically change focus or context in previous versions of the product no longer behave as they used to. For instance, in previous versions of FileMaker Pro, the Perform Script [External] script step at the end of a script would make the external file active. In most cases, this would bring the window of that external file to the front, effectively changing the focus to the new file. This behavior is no longer supported.

Similarly, the Go to Related Record (GTRR) script step itself no longer necessarily changes window focus, though it does change the context by changing the current layout and found set of records. In FileMaker Pro 8, the related table may be in the same file as the current table and thus the layout and the displayed records can change without a new window. In earlier versions, similar behavior was only possible in the special case of a self-join relationship.

FileMaker Pro 8 has a new script step, Select Window, which can be used to explicitly activate a given window. During conversion of an existing system, FileMaker Pro 8 will insert a Select Window script step after Perform Script [External] or Go to Related Record steps under certain circumstances (see below) in an attempt to approximate the behavior of previous versions. However, it may be necessary to modify or delete this added script step in the converted file to obtain the desired behavior.

For more information, read the documentation in the “Converting FileMaker Databases from Previous Versions” .pdf that is included with FileMaker Pro 8 and FileMaker Pro 8 Advanced.

Scripts that would explicitly open another file in FileMaker Pro 6 may behave differently in FileMaker Pro 8. In previous versions, if a script included an Open File script step followed by more script steps, the other file would open and then the window of the calling file would return to the front while the remainder of the script ran. In FileMaker Pro 8, if a script opens another file and then continues, that other file’s window will remain active while the calling script continues to run in the background file. It may be necessary to insert Select Window [Current Window] script steps after each Open File script step in a converted solution to emulate prior behavior. FileMaker Pro 8 will not automatically insert these Select Window script steps for you.

It should be noted that there is a big difference between the Open File script step and the New Window script step. When a script performs a New Window step, by default that new window will display the same layout and found set as the previously active window. The new window, therefore, displays the same table, and becomes the active window. The script will continue to operate in this new window unless the developer explicitly returns focus to the original window by using Select Window.



The Open script step (which becomes the Open File script step after conversion) has changed behavior in another significant way. In previous versions of FileMaker Pro, you could create a script to open a remote file by using either Open or Open Remote script steps. FileMaker Pro 8 can only open a remote file via Open File if the file was specified using the FMNET format (fmnet:/hostIP:FileName.fp7). Since this format was not available in previous versions, it means that any script that opens a hosted file via the Open script step needs to be modified post-conversion. Either the file needs to be re-specified using FMNET syntax, or else the Open Remote script step needs to be used in place of Open File. It would probably be easiest to simply modify all such scripts to substitute Open Remote for Open steps before converting the file, rather than trying to change them after conversion.

As previously stated, developers will need to take a more proactive role in managing the context of their solutions than they did in previous versions. Developers will find themselves relying heavily upon the Select Window script step. In previous versions of FileMaker Pro, developers could use the Open script step to bring another window to the front, even if that file was already open. In FileMaker Pro 8, this is best accomplished with the Select Window script step.

Script modification during conversion

During conversion of existing databases, FileMaker Pro 8 will insert the Select Window script step after certain specific script steps that used to change window focus in previous versions. One of those script steps is Go To Related Record. If the GTRR step references an external file, and is not immediately followed by another GTRR or a Perform Script (External) step, then FileMaker Pro 8 will insert a Select Window step. The Select Window step will either select the current window (effectively returning focus to the calling script's table) or else will specify the external file's window (if the related table resides in an external file).

Similarly, FileMaker Pro 8 will insert a Select Window step after certain Perform Script [External] steps. FileMaker Pro 8 will insert a Select Window step, specifying the window name of the external file, if the Perform Script [External] step is the last step in the script (not including End If, End Loop, or Comment) or if it is followed by an Else step and the corresponding End If has no other steps after it. FileMaker Pro 8 will insert a Select Window [Current Window] after a Perform Script [External] or a GTRR step if the step is not the last step in the script (other than the steps above) and is not followed immediately by another Perform Script [External] or GTRR step.

This can have many unforeseen consequences, and cause the focus to remain on the “wrong” file post-conversion. For instance, consider the following scenarios:

Scenario 1: Cascading Scripts

Suppose you have three files: File A, File B, and File C. File A runs a script that has a single step, to perform an external script in File B. The script in File B has a single step to run a script in File C, which switches to a layout in C.



In FileMaker Pro 6 and earlier, the scripts would end up with File C as the active file. But when these files are converted to FileMaker Pro 8, the scripts will end up with File B in the foreground. Why?

During conversion, FileMaker Pro 8 will insert a Select Window [External File's Window] step after each Perform Script [External] step. Thus, after File A calls the script in File B, it runs a Select Window script that brings File B's window back to the foreground (after running File B's script).

The developer can fix this admittedly simple example rather easily, by deleting the Select Window steps. However, with more complex scripts, it can be more difficult to determine where those steps are, and remove them appropriately.

Scenario 2: Halting Scripts

Imagine you have 2 files: File A and File B. File A calls a script in File B, which ends with a Halt Script step. In FileMaker Pro 6 and earlier, the scripts would end with File B active, because the Halt Script step implied a focus change to the file that executed it. But Halt Script no longer forces a change of focus in FileMaker Pro 8. The result is that focus will remain on File A after conversion. Note that FileMaker Pro 8 will add a Select Window [File B] after the Perform Script [External] in File A's script, but this will not change the focus from File A to File B in this example. Since File B's script runs a Halt Script step at the end, the Select Window in File A's script will never be performed. Unless File B's script explicitly performs a Select Window [Current Window] to force File B's window to the front, then File B's script will run entirely in the background, and focus will never leave File A.

Scenario 3: Hidden Windows

Suppose you have 2 files: File A and File B. The script in File A ends with a call to a script in File B, which ends with a Toggle Window [Hide] step. In FileMaker Pro 6, this construction would end with File A in the foreground, because when File B's script is done, it will hide itself. But after conversion, this will result in File B remaining in the foreground. Why?

During conversion, FileMaker Pro 8 will add a Select Window [External File's Window] step after the Perform Script [External] step. In our example, this Select Window step will bring File B back to the foreground after it hides itself, effectively "unhiding" File B.

Miscellaneous Issues

The issue of context and focus in FileMaker Pro 8 will give rise to other basic issues with converted files. Here, we will attempt to outline some of these issues and suggest possible approaches to solving them.



Multiple Tables per File and Layouts

In FileMaker Pro 6, it was obvious which “table” a given layout was associated with, because there was only one table per file, and a given layout was intimately tied to the file in which it was defined. Thus, there was no ambiguity about which table was currently active.

This has changed in FileMaker Pro 8, as we have seen, because each file can have multiple tables. Since a given file can have multiple layouts as well (as has always been the case), ambiguity arises regarding the context of each layout in a given file. Which table occurrence does a given layout belong to?

In the Layout Setup dialog box in FileMaker Pro 8, you can specify which table occurrence the current layout shows records from. Thus, each layout is intimately tied to a specific table occurrence in FileMaker Pro 8. This is a fundamental part of the layout’s definition. But when you are working with the layout outside of the Layout Setup dialog, there is no obvious indication of which table occurrence each layout is associated with. This has an impact on how scripts function, because unless a script explicitly changes the context to another file, window, or layout, that script will run in the current window, on the current layout (and thus affect the current table and Found Set in that file).

This situation can be resolved by a combination of several techniques. First, it will be essential to develop a good, consistent naming convention for layouts that clearly identifies which table each layout is associated with. It will also be important to incorporate explicit changes of context in each script, rather than relying on FileMaker Pro 8 to change context for you. Even when the context does not need to change (that is, when the script needs to run on the current table, with the current found set, on the current layout and in the current window), it will be advisable to include Select Window [Current Window] steps, for consistency and reliability.

The issue of a layout naming convention becomes even more important when you understand that FileMaker Pro 8 allows for duplicate layout names within the same file. Imagine a user’s confusion when they begin working with a solution that has three separate layouts all named “Invoices” (and each associated with a different table occurrence)!

Multiple Windows per File

In FileMaker Pro 6, it was obvious which file a window was displaying, because each file could have one and only one window open. When you saw a window titled “Invoices.fp5”, you knew that it was displaying the Invoices file.

This has changed in FileMaker Pro 8. Now, a given file can have many different windows open, and the developer controls each window name. Thus, there is now no obvious way to determine which file a given window belongs to. When combined with the fact that a file can contain many tables, the potential for confusion can seem overwhelming.

In the same way that a consistent naming convention can help with the issue of layouts in a file, a similar naming convention can help when dealing with windows. By default, FileMaker Pro 8 will try to name each window



after the file that it is displaying. (Note that this has nothing to do with the table or layout being displayed!) But the developer can rename any window via scripting, and likely should. In this way, the developer can avoid the confusion that might otherwise arise.

The naming convention defined by the developer will, by necessity, have to be applied after conversion, since there is no way in FileMaker Pro 6 to either generate new windows for existing files, nor to rename windows.

Opener Scripts

Like previous versions, FileMaker Pro 8 will allow you to specify a script to activate when the file is opened. Such scripts are often referred to as Startup Scripts, but a better name would be *Opener Scripts* because the behavior of the function has changed. In previous versions, an opener script would fire when a file was first opened, except under certain specific circumstances (like if the file were opened by a relationship). This is still generally the case in FileMaker Pro 8, but there are some circumstances where an opener script could fire unexpectedly.

For instance, if a file was opened in a hidden state due to a relationship or value list, and you open the first new window for that file, its opener script will fire when the new window is drawn. This could not happen in previous versions because each file could have only a single window. Note that this circumstance only occurs when you are creating the first new window for a file that was already opened via script or relationship.

You can avoid this by conditionally running your opener scripts based upon global data. At the start of each opener script, check to see whether a global has been set, and abort the script if it already has. At the end of the opener routine, set the global to prevent the opener script from running a second time. In addition to this, do not rely upon FileMaker Pro 8 to implicitly open your files. Instead, have the opener script for your primary file explicitly open each file in the solution, and set the global flags so that each file's opener script will not fire after it's opened. Many developers have already adopted this kind of approach, but in FileMaker Pro 8, it will become even more important.

Closing Scripts

Similarly, the functionality of closing scripts has changed somewhat in FileMaker Pro 8. In previous versions, it was obvious when a closing script was going to fire: it happened when you closed the file. But FileMaker Pro 8 supports multiple windows per file. Furthermore, it is possible for a file to be open without any open windows at all!

The behavior of FileMaker Pro 8 works this way: when you close the last open window for a given file, that file's closing script will run. This will happen even if the file itself remains open (say, due to related fields being displayed on a layout in another window). Yes, it's possible to close all the windows for a file without closing the file itself. The rules by which FileMaker "pulls open" files have changed, and it may not be able to close a file once it has been referenced during a session by other files which are open, even with a Close File script step. It is important to note that if you close the last open window for a given file (thus causing the closing script



to fire), then reopen a new window for that file and close it again, the closing script will fire again. The closing script for a given file will execute each time that the last open window is closed for that file. Also, keep in mind that when a closing script does fire, it will be running from the context of the closing window. Thus, it will operate on whatever found set and layout are currently active in that window.

The concept of opening or closing a window should not be confused with opening or closing a file, and also should not be confused with the idea of accessing a table. These are all distinct concepts in FileMaker Pro 8.

About the author

Darren Terry is Director of FileMaker Development for Pacific Data Management Inc. He is a four-year veteran of FileMaker Technical Support, and was formerly the Technical Liaison for Developer Relations at FileMaker, Inc., where he worked closely with the FileMaker Solutions Alliance and was responsible for representing the developer community in key team meetings. Darren has been a featured speaker at the annual FileMaker Developer Conference, and has written articles for FileMaker Pro Advisor Magazine and ISO FileMaker Magazine. Darren can be contacted at PDM at (408) 283-5900 or at darren@pdm-inc.com.



Security and Access Privilege Issues

Part One: First Examination of Converted Files

When a developer, an IS/IT/DBA manager, or a user converts a FileMaker® Pro file from an earlier version to FileMaker Pro 8, the security features contained in the earlier versions will also be converted. *How* these features convert and how they can be used post-conversion depend on a number of variables:

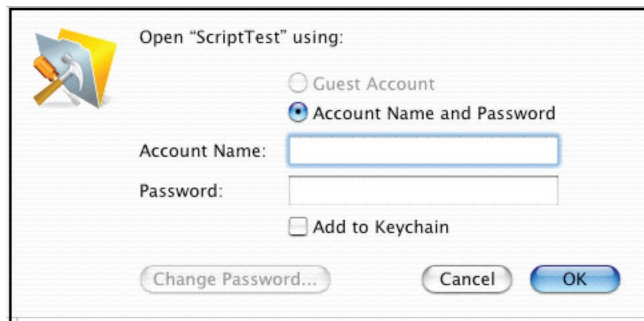
- Whether the security schema in the old files was well-formed;
- Whether the Groups feature was employed, and if so, whether it was employed as expected by the conversion tool;
- Whether the Groups structure, specifically the function `Status(CurrentGroups)` was employed for any type of conditional testing or access control; and,
- Whether the old files used the built-in FileMaker Pro security system or whether they employed some other system such as a “log-on” file.

A password in an older version FileMaker Pro file will convert to an Account with the Account Name the same as the old password and with the Account Password the same as the old password. This is a fundamental conceptual point to remember. It has implications for on-going file security, including initial access to the converted file.

Irrespective of any of these considerations, when initially opening a converted file that includes passwords, the Open File dialog proposes as the Account Name the User Name Specified in the FileMaker Pro 8 Application Preferences¹. This name in the Preferences most likely comes from the name supplied during the installation process. It is most likely not the correct Account Name. So, as a first step, if there is a value already supplied in the Account Name dialog, it should be overwritten with the correct Account Name, the name that is the same as the password.

Since that Account Name is shown in the clear when entered, potentially compromising the password, FileMaker Pro 8 allows you to leave the Account Name blank, and then just enter the password. An early first post-conversion step may be to change the Account Name to something other than the password or, alternatively, to change the password itself. Either will protect the confidentiality of the information; one or the other should be done. The following illustrates the log-on dialog box:





If you leave the Account Name blank and then enter what was the “master password” into the password field of this dialog box, the file should open with complete access. This will allow a developer to examine the converted security schema and Access Privileges. Under the *File Menu*, select Define→Accounts & Privileges to reveal a three-tab interface, one of which lists the Accounts and another of which lists the Privilege Sets in the new file. The third tab is for Extended Privileges, a new feature (but one with some familiar options) in FileMaker Pro 8. The tabs look something like this:



When exiting the Define Accounts & Privileges area, you may be prompted for an Account Name and Account Password. In that instance enter the password into both fields unless you changed the Account Name. The Privilege Sets are where the rules are made that enforce the security schema of the files. FileMaker Pro 8 attempts as faithfully as possible to duplicate the security features from the old files. However the new version does make some changes depending on those four conditions outlined above.

In order to understand what happened during conversion and in order to be able to deal with the consequences of conversion, developers, IS/IT/DBA managers, and other users need to understand how security works in the new version of FileMaker Pro 8. The security schema consists of three main parts: *credentials*, *authentication*, and *Privilege Sets*. A user’s credentials are the Account Name and the Account Password. When the user’s credentials are deemed authentic, the user is allowed access to the database with a set of privileges defined by the Privilege Set attached to the Account. There is one and only one Privilege Set per account. This figure illustrates the entire process:





The conversion process attempts to map the privileges associated with the password in the old file to the newly created Privilege Set in FileMaker Pro 8. In doing so it takes into account both the password defined privileges and the Groups defined privileges from the older version file. It consolidates identical instances of passwords and Group access privileges into a single Privilege Set with multiple accounts attached. This has implications for any older version file that has passwords linked to Groups. It also has implications for what was formerly called the “master password” in FileMaker Pro 6 and earlier versions.

In FileMaker Pro 8, the passwords are not stored in the file. Neither are they visible in the User Interface. When entered they are obscured; they remain obscured. Additionally, there is an intentional mismatch between the number of characters in the password and the number of characters seen in the obscured display in the User Interface.

Important note: FileMaker Pro 8 passwords, unlike those in earlier versions, are not retrievable, by FileMaker, Inc. or third-party tools. Do not forget your password, particularly the one attached to [Full Access] accounts.

All “master passwords” are converted into Accounts with the Account Name and the Account Password the same as the old “master password.” All, and that means **all**, such Accounts are then attached to the default [Full Access] Privilege Set. A developer **cannot** create a [Full Access] Privilege Set; you must use the default one. Neither can you delete or duplicate the [Full Access] Privilege Set. The default [Full Access] Privilege Set can have some additions to it however through the Extended Privileges options. There are also two other default subordinate Privilege Sets: [Data Entry Only] and [Read–Only Access]. *We recommend that neither of these ever be used*; developers should create their own custom Privilege Sets for these functions. The default subordinate Privilege Sets contain privileges that may be both inappropriate and surprising for their respective Privilege Set’s name, such as granting export privileges to Read–Only users.

It is in the conversion of subordinate passwords from the old files to accounts attached to subordinate Privilege Sets in the new ones that developers will engage in pre–conversion and post–conversion adjustment and amelioration.

Part Two: Conversion Guidelines

There are several core scenarios for describing the security schema of an older version FileMaker Pro file:

- No passwords
- Master password only



Master and subordinate passwords with No groups
 Master and subordinate passwords with one or more Groups
 Master and subordinate passwords with one or more Groups, and mal-formed schema².

Examine your older version files to see where your solution fits in this broad division. It is possible that one file in a multi-file solution will fall into a category and another file in the same solution will fall into a different category. In that case, you may wish to consider improving that condition before converting, or you can wait until after converting. Here are twelve specific, more detailed scenarios that describe a wide range of possibilities, although not every possible one:

No passwords; No Groups
No passwords; One or more Groups {somewhat unlikely, but possible}
Master password only
Master password; subordinate passwords; auto-enter; No Groups
Master passwords; subordinate passwords; No Groups
Master passwords; subordinate passwords; Groups assigned
Master passwords; subordinate passwords; Groups unassigned
Master passwords; subordinate passwords; Master passwords have unique Group; others unassigned
Master passwords; subordinate passwords; Master passwords have unique Group; others assigned to Groups
Master passwords; Subordinate passwords; Groups assigned one or more passwords. Some Groups have identical privileges and the passwords assigned to them have identical privileges.
Master passwords; Subordinate passwords; Groups assigned one or more passwords. Some Groups have different privileges. Passwords assigned to a specific Group all have the same privileges.
Master passwords; Subordinate passwords; Groups assigned one or more passwords. Some Groups have different privileges. Passwords assigned to a specific Group all have the same privileges. However the security system is malformed .

Here is what to expect on conversion. We want to emphasize that close post-conversion checking of Privilege Sets is required, because the possibility does exist for both conversion errors and unpredictable results.

FileMaker Pro 8 takes the FileMaker Pro 6 passwords and the FileMaker Pro 6 Groups, attempts to combine their privileges, and creates a Privilege Set that represents the union of those privileges. The name of that new FileMaker Pro 8 Privilege Set may or may not be consistent or identical to the name of the FileMaker Pro 6 Group. *This has the potential, post-conversion, to disrupt the intended result of any calculation test based on that specific Group Name.*



The privileges identified by the new FileMaker Pro 8 Privilege Set come from both the FileMaker Pro 6 password and the FileMaker Pro 6 Group. FileMaker Pro 8 will attempt to map the access bits (or key values) into a new Privilege Set and create a descriptive name. As stated, these access bits come both from the Password itself and from the Group. From Passwords they are the various checkboxes, the popup list for “Available menu commands”, and the three Record Level Access calculations. From Groups they are the privileges associated with layouts and fields (Accessible, Not Accessible, or Read Only).

The following is a pseudo-code sequence for the process by which this conversion occurs (identified by Ernest Koe, inResonance):

```
For each password:
  Figure out all the access bits for this password
  Does this set of privileges already exist?
  If Yes
    Use it
  Else
    Create the set
    Name the set

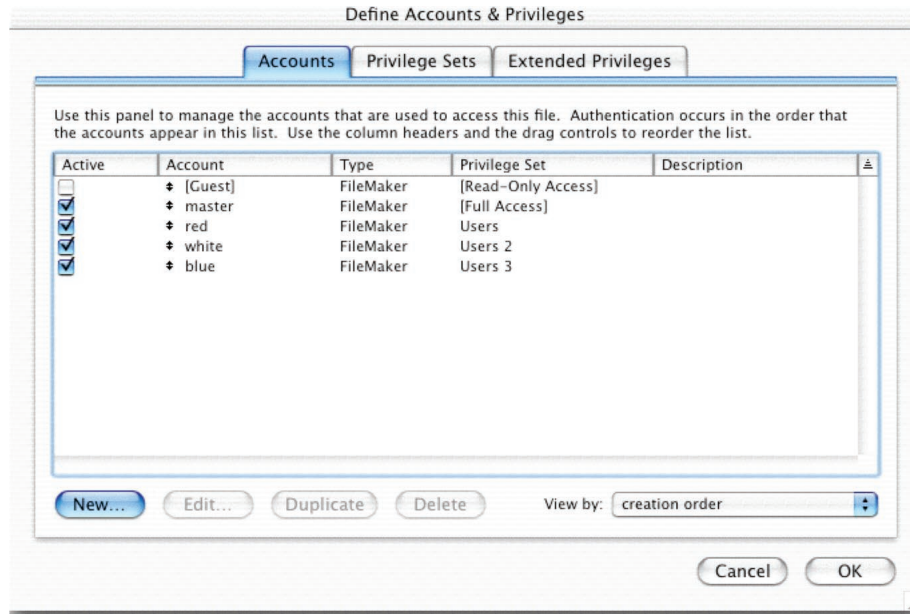
Are there fp5 groups associated with this password?
  If Yes
    Name the set using all groups...(group1/group2/.../group[n])
  Else
    Name the set “Privilege set [n]”
  End if
End if
```

In a nutshell, FileMaker Pro 8 names the Privilege Set using the Group names associated with the password (if applicable) and moves on. If it runs into a password that has the same set (but different group name), it will use that set regardless of whether it was actually assigned to a different Group in the original FileMaker Pro 6 file.

When multiple passwords were assigned to a single Group and all those passwords had identical privileges, the multiple Groups are consolidated in FileMaker Pro 8 to a single Privilege Set named for the first Group of the FileMaker Pro 6 file³.

Developers of FileMaker Pro 6 solutions frequently assigned passwords with dissimilar privileges to a single Group. FileMaker Pro 8, on conversion, will create multiple Privilege Sets and assign Accounts to them based on the FileMaker Pro 6 password names. Consider three passwords with dissimilar privileges: *red*, *white*, and *blue*. If the FileMaker Pro 6 Group was named, for example, *Users*, then there will be instances in the fashion of *Users*, *Users 2*, *Users 3*, and so forth, for the Privilege Set names. If two passwords had identical privileges, they will both be assigned to the same Privilege Set in keeping with the general FileMaker Pro 8 attempt to reduce file detritus.





Developers of FileMaker Pro 6 solutions also frequently assigned passwords to multiple Groups.⁴ When the old Groups had dissimilar privileges, the resultant FileMaker Pro 8 Privilege Sets will also have dissimilar privileges. This could cause issues in the FileMaker Pro 6 environment; however in FileMaker Pro 8 most of these issues have been rectified.

For simplicity's sake, assume only Groups have access bits⁵:

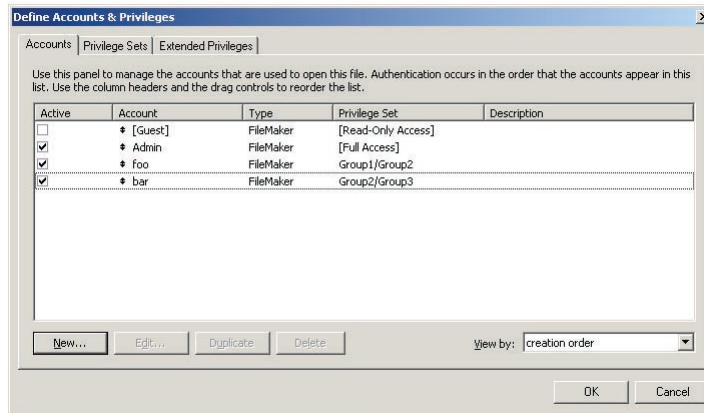
Group 1 - {-bcd-f}

Group 2 - {ab--e-}

Group 3 - {a-c-e-}

Assume also two passwords, foo and bar. Password foo is mapped to Group 1 and Group 2. Password bar is mapped to Group 2 and Group 3. From foo's point-of-view, the union of privileges are {abcdef}. From bar's point-of-view, the union of privileges are {abc-e-}. FileMaker Pro 8 then produces two accounts, foo and bar. Foo is assigned a Privilege Set with the bits {abcdef} and bar is assigned to a Privilege Set with the bits {abc-e-}. The Privilege Set for foo is named "Group 1/Group 2" and the Privilege Set for bar is named "Group 2/Group 3".





Part Three: Implications Of These Changes

If there are Groups with identical privileges but different passwords with identical privileges in FileMaker Pro 6, the conversion process, in keeping with the “anti-detritus” rule, will consolidate these into a single new Privilege Set with all the accounts converted from FileMaker Pro 6 passwords attached to that single, consolidated new Privilege Set. This can lead to some unexpected results in instances where Group names were used in various conditional tests in FileMaker Pro 6, `Status(CurrentGroups)` becomes `GET (PRIVILEGESETNAME)`. First, and foremost, all “master passwords” from FileMaker Pro 6 are now accounts attached to the [Full Access] Privilege Set. Second, multiple Group names may now have been consolidated into a single Privilege Set. Thus, for example, a ScriptMaker script step syntax based on the “master password” that said:

```
[If (PatternCount, Status(CurrentGroups), “developer_only”))]
```

that evaluated to True in FileMaker Pro 6 will **fail** in FileMaker Pro 8 because it now reads `[PatternCount (GET(PRIVILEGESETNAME) ; “developer_only”)]` and the Privilege Set name is [Full Access]. Similarly, in a situation where passwords with identical privileges have been individually mapped to different, identical Groups, a test that read, for example:

```
[If (PatternCount, Status(CurrentGroups), “SalesMgr”))]
```

that evaluated to True in FileMaker Pro 6 may now fail in FileMaker Pro 8 because the Group “SalesMgr” has been consolidated along with such Groups as “MarketingMgr” and “OperationsMgr” into a single Privilege Set named, for example, “MarketingMgr.”

Developers must check converted files to identify and to correct these potential anomalies. The following table lists several places where such tests might be found, although not necessarily all such instances.



Conditional Scripting [If...]	Calculation field formula
Record Level Access tests	Auto-entered calculated values
Field validations by calculations	Conditional value lists
Set Field and Insert Calculated Results ScriptMaker script steps	AppleScript or VB Script generated wholly or partially from calculated fields
Replace Function	Show Custom Dialog function

Additionally, the [Status(CurrentUserName)] function used in earlier versions in scripts, calculations, and Creator/Modifier Name in Field Definitions options tab should be replaced for greater accuracy and security with the new Account Name. This does not happen automatically during the conversion process. Note however that if the Status(CurrentUserName) construct was used for certain Record Level Access (RLA) tests or in other literal strings, developers will need to synchronize the User Name with the Account Name, either by making the Account Names the **exact** same as the User Name, or by going through the Creator and Modifier fields and changing the prior User Name to the new Account Name. Note also that if the Account Name is subsequently changed, that RLA will fail.

Pre-conversion correction can address some of these issues as well, especially those associated with any unique Group that was assigned to an earlier version's "master password." Before conversion and by using a tool such as New Millennium's MetadataMagic or the FileMaker Pro 6 Developer Database Design Report, or possibly the Analyzer tool from Waves in Motion, identify all instances in scripts and other locations where that unique Group name was employed. You are searching specifically for **literal text strings**. In the Groups definition area change the name of the "master password" only Group to *Full Access*. Then after having located instances where the test based on that name is employed, change the literal string to read *Full Access*.

As a result, when converted, the newly corrected old syntax of [If (PatternCount, Status(CurrentGroups), "Full Access"))] will be converted to [If [PatternCount (GET(PRIVILEGESETNAME) ; "Full Access")]]. The test will then evaluate correctly, because the FileMaker Pro 8 Privilege Set name is *Full Access*. Incidentally, the brackets [] used in the FileMaker Pro 8 naming convention for this Privilege Set, viz. [Full Access] are not evaluated in the conditional test. You can include them, or leave them out.

When FileMaker Pro 8 has created the new Privilege Sets in the converted files, developers will want to carefully review the privilege bits set in each one. In most instances, developers will likely want to change the privilege rules found in the old files to take advantage of the greater power and granularity found in FileMaker Pro 8. This could entail simply removing all the converted Privilege Sets and starting over according to role-based access rules.

Part Four: Mal-formed FileMaker Pro 6 Security Schema

I have referred in several instances to *mal-formed schema* when talking about earlier versions' security systems. There are several areas where developers may need to clean up their security schema before conversion.



FileMaker Pro passwords in earlier versions must be unique but are not case sensitive, and this results in mixed cases of the same password being indiscriminate. For example “Master” and “master” and “MasTER” and “MASTER” all are viewed as identical. FileMaker Pro 6 and earlier versions do not distinguish among these options; all are viewed as identical. Thus each will open files designed to use the same passwords whether directly or through “inheritance” of the password when the files are called by relationships, “open file” ScriptMaker™ steps, value lists, etc.

If the password is “master”, then wherever it appears it should be *precisely identical*. Earlier versions of FileMaker Pro forgave this behavior. However FileMaker Pro 8 Account Passwords are case sensitive.

The same issue applied to older version Group names, but here the situation becomes more complex. Groups names were not required to be unique. Nor were they case sensitive.

Group name creation in previous versions of FileMaker Pro could have resulted in the creation of duplicate or case *insensitive* Group names with unexpected and detrimental results in FileMaker Pro 8. If you expect an action to occur based on the password’s being assigned to the Group “managers,” and there are duplicate instances of that Group, then users may enjoy higher privileges than you anticipated or, conversely, they may lack sufficient privileges. Suppose that the first instance of “managers” is associated with a password that has delete records privilege; suppose further that the second instance of “managers” is associated with a password that lacks that privilege. You see rather clearly what the potential for error is. Requiring that Group names be unique within a given FileMaker Pro 6 file and be exactly identical among all files in a multi-file solution both makes testing of conditions much easier.

Here are several specific pre–conversion correction steps for mal–formed security schema:

- Assure that all instances of a given password throughout a multi–file solution are *exactly* the same, respecting case sensitivity;
- Assure that each Group name is also exactly the same, respecting case sensitivity, and that group names are unique; and,
- Examine files for instances of passwords being assigned to multiple Groups in a given file, especially if such Groups had dissimilar privileges.

As an aside, in FileMaker Pro 8, Account Names and Privilege Set Names are case *insensitive* and must be *unique*. Account Passwords are case *sensitive* and do not have to be unique. See the information in the Tech Info Brief on the FileMaker, Inc. website entitled *Upgrading to FileMaker Pro 8: How to employ the new, advanced Security system* for a further discussion of these items.

About the author

Steven H. Blackwell is a Partner Member of the FileMaker Solutions Alliance and President and CEO of Management Counseling Services [<http://www.FMP-Power.com>]. A two–time winner of the FileMaker Excellence Award, he specializes in custom FileMaker Pro development, FileMaker Pro security consulting, and FileMaker Server deployment.



With additional research and technical review by Barbara R. Levine of MicroServ, LLC, Ernest Koe and Cornelius Walker or inResonance; Partner Members, FileMaker Solutions Alliance.

(Footnotes)

¹ Scenarios developed by Barbara R. Levine.

² See Part Four for more information on mal-formed schema characteristics.

³ Based on the internal ID of the Group.

⁴ Reference Tech Info Letter 102417.

⁵ Scenarios developed by Ernest Koe and Cornelius Walker.



“Record Ownership” in Converted Solutions: Opening and Committing Records

Multiple users can create, edit and delete records in a FileMaker® Pro database all at the same time. To prevent conflicts, the database environment must negotiate “record ownership” among its guests. We do not want two database guests revising data in the same record at the same time, or one person deleting a record that another person is editing.

In previous versions of FileMaker Pro, record ownership issues were handled in a very dependable and straightforward way requiring no intervention by the developer, but not offering much control to the developer either. In FileMaker Pro 8, there are some important differences in the actions and ScriptMaker™ commands that cause a record to be locked and “released” (committed to the server). On the one hand, much more power over record ownership is now in the hands of the developer. But, on the other hand, when you develop a new solution or convert a solution from a previous version of FileMaker Pro, it is important to have a clear understanding of how “opening and committing records” works. In particular with converted databases, it may be necessary to modify scripts and to rethink your design in order to ensure optimal handling of record ownership.

“Open” and “Commit”?

What does it mean to “open a record” and “commit a record”?

Each record in a FileMaker Pro database has an internal unique ID number (as you may have seen if you ever defined a “Status(CurrentRecordID)” calculation in previous versions). When certain manual or scripted actions are performed on a guest computer, FileMaker Pro sends a message to the server to say, “This record ID is now in use and is ‘owned’ by this guest.” The guest has now “opened” the record, and it is “locked” against editing by others.

If another database guest attempts to edit the same record, a message like this will appear:

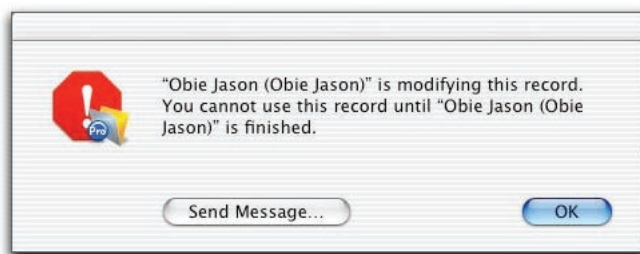


Figure 1: The “record is in use” alert.

When the first person is finished editing and exits the record, the changes they have made are sent back — “committed” — to the server, and the record once again becomes available for editing by other users.



If you have seen the above “record is in use” alert in previous versions of FileMaker Pro, your attention may have been drawn to the new “Send Message” button. If a manual action, or scripted action with Error Capture turned off, conflicts with another database guest who is already editing the record, you can now send a message to that other guest’s screen that looks like this:

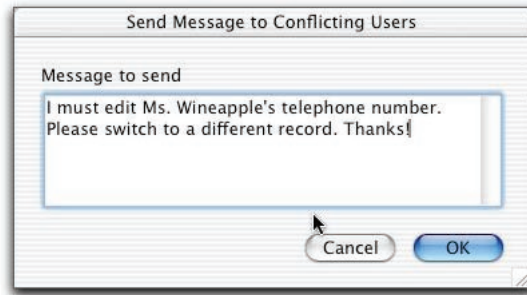


Figure 2: New feature — ask another user to release the record!

Opening a Record

When writing scripts, designing an interface, or even just using a database, a full understanding of exactly how records are opened will help you make optimal design decisions.

What Actions Open a Record?

Various manual and scripted actions will cause a record to be opened, and thereby locked against editing by others. The behaviors of some actions and certain ScriptMaker commands have changed significantly in FileMaker Pro 8. Here are some highlights of the changes:

A very basic difference, that applies across all possible manual and scripted actions that can be taken on a record, is that clicking or tabbing into a field (or scripting a “Go to field” command) no longer opens a record. In FileMaker Pro 8, only when the contents of a field are edited (in other words, typing or deletion takes place) is the record opened. On the one hand, this is beneficial in that a user can no longer lock a record against editing by others by merely entering a field. On the other hand, scripts that ensured record-ownership in earlier versions by means of using the “Go to field” command will not work dependably in FileMaker Pro 8. (See the “Changes You May Need to Make in Your Converted Database” section of this document, below, for the suggested fix).

Another change that will do away with record-locking problems seen in earlier versions is that a user will no longer open a record by clicking into a global field. Also, those of you who may have had to find work-arounds to contend with portal scroll-bars locking a record will be pleased to learn that this is no longer an issue.



In FileMaker Pro 8, the “New Record” command now not only leaves the record open (until a record-committing action is taken), but the new record cannot be seen on other workstations until it is committed.

Since it was introduced in FileMaker Pro 3, the “Set Field” command only momentarily locked the target record to make the edit, and then released (committed) the record again. In FileMaker Pro 8, however, *the “Set Field” command opens the record, and does not then automatically commit it.* A record on which a Set Field command has acted will remain open until it is committed by manual or scripted action.

See **Tables 1, 2 and 3** for a detailed analysis of record-locking behaviors and comparison to earlier versions of FileMaker Pro.

Who Can Open a Record?

As has always been the case, only database guests whose account privileges allow them to edit records can put a lock on them. However, there is a new “Run script with full access privileges” checkbox option in the script editor that, if checked, will temporarily enable a workstation to edit — and thereby place a temporary lock on — records via ScriptMaker commands.

! Before you check the new “Run script with full access privileges” box in the script editor, think through the record-locking ramifications. Does your record-ownership strategy depend upon certain users not being able to place a lock on a record? If so, use this script option with care.

The ability to have open records in more than one window may cause “Self-Locking”

There is a significant new record locking issue that could not exist in FileMaker 6, “self-locking.” A user can now edit a record in one window and that record will remain open when the user switches windows (either manually or via script). If an attempt is made to edit the same record from another window (either on a local layout or via a relationship from another table), the user can be locked out of the record in that window. This can occur during a script, with potentially serious consequences for data integrity given that edits may not occur as anticipated.

“New Window” Menu Item

The new FileMaker Pro 8 “New Window” command makes it easy for a user to initiate a self-locking situation. If a user has opened a record and then chooses the New Window command (or the “New Window” ScriptMaker command is run), the record in the original window remains locked. This is the alert that the user will see when this situation arises:



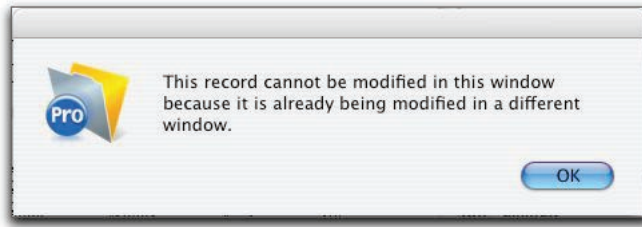


Figure 3: The “self-locking” record alert.

! If a database guest’s available menus allow them to use the “New Window” command, or if you decide to add this command to a script, keep the prospect of self-locking in mind. Self-locking can be caused without New Window as well, and can even put the workstation into an “endless loop” situation if a script attempts to edit a self-locked record anticipating that it would be available. Be sure to test carefully!

Self-Locking in Converted Scripts

In a converted solution, a script sequence that edited records in more than one file can now leave records locked when it leaves a window (formerly a file). For example, a script can encounter a self-lock initiated by a subscript called earlier in the same script. If Error Capture is on, it may be difficult to figure out what is happening.

The resolution to this sort of issue is to add a Commit Records script step before leaving a window. It may often be advisable to add the Commit Records step after a Set Field step or at the end of a series of Set Field steps.



Table 1: User Actions that Open a Record

(NOTE: When the information below notes that a record is “opened” this also means it is locked against editing by other users, and also is locked against editing by the same user in another window.)

SEE Table 2 FOR SPECIAL NOTES REGARDING PORTALS.

ACTION	EARLIER VERSIONS	FileMaker Pro 8
Click or tab into a local field	Opens the record.	Record is not opened.
Begin typing in a non-global field.	Record was already opened just by clicking in it.	Opens the record.
Click or tab (FileMaker Pro 6) or edit (FileMaker Pro 8) into a related field on the layout, when the current (local) record had not yet been opened.	Clicking or tabbing into the related field opens both the current record and the related record.	Editing in a related field opens both the current record and the related record.
Click, tab or type in a global field	Opens the record.	Record is not opened.
Select and Copy text	Requires record ownership; just entering the field opens the record.	Selection of text does not open the record; possible to Copy values from a record that is open on another computer.
“Replace/Relookup Field Contents” Menu Commands	These actions momentarily open each record as they move through the found set. Each record is immediately committed after it is edited, except for the record that is current when the action is complete: the cursor is left blinking in the replaced/relooked-up field and that record remains open.	Behavior is the same, except that the record that is current when the action is complete is also immediately committed. The cursor is left blinking in the field but the record is not locked against editing by others.
“New Record” and “Duplicate Record” Menu Commands 1. When the current layout has at least one enterable field in the Tab Order 2. When the current layout has no enterable fields in the Tab Order	1. A new record is opened on the workstation that created it, and it is immediately visible upon a screen refresh on other workstations. 2. The record is not opened; other users can immediately begin to edit it.	In either of these two situations: A new record is created and a record ID is obtained from the server, <i>but it remains open on the user’s workstation until an action or command commits the record (see Figure 4). Also, the record cannot be seen (nor, of course, edited) on other workstations.</i>
Change records (using a keyboard combination or the “book”)	If the cursor was in a field before the record-change, the record that is navigated to will be opened.	The cursor will remain in a field after the record-change, but the new record is not opened unless editing begins.

Table 2: Record-Opening Behaviors of the Portal

(NOTE: When the information below notes that a record is “opened” this also means it is locked against editing by other users, and also is locked against editing by the same user in another window.)

ACTION	EARLIER VERSIONS	FileMaker Pro 8
Click on a portal scrollbar	Opens the current record, prevents others from using the portal scrollbar, editing in the portal, or using portal-row buttons; also opens the first related record so that it cannot be edited by anyone working in the “child” file.	Neither the current record nor any child records are opened; portal scrollbar, fields and portal-row buttons remain accessible to others.
Click or tab into a portal-row field, or select the portal-row background	Opens the current record, prevents others from using the portal scrollbar, editing in the portal, or using portal-row buttons; also opens the related record whose field has been entered in the portal, so that it cannot be edited by anyone working in the “child” file.	Neither the current record nor any child records are opened; portal scrollbar, fields and portal-row buttons remain accessible to others.
Begin typing in a portal-row field	Record was already opened just by clicking in it.	Opens the current record and the child record whose field is being edited. Does not lock the portal scrollbar, but does prevent others from editing in any portal row (although they can edit any non-opened record “from the child side”).
Click a portal-row button whose action goes to the related record in a different window	In earlier versions, moving a window to the background always commits the current record in that window.	If the “parent” record was already opened before the portal-row button was clicked, <i>it will remain open even after its window is moved to the background.</i>



Table 3: Scripted Commands that Open a Record

(NOTE: When the information below notes that a record is “opened” this also means it is locked against editing by other users, and also is locked against editing by the same user in another window.)

As per the behaviors described in Table 1, if a related field is entered (earlier versions) or edited (FileMaker Pro 8), both the current record and the related record will also be opened, with the exception of the “Set Field” command (see below). All the “Portal Behaviors” described in Table 2 also apply for scripted editing actions.

ACTION	EARLIER VERSIONS	FileMaker Pro 8
Navigation Commands Go to field, Go to next field, Go to previous field; Go to portal row; or Go to Record when used from a layout that has at least one enterable field in the Tab Order.	All these commands will open the record.	<u>These commands no longer open a record.</u>
Editing Commands All commands in the “Editing” category	All these commands require record ownership and will open the record.	Behavior remains the same for commands that truly edit field contents. But Copy, Set Selection and Select All do not open the record nor require record ownership.
Field Commands Set Field <u>on a local field</u> Set Field <u>on a related field</u> All “Insert” commands Replace/Relookup Field Contents (formerly in the “Records” category)	If a record was already open, Set Field leaves it open, otherwise it momentarily opens the record and then immediately commits it again, by passing field validations. Opens (and commits, if records were not previously opened) both the current and the related record. Insert commands open the record(s) Scripted behaviors are the same as when invoked manually. See notes for FileMaker Pro 6 in Table 1 .	Set Field opens the record and <u>leaves it open</u> . Any subsequent action that commits the record will trigger validations unless explicitly bypassed with the option in the Commit Records step. Will not open the current record (if not previously opened); opens related record (and leaves it open). Behavior remains the same (includes new “Insert” commands). Scripted behaviors are the same as when invoked manually. See notes for FileMaker Pro 8 in Table 1 .
Records Commands New Record/Request and Duplicate Record/Request Open Record/Request Import Records	Scripted behaviors are the same as when invoked manually. See notes for FileMaker Pro 6 in Table 1 . New command in FileMaker Pro 8. If run with the “Replace” or “Update” option checked, opens each record as it moves through the found set, and then immediately release it	Scripted behaviors are the same as when invoked manually. See notes for FileMaker Pro 8 in Table 1 . Opens the record (even if the user is not editing a field) until an action or script command commits it. Same behavior as earlier versions.

When and How are Records Committed?

When a database is being accessed by multiple guests, we don't want records being left open longer than necessary, nor do we want a record to be committed to the server when subsequent actions depend on persistent record ownership. Some of the rules for committing records have changed, and these changes both offer some interesting new possibilities and present some potential pitfalls to keep in mind.

Perhaps the most dramatic change is that *all records edited in a portal will remain open until the local (parent) record is committed*. Because of this new approach to opening and committing records in FileMaker Pro 8, if the local record is reverted, all edited portal records will revert, too.

Particular attention should be paid in your converted solution to the fact that when a record in a background window is opened by a scripted action, or if a user opens a record and then manually brings a different window to the foreground, *the record in the background will remain open until it is explicitly committed by manual or scripted action*. This applies only to windows that belong to an external file — in FileMaker Pro 8 you cannot run a script in a hidden window if the displayed foreground window belongs to the same file. It is converted solutions that will most be affected by this, since after conversion they will remain in the previous one-file-per-table structure. Every “Set Field”, for example, that takes place in an external file will leave the record in that background file open.

See Table 4 (on the next page) for a detailed comparison of the record-committing behaviors of FileMaker Pro 8 vis-à-vis earlier versions.

Two New Features Give You More Control

While this information does not pertain to converted solutions, once you start enhancing your database with FileMaker Pro 8 two new features can offer increased flexibility when it is time to commit or revert a record. When you define a field as an auto-entered serial, you can now specify that the serial value will be generated “On creation” (as in earlier versions), or “On commit”. If you choose the latter option, when a new record is created the serialized field will remain empty so long as the record remains open.



Figure 4: Control when a serial number is assigned.

At the moment the record is committed, the server then assigns to it the next available serial value. Thus, it is now possible to delete an uncommitted new record without creating a gap in the serial numbers.

Another new feature solves an old dilemma: how to prevent changes to a record from being committed by merely clicking on the layout background or navigating out of the record? In the past, in order to provide full

“reverting” of changes to a record it was necessary to build an artificial “edit mode” involving separate layouts, global fields, lots of scripting and tightly controlled navigation. The developer’s job (and the user interface) was even more complicated if edits in portal rows might need to be reverted.

Now, simply by unchecking a new “Save changes automatically” option in the Layout Setup dialog, you can provide a way to back out of changes. If edits are made to data when working in that layout, before the record is committed a prompt offers the options to Save the changes, discard the changes (“Don’t Save”), or to Cancel committing the record altogether.

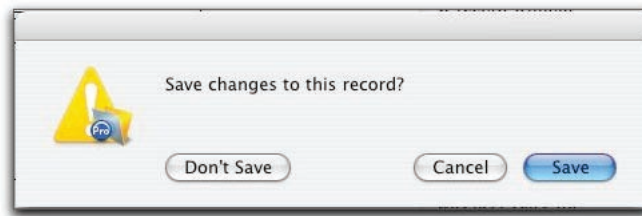


Figure 5:A new opportunity to “back out” of edits made to a record.

Because any edited portal row now remains open until the parent record is committed (see Table 4 for more detail), the “Save changes?” prompt makes it possible to completely revert the changes in the related records as well as in the local record.



Table 4: User Actions and Scripted Commands that Commit a Record* <small>*or that will trigger the “Save changes to this record?” prompt if the record is edited from a layout whose “Save changes automatically” box has been unchecked in the Layout Setup dialog</small>		
ACTION	EARLIER VERSIONS	FileMaker Pro 8
USER ACTIONS		
Click on the layout background, click on the “book” in the status area to change records, use the layout menu to change layouts, change modes, switch the “View as” option, or run a script whose action causes a change of records	Commits the record (and any opened related record).	Commits the record (and any related records that have been opened -- see next row of this chart).
Move cursor focus from one portal row to another	Commits the portal-row record that is exited and opens the newly entered portal-row record.	All portal-row records that are edited remain open until the parent record is explicitly committed.
Minimize the window	On Mac OS: the record remains open until a record-committing action is taken. On Windows OS: commits the record.	On Mac OS: Behavior unchanged, the record remains open. On Windows OS: The Minimize Window menu command and Adjust Window (Minimize) script step will leave the record open; but clicking on the window “widget” to minimize the window will cause the record to be committed, as will using the menu in the upper left corner of the window.
Hide the window	Commits the record.	Behavior remains the same: hiding the window commits the record, however a record can be locked by a subscript in a hidden window!
Running a script in a background window	If focus remains on the foreground window when script is complete, records in background window are committed.	If record-opening action is taken on records in a background window belonging to an external file, those records remain open until explicitly committed.
Switch to another window either by script (Go to Related Record, Perform Script(External), Open File) or by clicking on a background window or using the Window menu	Commits the record. (Note that in FileMaker Pro 6, in the interface, a window is synonymous with a file)	The record remains open in the window (file) that is now in the background, until it is explicitly committed.



<i>Close the window</i>	Commits the record.	Behavior remains the same: the record is committed.
SCRIPT COMMANDS — Commands that automate the above actions follow the same rules as noted when the action is performed manually by the user. There is one important change:		
<i>Exit Record/Request</i>	Commits the record.	Renamed “Commit Records/Requests.” Commits any open related records as well as the current record. Includes new option: “Skip data entry validation”

Changes You May Need to Make in Your Converted Database

Do not rely on entering a “global field” to lock a record

FileMaker Pro 8 does not put a lock on a record when a database guest clicks into or edits a global field. So if your solution depends in any way upon locking a record, for dependable performance after conversion be sure you do not rely on entering or editing a global field to gain record ownership. Modify your script so that it edits a non-global field, or simply uses the “Open Record” command, in order to put a lock on the record against editing by others.

Use an “Insert” command rather than “Go to field” to test for locked records and/or to obtain record ownership

A commonly used technique for ensuring that a guest machine obtains ownership of a record, and to prevent other guests from editing it, has been to use the “Go to field” command. In previous versions of FileMaker Pro, this command, if it succeeded without error, would open the record; or it would generate the error code 301, “Record is in use,” if the record had already been opened on another computer.

The “Go to field” command no longer opens the record in FileMaker Pro 8. In order to test for locked records or to obtain ownership of unlocked records, you will need to revise any script that uses “Go to Field” for this purpose. Modify the script to use an “Insert” or “Set Field” command instead. An advantage of using an Insert command (such as Insert Calculated Result) is that it will achieve the record lock in both FileMaker Pro 6 and FileMaker Pro 8, so the change can be made in a working solution in FileMaker Pro 6, prior to conversion.

You may need to add a “Commit Records” command after using “Set Field”

A record that has been edited by the Set Field command is no longer automatically committed. Examine your scripts for all instances where the Set Field command takes action on a non-global field, and consider whether you might need to add a Commit Record command to make the change visible to other guests, or to simply release the record lock.



If a script uses “Set Field” to edit a value in a related table, the related record will remain open until the parent record is committed. If the related record happens to be in an external file (very likely to be the case in a converted solution), calling an external “Commit Records” command in the related file will not commit its open record. The “Commit Records” command (or action) must be run directly from the window from which edits were made in order to commit all records that were opened.

You may need to add a “Commit Records” command before navigating to a different file

Remember that switching windows in FileMaker Pro 8 does not serve to commit an open record in the window you are putting in the background. Examine all your scripted navigation with this in mind, and add the Commit Records command anywhere you want to ensure a record is committed before a different window is brought to the front.

Prevent the New Window menu item from being available to users

To prevent self-locking which could compromise scripted as well as manual data entry, it is advisable to disable the “New Window” menu item. This can be accomplished either by setting the Available menu commands to “Editing only” or “Minimal” for all Privilege Sets.

Use SecureFM to disable the New Window and the Show Window menu items

To protect against self-locking with a multi-file solution, you can selectively prevent user access to both the New Window and the Show Window menu items using the SecureFM plug-in from New Millennium Communications (www.nmci.com).

Test, Test Again, and Test Some More!

The information and suggestions in this article are based on a good deal of testing, but should be considered “early research”. Whether you will be deploying a converted database or building fresh in FileMaker Pro 8, test the database thoroughly. Only after much more testing by the community of FileMaker developers and users, in a variety of environments and on a variety of database solutions, can we collectively evolve complete information and a set of recommended “best practices.”

About the author

Ilyse Kazar, CEO of Datatude Ltd. in New York, is a leading FileMaker Pro database consultant and developer. Since 1995, she has engineered and managed numerous successful custom development projects for clients in a broad variety of industries. Ilyse has a special interest in the design issues involved in creating multi-user solutions.



Migration and Web Publishing

The web publishing capabilities of FileMaker® 8 are among the most compelling features of the product. Instant Web Publishing (IWP) has been greatly expanded, yet has retained its unparalleled ease of use. Custom Web Publishing (CWP), which now consists exclusively of XML/XSLT, is tied directly to FileMaker Server 8 Advanced (FMSA), thereby offering much greater stability and performance than were previously possible.

When migrating existing solutions to FileMaker Pro 8, there are special issues that must be considered if those solutions are web-enabled. Those issues are largely dependent on the particular tool that was used to web-enable the solution. Before converting any web-enabled solution, it is important for you to be familiar with both the new web-publishing infrastructure and the specific programming changes made to both IWP and CWP. A thorough discussion of either of these topics is well beyond the scope of this paper. Here, I will simply present an overview of the new architecture and the conversion issues facing users of various tools.

Overview of the new web publishing architecture

If you are considering migrating existing web-enabled solutions to FileMaker 8, it is important that you understand the benefits offered by the new web architecture. Virtually all of the web tools from previous versions of FileMaker Pro have been rendered obsolete. In FileMaker Pro 8, there is no Web Companion, no Web Security Database, no FileMaker Unlimited, and no Web Server Connector.

There are, however, still the same two basic methods of web-enabling FileMaker Pro data, IWP and CWP; both tools have become much more sophisticated and powerful. One of the most significant architectural changes is that FileMaker Server 8 Advanced can act as the host for both IWP and CWP solutions. FileMaker Pro 8 can itself still host IWP solutions as well (up to 10 files for up to 5 users), but server-based hosting will provide the most power and stability.

Instant Web Publishing is a quick and easy way of extending your FileMaker Pro 8 solution to a set of remote users. IWP renders your FileMaker Pro 8 layouts as web pages, and web-compatible scripts will function the same as they do for FileMaker Pro 8 users. There are nonetheless some important differences between the IWP experience and the FileMaker Pro 8 experience, including the following:

- There is no Preview mode in IWP. Functions that require Preview mode, such as sliding, columnar reports, and subsummary reports, are therefore not possible via IWP.
- IWP has no tools for editing the database schema; you can only work with data through the tools constructed in FileMaker Pro 8.
- On the web, there is a distinction between Edit mode and Browse mode that does not exist in FileMaker Pro 8. Some FileMaker Pro 8 routines, such as filtered portals and auto-entering looked up values, do not translate well to the web.



- In IWP, list views always contain a maximum of 25 records and table views always contain 50 records. The IWP status area has next and previous controls for paging through data that do not have analogs in FileMaker Pro 8.
- Only a subset of the available script steps is web compatible. You can see which ones are by checking the “Indicate web compatibility” checkbox when viewing scripts; more than 70 scripts steps are web compatible. Scripts that require user interaction (such as Show Custom Dialog) or that interact with the operating system (such as printing, importing, and exporting) are generally not supported by IWP.

Despite these restrictions, IWP is ideally suited for extending a FileMaker Pro 8 solution to a set of known remote users. There are some browser restrictions and connection limits, so it may not be appropriate for most publicly accessible web sites.

The centerpiece of the new server-based web architecture is the FileMaker Server 8 Advanced Web Publishing Engine (WPE). The WPE is configured to work with both a web server running either Apache on Mac OS X or Microsoft’s Internet Information Server (IIS) and a copy of FileMaker Server 8 Advanced. It hands off web requests to FileMaker Server 8, then conveys the results back to the web server. These three pieces (WPE, FMSA, Apache/IIS) can be deployed in a one-, two-, or three-machine configuration. Other pieces of the web infrastructure include the Web Server Module, which is a plug-in for either Apache or IIS, and the FileMaker Server Administration Console, a web-based application for enabling and configuring WPE functions.

Some of the particular benefits of the new web publishing tools in FileMaker Pro 8 include the following:

- **Unified Security**

Regardless of what method of web publishing you use, all security is controlled by the same accounts and passwords that govern all other access to a file. You can grant users web access to a file by adding one or more particular extended privileges to their privilege set. For IWP access, enable the fmiwp extended privilege; for Custom Web Publishing, you need to create two new extended privileges, fmxml and fmxmlt, and attach these to one or more privilege sets.

Whatever actions are permitted or denied by a user’s privilege set will apply to their web interactions. You can prevent web clients from adding, modifying, viewing, or deleting records just the same as you would for FileMaker Pro 8 clients.

- **Session-based**

When a web user connects to a database, they initiate a virtual session on the host. That session persists until the web user logs out or until a timeout threshold has been exceeded. The virtual session “knows” such thing as the values that a web user has assigned to global fields, what windows, found sets, and records are available and active, and what record a web user is modifying. Web users will be notified if a record has been locked by another user (and therefore cannot be edited). Similarly, when a web user is modifying a record, the virtual session ensures that other users on the network will not be able to edit that record until the record has been committed. In sum, the fact that the new web architecture is session based makes for a much more FileMaker Pro 8-like experience for users.



- **Greater stability and performance**

One of the limitations of the previous generation of FileMaker Pro web tools was that all web traffic flowed through the Web Companion, which is single-threaded application. This led to the possibility of performance and stability issues. For instance, if one web user initiated a sort of a large set of records, other requests from web users would be queued up and would not start until the sort was complete. Additionally, if any action caused a modal dialog box to appear on the Web Companion machine, until the dialog box was cleared, the Web Companion could not respond to queries. The new server-based hosting is much more stable and powerful. In and of itself, this is a compelling reason to migrate existing solutions to FileMaker Pro 8 and FileMaker Server 8 Advanced.

Current IWP solutions

None of the Instant Web Publishing setup options from previous versions are retained during conversion to FileMaker Pro 8. Indeed, this is because there is no need for them in the new version. The themes and views that comprised the setup options in previous versions are now obsolete. The new IWP status area, previously specified as a theme, now resembles in both look and functionality the status area of FileMaker Pro 8. Its layout pop-up menu allows users to navigate to any layout permitted by their privilege set, obviating the need for “views”.

Rebuilding your existing IWP capabilities following migration to FileMaker Pro 8 will take only a small effort. You will first need to configure the host (either FileMaker Pro 8 or FileMaker Server 8 Advanced) to allow IWP access. Then, simply assign the fmiwp extended privilege to the privilege set of users for whom you want to grant web access. If you want to restrict users to the layouts that you had previously designated for IWP access, modify the privilege sets used for IWP to only allow access to those layouts.

Given the much enhanced script support of the new IWP, you will find that most FileMaker Pro 8 functionality translates well to the web; you will be able to accomplish much more with IWP than you ever could previously.

Current CDML solutions

As mentioned above, FileMaker Pro 8 does not support CDML solutions. If you have an existing CDML solution that you want to migrate to FileMaker 8, you can use the CDML to XSLT conversion tool (which comes with FileMaker Server 8 Advanced) to convert your format files into XSLT style sheets.

You should learn the basics of XML and XSLT before deploying a converted CDML solution. While the conversion tool attempts to retain as much functionality as possible from your format files, you should thoroughly test the converted files and be prepared to modify them if necessary.

Current third-party solutions

There are several other popular methods for web-enabling FileMaker Pro databases, such as Blue World's Lasso product line and PHP. Check Blue World's website (www.blueworld.com) for information on using Lasso with FileMaker Pro 8. For PHP solutions using the popular FX.php class, see the FX web site at www.iviking.org.



About the author

Bob Bowers, president of the Moyer Group, is the co-author of three books on FileMaker Pro, including Special Edition Using FileMaker Pro 8. In addition, he's a columnist and contributing editor for FileMaker Advisor magazine, and is one of only a handful of trainers authorized to teach the FileMaker Professional Foundation Training Series.



Methodologies

Conversion Basics

Convert and Restore

This document discusses the process of converting a FileMaker® .fp3 or .fp5 solution and making the modifications necessary to deploy it successfully as a FileMaker Pro 8 solution. This methodology will get you up and running reliably in FileMaker Pro 8, without enhancements, and with a solution architecture which is identical to your original solution. Read subsequent sections of this document for strategies for extending the features of your solution, or redesigning your solution entirely, to take advantage of the new capabilities of FileMaker Pro 8.

The alternative to following this process is to rewrite your solution entirely. There are at least three possible motivations for taking the “Convert and Restore” path.

1. You may want to realize some of the immediate benefits of FileMaker Pro 8 without investing in the learning time or the development time required to redesign your solution¹.
2. You may be deploying a new FileMaker Pro 8 solution, or a part of an existing FileMaker Pro 8, yet there is another part of your solution, or an associated set of files, which exist in FileMaker Pro 6 and which need to be integrated with the new files. Since FileMaker .fp5 and .fp7 files cannot be directly related to each other, it may be desirable to convert the associated files so you can integrate them with the new FileMaker Pro 8 solution. You may not have the time or desire to rewrite all of a solution, and therefore one part of the solution will need to be upgraded using the Convert and Restore method.
3. Much has changed in FileMaker Pro 8 from previous versions. There is no better way to learn the nuances of the changes than working to identify the instances of the changed behavior and learning how to integrate the changed functionality. You will become intimately familiar with the new behavior and will master it, with benefits to your expertise for future projects in FileMaker Pro 8.

There are different scenarios that affect the exact methodology you follow. You may be freezing all further development in FileMaker Pro 6, and converting your solution to FileMaker Pro 8 in a short period of time. Alternately, you may have a large solution that you plan to migrate, and you may want to begin preparing that solution for future conversion while continuing to enhance it in FileMaker Pro 6. In the latter scenario, there are a number of things that you may need to modify in your FileMaker Pro 6 solution to enable it to convert efficiently and successfully to FileMaker Pro 8, while continuing to work reliably in FileMaker Pro 6.

Regardless of your scenario (except for very simple solutions), it is wise to anticipate converting the solution numerous times. It is most efficient to do pre-conversion analysis, dry-run conversion, and post-conversion



testing. Then go back to the pre-converted solution, re-analyze, make more modifications, and reconvert. Repeat this process until the efficiency of pre-conversion tasks has been exhausted, and then do a final conversion and make necessary changes that can only be made in FileMaker 8.

Plan well, and follow the plan. Be methodical. Expect to change course or discover things during the process that you did not anticipate. Do not begin to add features to your converted solution before finishing your modifications to existing features, including testing and confirming that they are working correctly. You do not need to be trying to distinguish whether you are dealing with a conversion problem or a new bug that you have introduced by your enhancements. Think like an engineer – isolate and solve one issue at a time.

The Most Common Issues

The issues that manifest most frequently as problems in converted solutions are in the following areas:

- a. Conversion of access privileges
 - i. Case sensitivity of passwords
 - ii. Status(CurrentGroups) - changes to group names
- b. Scripting and calculation changes
 - i. Navigation scripts
 - ii. On Open and On Close scripts
 - iii. Several specific calculation result changes
- c. Problems with file references
 - i. Performance issues
 - ii. Files can't be found
- d. Managing data integrity
 - i. Changes in when and how records are locked and committed
 - ii. Changes to New Record, Set Field, Validations
 - iii. Consequences of multiple windows – scripts can leave records locked in more than one window at a time, so scripts can fail due to record locking even in a single user environment
 - iv. Changes in field indexing, especially word separators
- e. Plug-in compatibility and interaction with third party products

See the in-depth documentation of these issues in other areas of this larger document, in the “FM7 Converting Databases” .pdf document, and in the “Conversion Issues and Resolutions” appendix to this larger document.

Methodology Overview

The process will look like this (with the pre-conversion stage being considerably longer than the post-conversion stage):



Preparation

Getting educated

Getting experience

Required tools

Pre-conversion

Analysis

Pre-conversion modifications

Dry-run conversion

Testing

(Repeat)

Post-conversion

Post-conversion modifications

Testing

(Repeat)

Deploy (over the course of a few days, weeks, or months – as applicable)

Use Checklists!

There is a very helpful checklist that is part of the “FM 7 Converting Databases” .pdf that covers many conversion issues. There is also a checklist of the issues that are included in the “Conversion Issues and Resolutions” appendix of this larger document and available in database form in the MetadataMagic™ “Conversion Issues Report”. Create your own checklists as you become familiar with the issues and the tasks required.

There are a few different classes of checklists that are required:

- Checklists of known conversion issues to make sure that all potential problems have been considered and attended to if necessary.
- Checklists of processes in your solution which you intend to verify for accuracy and reliable behavior post-conversion.
- Checklists derived from the metadata of your solution of all instances of a particular issue in your solution, which you can use to methodically remedy the issue either pre-conversion or post-conversion.

Getting Educated

- I. Read the “FM 7 Converting Databases” .pdf (included with FileMaker Pro 8 and FileMaker Pro 8 Advanced). See if a more recent version of this document is available from the FileMaker web site.



2. Read the Tech Brief on Migrating Existing Solutions to FileMaker 8.
3. Read other Tech Briefs on Upgrading to FileMaker 8 –Security, Server, and Web Publishing.
4. Read this larger document – all of it!
5. Third party books and trainings – There are (or will be) many books and other white papers about FileMaker Pro 8. There will also be excellent training programs available from several sources.

Getting Experience

6. Learn FileMaker Pro 8. Working on a converted solution is a very difficult way to learn the new application. It is ideal to work on a new solution, even a moderately simple one, prior to converting an existing solution. FileMaker Pro 8 is more different from prior versions of FileMaker Pro than it may first appear to be.
7. Collaborate with others who are familiar with FileMaker Pro 8. Participate in online discussion lists. Ask questions. Do not make too many assumptions.

Tools

8. The only required tool is FileMaker Pro 8. However, it is strongly recommended that you use FileMaker Pro 8 Advanced so that you can use the Script Debugger to find problems, as well as being able to use the Database Design Report for analysis.
9. MetadataMagic™ – available from New Millennium Communications – includes File Reference Fixer, which is an essential pre-conversion utility, as well as a “Conversion Issues Report” which will help to identify potential problems in your solution.
10. Conversion Log Analysis Tool™ – available from New Millennium Communications – this utility will import the Conversion.log file that is created when FileMaker Pro 8 converts a solution, and will enable you to efficiently analyze it to identify problems.
11. FMrobot™ – available from New Millennium Communications – this application will allow you to automate the creation of tables and fields in a FileMaker Pro 8 file, based on field definitions in an existing FileMaker Pro .fp5 or FileMaker Pro .fp7 file.
12. Please visit the FileMaker web site (www.filemaker.com) periodically for information about new third-party tools that may be released to assist you in your conversion from previous versions of FileMaker Pro to FileMaker Pro 8.

* **Please Note:** FileMaker Pro 8 and FileMaker Pro 8 Advanced now provide the ability to import tables. FileMaker Pro 8 Advanced also includes the ability to copy/paste scripts, fields, and tables.



Preparing for Iterative Conversion Testing (in your FileMaker Pro 6 solution)

13. If you have not already, create a script in each file, and a central calling script which will create clones of all of your files so you can test without having to convert data. You may need to add at least one record to each file to make sure that it works correctly. This can also be done via a script in each file that is called from a central file. If you have any files that contain required data (reference tables), then make copies rather than clones of those files.
14. Put branches (If, Else), based on “Status(CurrentAppVersion)”, into all appropriate scripts of a solution. The branches for FileMaker Pro 8 may bypass certain steps in a script, or simply call an “Exit Script” step, or they may call a subscript that will perform certain script steps only in FileMaker Pro 8. The If statement calc to use is: “TextToNum(Status(CurrentAppVersion)) < 7” (or <8).
15. Put placeholder subscripts into all appropriate locations in a solution. These placeholders will later contain steps that can only be added in FileMaker Pro 8 (because they do not exist in FileMaker Pro 6). By using a placeholder subscript, it will be possible to add the step or steps quickly after conversion and they will be immediately functional in all desired locations. In some instances you can put steps into scripts which are innocuous in FileMaker Pro 6 files but which will enable a script to convert and behave perfectly in FileMaker Pro 8.
16. On Open and On Close scripts have changed their behavior subtly but with potentially significant consequences. Without remediation you can even be prevented from opening your converted solution at all, and you can get stuck in endless loops when closing files. Put a branch into your On Open and On Close scripts so they can be turned on or off by referencing a developer toggle (a flag field which you can set to make the action conditional). The developer toggle can be a global field in a central file, referenced through a constant relationship or a non-matching relationship. Before conversion, put a conditional step at the top of these scripts so that they can be exited without executing the remaining steps. You may want to make the conditional test branch on the version of FileMaker Pro that you are running. Alternately, you may want to turn off these scripts altogether in Document Preferences (in FileMaker Pro 6) or File Options (in FileMaker Pro 8). Read the section of the larger document on “Scripting Issues” for more information with the issue.
17. Any dependencies on plug-ins may need to be disabled until those plug-ins are available for FileMaker Pro 8.
18. Turn every instance of Set Error Capture [On] into a sub-script so that you can control it with a developer toggle. This enables you to see certain errors as they occur in your scripts because they will not be suppressed.
19. Add a “Commit Records” script, with one script step (Exit Record), to every file. After conversion, check the checkbox “skip data entry validations”. You may need to call this script after New Record steps or Set Field steps in your solution to restore legacy behavior. You may also want to put this script into the Scripts menu, so that if you get hung up when a script is executing you can get past it. This recommendation addresses issues associated with the changes to records locking and record commitment. These changes have far-reaching consequences.



20. While you are at it, you may want to add a “Halt Script” script (with just one step) and put it in the menu of every file. Again, you can make this feature conditional on a developer flag.
21. After conversion, the behavior of navigation between files can change, depending on the exact script step sequence. By adding steps in advance of conversion after certain instances of Perform Script [External] or Go to Related Record, it is possible to cause the script behavior to convert reliably.
22. Standardize your passwords so that a given password in all files is the same case (lowercase, uppercase, mixed case). You can do this manually or with the “Password Standardizer” feature of Password Administrator. This will prevent passwords from failing to open files when they would have in FileMaker Pro 6. See the section of this larger document on conversion of Access Privileges for more information.

Analysis – Using MetadataMagic

23. Run MetadataMagic (available from New Millennium Communications) on your solution. If there are any processing errors, see which items are causing the problem. They may represent some corruption that could cause a problem in the conversion to FileMaker Pro 8. Fix the problem, usually by replacing the object, and then re-process your solution until it processes cleanly.
24. Fix file references before conversion by using the “File Reference Fixer” feature of MetadataMagic. It can be a quick automated pre-conversion process and a very laborious manual process if performed after conversion – with significant consequences for solution performance, reliability, extensibility, and ease of ongoing development. Make sure you include all files that are part of the solution (or which are ever referenced by the solution) in the folder when processing. If your solution is always run with all files in the same folder or on the same server, then you can use the Auto-Fix feature. If you sometimes reference files that are in a different folder or on a different server, then carefully read the instructions regarding “Special Situations” in the File Reference Fixer documentation. Among other consequences, problems associated with file references include very slow opening of converted solutions, or the potential of the application “not responding” during opening of files.
25. Use this opportunity to clean up obsolete scripts, fields, and layouts in your solution. You can use the “unreferenced items” flags in MetadataMagic to find all such items. It is also a good time to fix any errors. There is a dedicated “Errors” report file in MetadataMagic.

Iterative Conversion

26. In rare cases, you may have some failures in the conversion. That is, FileMaker Pro 8 may not be able to successfully convert a file. Sometimes this is caused by corrupt layout objects, which you may be able to find with MetadataMagic. Otherwise they can be found by a process of elimination, by deleting layouts selectively and reconverting. Another known cause of files occasionally failing to convert successfully is that there may be illegal characters in field names or relationship names. Sometimes these characters



are visible on one platform and not on another (Macintosh/Windows). Try clearing the old name, and re-creating it. If necessary delete the object and replace it. Sometimes a layout object cannot be accessed without causing a crash, which effectively prevents the object from being deleted. In this scenario, try accessing it on a different platform. If that does not work, the only alternative is to delete the entire layout and recreate it (and reconnect all references to it). To delete a layout with this problem, go to an adjacent layout in layout mode, scroll way down so that the layout contents are out of sight, then switch to the layout that has the corruption problem and delete it. By not displaying the problem element, the crash is prevented.

27. Analyze the Conversion Log, using the New Millennium “Conversion Log Analysis Tool”. You can scan it visually, but it is very long and filled with non-essential information, so it is easy to miss the few important bits of information. Look especially for items identified as “damaged”, “errors”, or “invalid”. Try to alter or re-create these items in FileMaker Pro 6, and then re-convert. In rare cases, FileMaker Pro 8 may not be able to identify the name of a damaged item, so that the log will simply say that an “unknown” item was not converted. For instance, a damaged field may not be brought over. In this case you will need to carefully compare the names of the fields from the pre-converted file with the post-converted file.
28. Review the “FM7 Converting Databases” .pdf document and identify the conversion issues that are relevant to your solution. It is also essential to become familiar with the appendix of this document, entitled “Conversion Issues and Resolutions”.
29. The “Conversion Issues Report” in MetadataMagic may be helpful for finding all instances of a specific problem in your files.
30. Fix all instances of issues that are most efficiently fixed pre-conversion, and then convert and test. Repeat as necessary. The “Conversion Issues and Resolutions” items include a distinction of which items are most efficiently fixed pre-conversion.
31. Fix data that will be affected by conversion. There are changes to how certain characters are evaluated in certain field types (for instance with date and time separators, and with text in number fields). There are also significant changes to indexing – including to values that are used in keys, or from which keys could be derived. For instance, dashes have changed behavior as a word separator. This could have significant consequences for data integrity.
32. Evaluate your layouts after conversion. You may want to do some cosmetic tweaking between conversion attempts -- replace fonts, replace/resize images, change hairlines to 1 pixel, etc. If there is an issue with certain fonts or graphics or other layout items, isolate the optimal way to prep layout elements for conversion by using a dedicated file with no scripts or fields, so that you can test iterative conversion efficiently. Some graphics may render differently depending on the platform on which the files are converted. If you are having a problem, try converting on the other platform. Save the converted files from both platforms. You may later come across a layout element that you want from “the other” converted set.
33. It is important, after conversion, to test script behavior and calculation results for reliability and consistency with prior behavior. Note, however, that comprehensive testing is very difficult. You may not



think to test features in a certain sequence that could cause a problematic result. The only substantive remedy is to have a good familiarity with the possible issues so that you know what to test, and/or know what to fix in the first place.

See the Conversion Issues documentation in this larger document for the specific issues and for advice on which ones are best addressed pre-conversion and which post-conversion.

34. Include users in the testing process. Include the people who are most familiar with the use of the system in testing and evaluating that the features are working reliably. They will inevitably catch things that a developer will miss.

Post-Conversion Tasks – After the Final Iterative Conversion Testing

35. Change account names. FileMaker Pro 6 passwords become both the account name and the password in FileMaker Pro 8. Furthermore, consider changing your security structure more extensively post-conversion. The access privilege architecture is very different, and much more powerful, in FileMaker Pro 8. In FileMaker Pro 6 solutions, it was commonplace for multiple users to share passwords. Due to the enhanced access privilege management features of FileMaker Pro 8, it is practical for each user to have their own account, and it is strongly recommended that developers implement their security structure accordingly.
36. Populate the placeholder scripts, described above, implementing any features which can only be added in FileMaker Pro 8. For example, check the checkbox in the Commit Records script step, “Skip Data Entry Validations”. It is possible that you will have already done this in each round of iterative conversion and testing.
37. Organize the relationships graph. After conversion, all relationships in FileMaker Pro 6 become table occurrences on the relationships graph in FileMaker Pro 8. They are arranged neatly, grouped by common primary keys. However, they are arranged in two columns, potentially very long columns if there are many relationships. It is hard to find particular relationships and manage them without organizing them better. Consider using the tools at your disposal to manage the relationships graph – the primary tools being space, color, and naming conventions.

Importing Data

38. It is likely that in any complex solution it will be most efficient to practice conversion on cloned files. Therefore, it will be necessary to import data post-conversion. The export/import process works similarly to the way it did in previous versions, with just a couple of exceptions. Read the documentation on conversion issues associated with import and export for more information in both the “FM 7 Converting Databases” .pdf and in the “Conversion Issues and Resolutions” appendix to this larger document. Notably,



importing from closed files will now import all records from a given table, so, if you want to import just a found set, the source file must be open.

39. There are multiple approaches to consider for transferring data from previous files to new files. For some scenarios it will be appropriate to use an interim file as part of the process. You can export your data to a tab-delimited text file, a merge file (.mer format) which allows you to import using “matching names”, an XML file which you will be able to transform with an XSLT to change field names or restructure the data, or you can export in FileMaker Pro format which has the benefit of preserving container field contents. There is no need to export calculation or summary fields (unless you intend to import those values into non-calculation fields).
40. For certain situations, especially commercial solutions for which the conversion is intended to be automated, it may be useful to pull the data rather than push it. You can create an interim FileMaker Pro .FP5 file that is designed to import the data from the previous version files, and then that is the file to convert. Among the benefits of this approach is that the interim files have no “on open” scripts to deal with (in fact, no scripts at all), no file references, nor any indexes, so the process can be much faster and simpler.

Managing Your Initial Deployment Environment

41. Try to replicate the previous FileMaker Pro environment as much as possible. Use either “Editing Only” menus, or SecureFM (from New Millennium Communications), to disable the New Window menu item. Use SecureFM to disable the Show Window menu item.
42. DO NOT add new layouts to your converted file that use any of the external table occurrences generated by the conversion. Your scripts were written without the need to manage their context. For instance, if you add a layout to your “Invoices” file, using the table occurrence that was created for your relationship to “Contacts”, and then run your “New Record” script, it will create a new contact, not a new invoice. Similarly, but more seriously, delete record will delete records from the table of the current layout.
43. Hardware and OS issues – FileMaker Server 8 requires a much more substantial machine than did FileMaker Server 5.5 and earlier. FileMaker Server 8 is supported only on Windows 2000 Server, Windows 2003 Server, and Macintosh OS X Server. The OS requirements of FileMaker Pro 8 have changed.
44. As with any technology deployment, it is important to isolate components of the system and to make sure that they are working reliably before aggregating them.



About the author

Danny Mack is the President of New Millennium Communications, Inc., a FileMaker Solutions Alliance Partner based in Boulder, Colorado. New Millennium specializes in FileMaker Pro consulting and solution development, and is the publisher of numerous plug-ins and tools that facilitate the work of FileMaker Pro developers, available at <http://www.newmillennium.com>.

(Footnote)

¹ These instant benefits include: no file size limit (for all practical purposes); no file count limit; container fields that can store any type of file, text fields that can hold up to 2 GB of data; calculations don't lose text formatting; WAN performance improvement; Server-based encryption of network traffic; Unicode support in text fields; robust, nearly incorruptible, file format; greater security of the file format; precise math; sticky portals; and many new features for going forward – including the new relational model; access privileges; Instant Web Publishing; and an extraordinary number of large and small efficiency improvements.



Adding a New Interface File To An Existing Solution, Later Consolidating Tables

FileMaker Pro users who want to begin leveraging the powerful new features of FileMaker Pro 8 but have a significant investment in a FileMaker Pro 6 solution may face a difficult dilemma, especially if the FileMaker Pro 6 solution is in use every day providing value for their organization.

They may be tempted to rewrite the solution from scratch. Although this may be the appropriate solution for some scenarios, it does have some significant drawbacks. Among other problems, the time and resources required may be significant.

Alternatively, they may decide that they need to get some return on investment (ROI) as early as possible and therefore it makes sense to convert and restore the solution to its original FileMaker Pro 6 behavior. The problem with this approach is that at the end you have a solution that is FileMaker Pro 8 *compatible* but not FileMaker Pro 8 *optimized*. This may make it difficult to extend the solution in the future.

What is needed is the ability to bring a solution from the “converted and restored” stage (i.e., FileMaker Pro 8 compatible) to a fully optimized for FileMaker Pro 8 stage that is better suited for continuing development and extension. This would allow both a quick return on investment and the ability to continue to use the system as a reliable base for future development. Fortunately the new flexible application model of FileMaker Pro 8 makes it possible to do just that.

This section of the document discusses a methodology that developers can use to move their “converted and restored” FileMaker Pro 8 solutions along a path of multiple stable points to a stage that is suitable for ongoing development, i.e., FileMaker Pro 8 optimized. Important features are pointed out and a step-by-step guide for the methodology is outlined.

Why rewriting doesn't always makes sense

The decision to rewrite should not be made lightly. Costs and benefits have to be weighed very carefully. The benefits of moving to FileMaker Pro 8 are quite clear but the cost of rewriting may not be. Among the costs that need to be considered are: time and resources required to rebuild, effect on current operations, potential for serious design flaws due to lack of experience, and length of time to begin to realize a return on the investment.

End users will be left using the current solution until the new one is ready to go into production. Depending on the complexity of the solution this could take many months. During this period, development efforts will have to be split between building the new solution and maintaining the existing one.

Converting the solution to FileMaker Pro 8 and restoring the original functionality may be the fastest way to get some ROI. There are some significant benefits to be realized from simply running in FileMaker Pro 8. (See the tech brief on “*Upgrading to FileMaker 8: Migrating Existing Solutions*” for more specifics.)



Why Convert and Restore isn't good enough

The Convert and Restore option does have some drawbacks, however. Although the solution is now FileMaker Pro 8 compatible, it can hardly be said to be using an optimal design for FileMaker Pro 8. It has more files than necessary and is probably full of techniques that are now obsolete in FileMaker Pro 8. Although this may not affect the usability of the system, it probably does affect the maintenance and extension of the system.

The great strength of FileMaker Pro is its flexibility. Developers can quickly add new features and make changes to live systems almost at will. Even if they are not working in the live files, there are few applications, if any, that can match FileMaker Pro as a Rapid Application Development (RAD) tool. A converted and restored solution may compromise some of these benefits, if it is left in a subtly fragile state. The solution will work, but unless the developer is careful it may be difficult to extend.

One of the most important things to understand about the state of a converted and restored solution is that it was designed for an environment that was constrained to one table and one window per file. Removing those constraints may cause problems with “context”. A reasonable approach might be to add those constraints back to the system.

The “New Window” feature can cause issues for scripts that were not designed to handle multiple windows with their different found sets and potentially multiple locked records. In order to ensure that a solution works as it did in FileMaker Pro 6, the “New Window” menu item should be disabled. This can be done either with the FileMaker Pro 8 access privileges by defining all privilege sets’ menus as “Editing Only” or “Minimal”.

Converted and restored solutions do not have problems associated with having many tables per file because they only have one table in every file. The constraint of *one file per table* is still in place as long as the developer does not add new layouts based on other tables.

When you add another table to the file, table context is no longer guaranteed, and script logic can fail. In fact, it is a bit subtler than it at first appears. Files will already have “Table Occurrences” from other files in them. When a .fp3 or .fp5 file is converted to FileMaker Pro 8 format, all relationships are converted to table occurrences in the new file. The developer may be tempted to create interfaces and logic based on those foreign table occurrences. This seemingly innocuous action can also break table context, since there is now script logic and interfaces in place that may shift table context away from the main table in the file.

Consider something as simple as this: In a converted Invoices file there is the Invoices Table Occurrence (TO). This is the only TO in the file that is not from a foreign file. All of the others come from other files. The other TOs are there to show relationships to other files. Therefore an Invoice Items TO will be there to describe the relationship between the Invoices and Invoice Items.

It might seem relatively harmless to create a new layout in the Invoices file based on the Invoice Items TO and use Go to Related Record to display a list view of Invoice Items related to the invoice. But unless you are careful, you run the risk of leaving the file with the table context set to Invoice Items. This will happen simply



if the user leaves that window open on the Invoice Items layout. This may cause problems the next time a user runs the “New Invoice” script and it creates a new record in the Invoice Items table instead of the Invoices table. This is more likely to be a problem if scripts that manipulate records or edit data are initiated from another file.

Although convert and restore has the potential to provide some early return on investment (ROI) and experience in using the new application there may be difficulties involved with extending or optimizing it. Luckily this is not the case. The new application model makes it possible to begin to immediately extend your solution without breaking logic that the organization is relying on every day to do its business.

Extend: Add a New Interface File

One of the most important changes to the FileMaker Pro 8 model is the ability to access data from a different file *as though it were in the same file*. Scripts and Layouts can be designed in one file but get their data from tables in another file. This very flexible feature is what makes it possible to begin to extend a converted and restored solution immediately.

As was pointed out above, it is difficult to add features to an existing file without potentially affecting the code that was already there from pre-conversion. So don't! Leave the old files functioning as they are. Instead create a new empty file and access the data in the old files. This new file has no legacy logic to get in the way.

Consider the advantages: The logic in the new interface file can be completely isolated from the rest of the system. Scripts and layouts that are designed in this new file won't affect the logic in the old files. In a sense, the developer has a clean slate. New features can be designed that use optimal FileMaker Pro 8 designs without concern for breaking the context of old logic.

Also consider that after some initial testing on an offline set of files, the developer can roll out this new file with new features to users by simply including it with the rest of the files on the server. New versions with new features can be rolled out to users whenever necessary. There is no need to have to take the files off the server and import data from the data tables, since there is no data, just interface, in this new file.

Instead of designing new features, the developer can rewrite pre-existing features in this new interface file. This approach may be the best if there is no urgency for new features or if some features really cry out to be redone using the FileMaker Pro 8 optimized design. As each rewritten feature is completed the old version can be retired and the detritus cleaned up. Using this method the developer can roll out features one by one as they become ready for use.

Note that it is possible to add temporary or permanent navigation scripts between the new interface and the old interface as desired. Some minor enhancements (new scripts, selected calculations) may be added to the old files, but it will be possible to keep these to a minimum.

Another variation on this theme is that the developer can bring over some old interfaces and logic from the original files because there is no need nor desire (nor budget) to rewrite them from scratch. FileMaker Pro 8



makes it much easier to “port” code from one file to another because FileMaker Pro elements now map by name instead of by ID when importing scripts or pasting layouts. The exact steps needed to accomplish porting code will be outlined later in this document.

The developer can decide how much of the old interface and logic is brought over from the original files. It might make sense to bring over all of the interfaces, or just a few scripts and layouts.

There is an important caveat to bringing over interface and logic from converted files. The logic in the converted files was based on the table context of one file per table. Moving it over into an interface file where that context is not enforced will require some modification. In short, some retrofitting of table context will be required, but since it can be done on an as needed basis, the scope of the problem should be less than when having to retrofit context across the entire solution.

Consider the benefits of this approach: The developer has the time to become familiar with designing solutions in FileMaker Pro 8. They have an active real world environment in which they can deploy solution components at regular intervals and that can provide the valuable feedback that is necessary for the success of a solution. The users begin to benefit from improved design very early on in the process, and the organization benefits from phased deployment of new technology and earlier ROI.

Eventually all of the interface can be consolidated into a single file or a few files if desired. Outdated and obsolete logic in calculation fields and scripts can be removed. The old interface is no longer used. The solution is now in a state of having a FileMaker Pro 8 optimized interface but its data tables are still contained in the original converted files.

Although the solution is closer to being FileMaker Pro 8 optimized, it isn't there yet. There is much to be gained from consolidating the data tables to fewer files. Once again the improved portability of FileMaker Pro 8 elements can help the developer do just that.

Consolidating Data Tables

The ability to directly edit file references makes it possible to change the data files from which the new interface file is pulling the data. The developer can re-point the file references of the new interface file at a new consolidated data file that contains the exact same tables and relationships as the separate data files. The process is relatively straightforward. First a new consolidated data file (or files) is prepared with exactly the same tables as the original data files had. Then a simple process can be followed that re-points the new interface file at this new data file.

This new “consolidated data file” could be the same file as the interface file, though there are significant benefits in the FileMaker Pro 8 application model to keeping the interface in a separate file from the data tables.

The data tables are consolidated into a single file by following a simple step-by-step procedure outlined below. All data fields (text, number, date, time, container) and all necessary calculation or summary fields, including



those that are referenced on layouts or in scripts, need to be recreated with the identical name. Some 3rd-party tools are available to help automate the procedure. Most specifically, FMrobot™ from New Millennium Communications can automate the process of creating new tables and fields, including field definitions, based on the fields in an existing FileMaker Pro .FP5 or .FP7 file.

There are a couple of techniques to getting the new interface file to correctly map to the new data tables, but they are very simple. First, the interface file is duplicated and all script and layout contents (not the actual layouts) are deleted from the copy. The original interface file is saved for a later step. Next, the file references are re-pointed at the new data file. This will break the TOs and relationships on the graph but they can be fixed by referring to the graph in the original interface file.

Once the graph has been restored, the scripts can be re-imported into the file from the original interface file. Finally, all the layouts can be re-pasted back into the new interface file from the original interface file and their tab order re-created. Since importing and pasting maps FileMaker Pro 8 elements by name, all references will resolve. This new interface file will now function as it did, but it will now access data in the consolidated data file, instead of the old separate data files. The only step left is to import the data into the new data file.

The solution has now been completely migrated to an optimized FileMaker Pro 8 design. It has a new interface file and a new data file. The old files are now gone. The developer has had time to learn how to use FileMaker Pro 8. The users have been able to benefit from improved workflow and interfaces. The organization has already experienced some ROI. The solution architecture is such that it can be used as a reliable base for ongoing development.

Following is the step by step methodology for migrating existing solutions to FileMaker Pro 8, affectionately known as either “CREC”, standing for “Convert, Restore, Extend, Consolidate”.

***Please Note:** FileMaker Pro 8 and FileMaker Pro 8 Advanced now include the ability to import tables. FileMaker Pro 8 Advanced also includes the ability to copy/paste scripts, fields, and tables.

Step By Step

1) Convert and Restore original behavior

- a) Modify, Convert, Test, Modify
 - i) See the documentation in the “Conversion Basics” and “Conversion Issues” sections for detailed information
- b) Deploy – [Stable Point](#)

2) Extend

- a) Add new features to a new interface file
 - i) Create a new (empty) interface file
 - ii) Create File References to each of the old files
 - (1) Note that FileMaker Pro 8 will create a single table and layout in the file, which you can ignore
 - iii) Create Table Occurrences on the relationships graph for each table (file) you want to access
 - iv) Create Relationships as needed



- (1) Note that to be able to display a portal or to “go to related record” in your interface file, you will need to create a relationship which may be identical to one which exists in the old file
- v) Now you can add new features – you have complete flexibility to create new relationships using the possibilities of FileMaker Pro 8 and to leverage the new relational model
 - (1) New Relationships
 - (2) New Layouts
 - (3) New Scripts
- b) Create temporary (or permanent) navigation between the new interface file and the old files
Deploy – Stable Point

3) Consolidate

a) User Interface

- i) Re-create the entire old interface in the new file or bring over pieces of your old interface and recreate other pieces using a new design
 - (1) This can be done all at once or in phases. Retire old interfaces as new ones become available to replace them
- ii) Bring over interface from the old files – step by step
 - (1) In the old files
 - (a) If any old files have only one Table Occurrence then create an extra one. This will allow “Go To Layout” script steps to import correctly.
 - (b) Add a file-specific prefix of your own choosing to script names. This will make them easier to understand once they are all in the same file.
 - (c) Add a file-specific prefix to layout names, for the same reason as for scripts.
 - (2) In the New Interface file
 - (a) Create file references to old data files
 - (b) Create Table Occurrences with exactly the same names as in the old files
 - (c) Create the same relationships as in the old files
 - (d) Create Layouts with the same names as in the old files (with prefix). Do not bring over the layout contents yet.
 - (e) Import scripts from old files into the new Interface File (rename all to remove ‘imported’)
 - (f) Copy and paste layout contents from the old files into the new files. All buttons and fields should resolve correctly. Scripts that referred to external files in the old system will still go to the old files.
 - (g) Point any “Perform Script [External]” script steps at the new local imported versions of them and add a preceding “Go to Layout” step to appropriately change table context.
 - (h) Retrofit Context – Make sure that scripts are always acting on the appropriate table
 - (i) Examine any Go to Related Record script steps and make sure that the desired target layout is selected
 - (ii) Some scripts may need to have a “Go to Layout” step added at the top to ensure the correct context
- iii) Clean Up
- iv) Deploy – Stable Point
- b) Data Tables



- i) Decide how many files are required.
- ii) Create a brand new empty file.
 - (1) Create tables
 - (a) Create fields
 - (i) Field Names should be exactly the same as in the original tables.
 - (ii) Calculation fields need to be defined as empty ("") calculations temporarily, since dependencies may not be in place yet.
 - (b) This can be automated to a great extent using FMrobot, available from New Millennium Communications, or by using the new table import features available in FileMaker Pro 8 and FileMaker Pro 8 Advanced.
 - (c) Create TOs and Relationships to support field calculations.
 - (i) These must be named exactly the same as they are in the original files.
 - (ii) Sometimes it may be necessary to change the names of the relationships in the original files as well, because relationships and or TOs from other files may have the same names. It may be desirable to change the names just for clarity as well.
 - (d) Paste calculations from the original tables into the new tables. Since the field names and relationship names match, calculations will resolve.
 - (e) Set any other field definitions that depended on the TOs and Relationships – lookups, validations, etc.
 - (2) Create any other relationships needed to support referential integrity and functional dependencies
 - (3) Re-point the interface file at the new data file
 - (a) Make a copy of the Interface File (IF 1). The copy will be referred to as IF 2. Put IF 1 away for later use.
 - (b) Modify IF 2
 - (i) Delete all the scripts
 - (ii) Delete all the layout contents, not the layouts themselves.
 - (iii) Re-point the file references at the new consolidated Data File
 - (iv) Remap Table Occurrences to the correct tables.
 - (v) Repair relationships. Use the graph in IF 1 for comparison.
 - (c) Import all the scripts back into IF 2 from IF 1
 - (d) Paste all layout contents from IF 1 to IF 2.
 - (i) Re-create the tab order
 - (4) Clean up (remove “ imported” from script names, etc.)
 - (5) Deploy – Stable Point

Appendix: FileMaker Pro 8 element mapping rules.

When you move FileMaker Pro 8 elements such as layouts, scripts, or calculations from file to file (or even in the same file), FileMaker Pro 8 tries to resolve any dependencies by name, not by ID as was sometimes the case in FileMaker Pro 6. This only applies to the process of bringing elements in. Once the elements are in place, IDs are used to maintain the links.



This is similar to the process of looking up a Contact ID by using Contact Name in an Invoice. The Contact Name is used to find the Contact ID when the record is being edited, but it is the Contact ID that is used to maintain the relationship.

For example, when a script is imported it finds all the fields, layouts, and table occurrences it needs by name. Once it finds them it uses the FileMaker Pro 8 internal ID of each element to link them together. If there are two layouts or scripts with the required name, then the first one is used. The same rules apply to copying and pasting layout contents and copying and pasting field calculations.

One exception in the FileMaker Pro 8 element mapping rules is when importing scripts that contain the Perform Script script step. From FileMaker Pro 8 Help:

“When importing scripts that contain the Perform Script script step, the link between scripts will be retained only if the linked scripts are imported at the same time. If a script with the same name is already present in the target file, no attempt will be made to link between the two scripts. If the Perform Script script step references an external script, the imported script will also reference the external script.”

The consequence of this is that references to pre-existing scripts in the same file will need to be reconnected manually.

About the author

While doing Genetics research in 1990, Todd Geist got hooked on database design. He left academics to form a FileMaker development company in 1997. In March 2000, he joined New Millennium Communications where he is now the senior software architect. Todd was the lead designer of Genesis Business Operating Environment® 3.0. Todd was also the lead designer of two of the most popular developer tools: Password Administrator and ScriptOrganizer.



Case Study: Migrating Using the Hub & Spoke Approach

Overview

The purpose of this paper is to provide an overview methodology that will help current FileMaker® Pro users upgrade their system to FileMaker Pro 8 using the Hub & Spoke Approach (HSA). This paper does not provide a step-by-step instructions but rather a concept overview for this approach.

Many of the benefits of FileMaker Pro 8 cannot be achieved through conversion alone. In addition, maintaining several complex relationships graphs in many files can be difficult. Since many companies do not have the time or resources to invest in rebuilding a solution from scratch, this paper will provide guidance on how current users can convert their files and then implement a series of changes in order to take advantage of the FileMaker Pro 8 new file structure on an as needed basis.

Methodology: The Hub & Spoke Approach

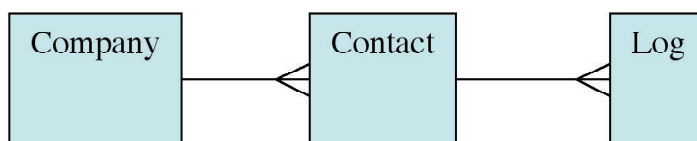
In many solutions there is one file that contains the bulk of fields, layouts, and scripts. We call this file the “Hub”. The hub presumably contains 75% of the programming logic, interface, scripts, etc. The other files may contain a great deal of data, but a lesser level of complexity in terms of fields, scripts, and layouts and are considered the “Spoke” files. In complex solutions, there may be many natural hubs within a system.

The goal of the Hub and Spoke Approach is to convert all the files, and then work to recreate the “Spoke” files as new tables in the Hub. Some advantages of this approach are that your system will be easier to maintain, that you have fewer files in which to manage accounts and privileges, and you can take advantage of modular scripting and layout techniques in FileMaker Pro 8.

In addition to achieving the benefits of having a many tables per file architecture, we expect the step-by-step nature of the approach will provide a viable and scaleable method for companies to implement changes over time.

Case Study: Contact Management Database

We have selected a simple three-file contact management system for our example. This system includes a Company file, a related Contact file and a related Log file.



For users who just do a straight conversion to FileMaker Pro 8 their architecture would look like this:

BEFORE CONVERSION	AFTER CONVERSION
Company.fp5	Company.fp7
Contact.fp5	Contact.fp7
Log.fp5	Log.fp7

For users who convert using the Hub & Spoke Approach their converted solution would result in one file with multiple tables and look like this:

BEFORE CONVERSION	AFTER CONVERSION
Company.fp5	Company.fp7 -Table 1 (Hub): Company -Table 2 (Spoke): Contact -Table 3 (Spoke): Log
Contact.fp5	
Log.fp5	

Pre-conversion activities:

As with any conversion, there are a number of steps you should take before converting your files. You will also need a copy of FileMaker Pro 8 Advanced and New Millennium's MetadataMagic. Some pre-conversion suggestions include:

1. Make a backup of your original files.
2. Make sure the passwords in each file are consistent and case sensitive (i.e. make sure there are no instances of "Master password" and "master password" as FileMaker Pro 8 would recognize them as two unique passwords.
3. Remove unnecessary field definitions, scripts and buttons (layout #125, layout copy 2 etc.). This is a small solution so cleanup is easy. In the case of extremely complicated solutions, or solutions where you are not sure if certain fields or layouts are still in use, do not delete them.
4. Run the DDR using FileMaker Developer 6.
5. Run MetadataMagic
6. Run MetadataMagic's File Reference Fixer to clean up relationships.



Conversion:

Once you have followed any pre-conversion activities, we suggest you perform an “alpha” conversion. If you have a lot of data in the files, make a clone and convert the clone. Essentially, run your system through the converter first, review the conversion log file, and then run the DDR. Your goal is to identify any issues that should be handled before conversion such as unnecessary relationships. For example, you may have relationships pointing to files you no longer use, or unnecessary duplicate relationships.

Once you have cleaned up your files, convert the files with all the data in them. This may take longer than the initial conversion. This is an important step since you will need to import data from external files into your new internal (spoke) tables.

Combining Tables into the Hub

The basic process of recreating tables in the hub file involves creating the tables and field definitions, resetting the relationships and value lists, recreating layouts and redoing scripts and buttons. Be sure to make backups at every stage of the conversion. That way you can roll back to prior changes if you make a mistake.

The following is an example of the steps taken to combine the Contact and Log table into the Company file. We will be providing updates and more details to this approach via our website (www.moyergroup.com).

1. Before converting our files, we identified the Company file as the “Hub”. We identified the Company file because it had the most fields, layouts, and more complicated scripting than the Contact or Log file.
2. Once we identified the hub file, we printed out the field definitions for the “spoke” files. In this case we printed field definitions for the Contact file and the Log file.
3. Next we made a functional reference list (FRL) of key features that this system must include. This list will be used to update and test the converted system and included items like “create new contact, search for duplicates, create labels, etc. Depending on the number of users and the overall complexity of the system, this list could be quite long. In other solutions, like our contact manager, the key is to make sure you have a list of objectives to test at the end of your conversion.

Create new tables

Using your printed field definitions, create all the field names from the Contact file being sure to use the exact same naming conventions. At this time, the goal is to create all the field names (text, date, number) but not define any calculations, lookups or auto-enter values. These definitions will depend on relationships that we will build in the next step.



Reset table references

Now that you have created new table occurrences (TOs), you need to repoint the existing external table references to the new internal TOs. When you created the new files, new TOs were created on the relationship graph. Do not delete these TOs at this time.

1. Select external TO.
2. Copy the TO name.
3. Redirect the TO to use the internal table.
4. Paste the original TO name (this will help you in accurately recreating your calculations and scripts).
5. If necessary, reset the match field for the relationship (unless fields have been created with the exact same internal FileMaker ID's the fields will not match)
6. Add TOs to reflect relationships in the spoke files (i.e. the Contact file points directly to the Log file.)

Complete field definitions

Now that the relationships are created, you need to complete all the auto-entry functions and calculations so they are based on the appropriate internal file reference. Use your field print out (if you make a .pdf you can copy and paste field definitions into the new file). Be careful to select the proper TO when writing calculations.

Import Data

The next step is importing data from the external files into the new internal files. In order to import data you need to be on the layout with the proper context (i.e., you must be on a Contact layout to import contact information).

For larger solutions, and or solutions that are live, you will want to develop the new version in clones, convert the data when the new system is ready, and import the live data into the new solution.

Copy and Update Layouts

When copying and pasting layouts between external and internal files be sure to set the context for the layout. An important tip is that when copying and pasting layouts, the table reference names in the external file must match the internal reference names. That way you will not have to reset the field names when they come over; it is important that as you copy and paste layouts between tables, you need to check all field references on each layouts.

If everything is named identically, copy and paste should resolve everything. Also note that if the scripts are in place with the same names before pasting the layouts, then the buttons will resolve to the correct scripts.



We suggest the best way to update functionality is to go through each of the layouts in a systematic way, and make sure buttons are working. This will require modifying and creating new scripts. As you modify or make a new script, we suggest you segregate the scripts that have been reviewed so you can distinguish scripts that have been tested. You will find that the new structure will render many of these scripts obsolete. For example, you only need one “Print Portrait” for all three tables. For complicated scripts you can still import them, but we do not advise importing all scripts as many of the scripts will be rendered obsolete by the new structure.

You will also need to review and update value lists in each of the new layouts.

Conclusion

As a final step you will want to review your functional reference list to make sure all the key components of your system are working correctly. Running the DDR using FileMaker Pro 8 Advanced is also an excellent way to review the new system structure.

While the HSA method takes time (this three file solution took about three hours to convert and modify), we believe this method will allow people to make changes to their systems over time. We also expect there to be third party tools to assist in functions such as importing field definitions that will reduce the amount of time needed to create new files.

About the authors

Molly Connolly is the Director of Business Development for The Moyer Group in Chicago. She joined the firm after running Thorsen Consulting, a consulting firm she started in 1996. There she developed FileMaker Pro solutions for a variety of education, advertising, and not-for-profit organizations. Prior to starting her company, Molly was a senior consultant for Ernst & Young where she gained cross-industry experience in financial and workflow management processes. She has spoken at the FileMaker Developer’s Conference and was the technical editor for the book *Advanced FileMaker Pro 5.5 Techniques for Developers*.

Bob Bowers, president of the Moyer Group, is the co-author of three books on FileMaker Pro, including Special Edition Using FileMaker Pro 8. In addition, he’s a columnist and contributing editor for FileMaker Advisor magazine, and is one of only a handful of trainers authorized to teach the FileMaker Professional Foundation Training Series.



The Separation Model: A FileMaker Pro 8 Development Method

Overview

The definition of the Separation Model is an architecture that separates the data layer from the presentation and business layer(s). Developers utilize the Separation Model (SM) to achieve a variety of goals, including; facilitate solution upgrades (i.e. updates without data imports), impose complex business logic and rules, emulate a transactional model and create modular re-purposeable solutions. The new relational model in FileMaker Pro 8, along with a host of new features, supports and encourages this development model more than ever before.

If you have a FileMaker® Pro solution prior to FileMaker Pro 8 that you plan to upgrade we recommend you un-hitch the FileMaker Pro 6 thinking and engage in new FileMaker Pro 8 solutions from the ground up. There are numerous benefits inherent in creating that solution in FileMaker Pro 8 using the separation model, all of which are best achieved by taking a ground up approach. At the end of the process you have a FileMaker Pro 8 solution optimized for your business and development needs; whether that be a modular system to be rolled out by department, or a complex system agile enough to keep pace with today's ever-changing business environments.

This paper presents not so much a recipe for migrating your previous FileMaker Pro solutions as an in-depth overview of a development model that is enhanced greatly by FileMaker Pro 8. A variety of developers have been implementing the separation model using previous versions of FileMaker Pro for a number of years. They have been creating solutions that attempt to classify the purpose of a file and assign it to either the data layer, interface layer, or business logic layer. The structure of files belonging to a typical SM solution created in earlier versions of FileMaker Pro is outlined below.



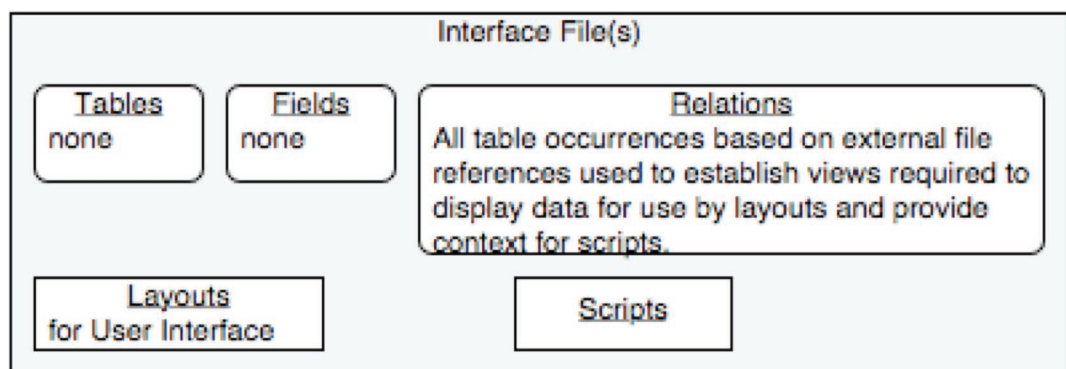
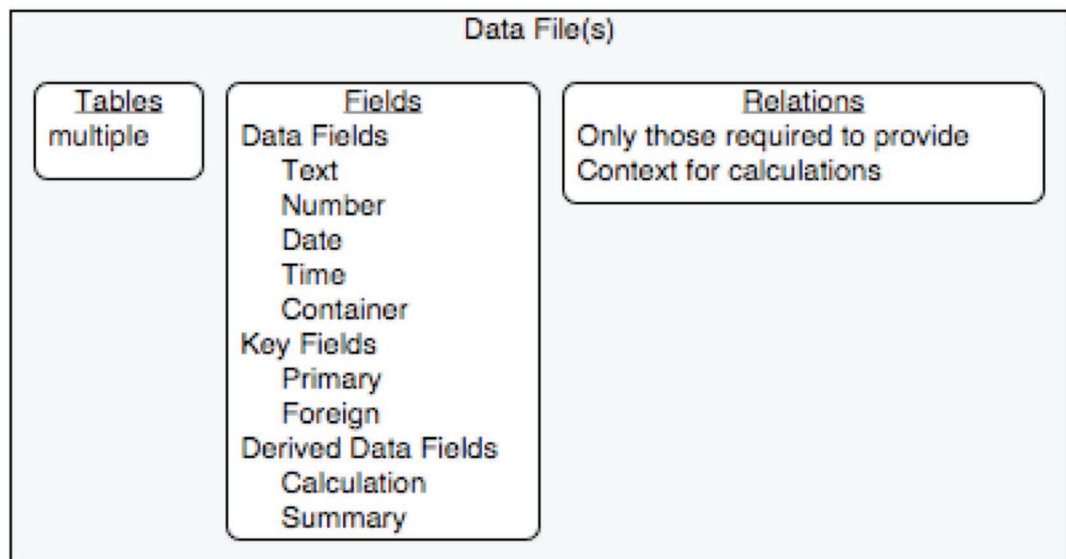
Data File(s)	Interface File(s)	Business Logic File(s)
Data Fields Text Number Date Time Container Key Fields Primary Foreign 'Pipeline' Redundant Data Fields Lookups Unstored Calcs Derived Data Fields Calculation Summary Logic Fields Global 5% of Solution's Scripting 1-2 List View Layouts 0 Form View Layouts Reporting Layouts Handful of Relationships	Key Fields Globals- lots! Unstored Calcs Derived Data Fields Unstored Calcs Logic Fields Global Text for Session Mgmt 85% of Solution's Scripting Most User Interface Layouts Large Number of Relationships	Key Fields Text Number Derived Data Fields Text Number Calculation Unstored Calcs Summary Logic Fields Text Number Date Time Container Global 10% of Solution's Scripting Reporting Layouts Handful of Relationships

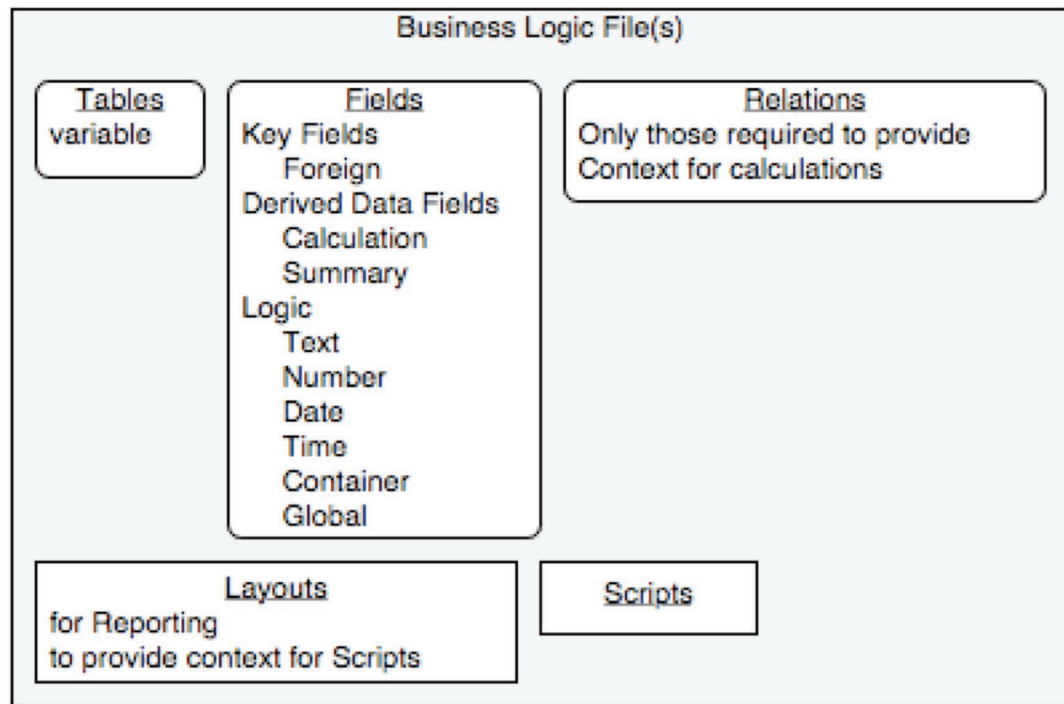
As you can see from the diagram, while separation is the goal, it was not completely attained. Key features in FileMaker Pro 8, most notably external table occurrences and the new relational model, allow developers to come dramatically closer to said goal.

External table occurrences, new to FileMaker Pro 8, allow one file to access the contents of another file as though contained within the current file. This means that we can define all data tables in one file and, through external table occurrences, access those tables in a separate interface file. External table occurrences do not require the developer to establish a relationship between the local file and the external file in order to gain access to the external file's tables and data. Rather, once a file reference is established, the local file has the ability to utilize any table within the external file in exactly the same manner as a locally defined table (with the exception of defining fields, this can only be done in the local file.) So now a local file can contain a list view, not only a portal, of data from an external file.

Because external file references do not require a relationship there is no need to create any tables or fields in the local interface file. The illustration below is an example of what an SM solution, which meets the same needs and solves the same problems as the typical SM solution designed in earlier FileMaker Pro versions, might look like when created in FileMaker Pro 8.







As we come closer to a complete separation of the data, interface and business application layers, we reap more of the benefits mentioned earlier and are able to leverage our development efforts over more projects.

Prior to beginning an explanation of solution modeling using the new FileMaker Pro 8 new relational model it is critical that the reader have a fundamental understanding of that model. Please take time to read and become familiar with the 'FileMaker 8 Relational Model' found in the second chapter of this document. We strongly encourage all developers to learn and understand the new application prior to launching any new development projects.

The Process

Phase I: Analyze the solution

Utilizing your development documentation, the FileMaker Developer 6 Database Design Report, or third-party products such as MetadataMagic, BrushFire, or Analyzer review the entire structure of your solution. For additional information regarding third-party products visit http://www.filemaker.com/products/third_party.html

Preliminary Review

Review the solution with key users or other developers in order to identify fields, layouts, scripts, and other



functionality that are no longer required by the solution. We refer specifically to those elements no longer used; layouts created for a specific purpose or person which no longer exist, abandoned fields and scripts. Abandoned functionality is clutter and may result in migration of more features than necessary.

Create a Data Dictionary

The goal in your data layer is to slim the files down to the extent possible to contain pure data. Therefore, to begin, study your file's field definition lists identified as containing data. Identify and categorize your fields as noted below. With the exception of retaining the logic behind reporting summaries, which will be migrated to a reporting file (in the business layer) most of the FileMaker Pro 6 centric thinking fields will be eliminated.

Data

- pure data – text, date, number, etc
(Bear in mind that you may have plain text or number fields that are holding derived data; data that was calculated in a business layer and stored in a text or number field.)
- keys

Business Logic

- derived data – calculated results such as extended price
- summary
- session controls

Interface

- any fields used for the sole purpose of presenting data (status current record, status current found count)

FileMaker Pro 6 centric thinking fields

- key fields used to pipe data (though there may still be a need to pipe data)
- fields used to display or search data from another file (unstored calcs, lookups)
- redundant data entry fields for tracking modifications and controlling record commits
- parameter passing

(This process is more quickly accomplished if you adhere to a naming convention. Further information regarding naming conventions is offered by Core Solutions: <http://www.coresolutions.ca>)

Security

Review and understand the security structure in the solution files. Reassess the security requirements for the solution in order to determine if this structure forms an appropriate base to utilize in your FileMaker 8 files. The security model in FileMaker Pro 8 is vastly improved and dictates that you re-engineer your security implementation. See [Upgrading to FileMaker Pro 8: How to employ the new, advanced Security system](#) by Steven H. Blackwell for an in-depth discussion.

Functional Specification

Refer to the original 'needs analysis' documentation or review the solution with key users in order to re-establish your understanding of the business requirements or purpose this solution is designed to address. Too



often we, as developers, fall into the habit of 'building a better mouse trap' when in fact the mice are long gone and the purpose of the trap today is to support the wobbly table leg. Reconnect with the problems this solution was originally designed to solve so that, moving forward, you can apply FileMaker Pro 8 in the most appropriate manner— leaving behind the work-arounds.

Phase 2: Select Data Model and Design Entity Relationship Diagram

With our data fields identified in Phase 1 you can more easily begin to establish a data model for your solution. We recommend you start by arranging data fields into tables, our goal is to achieve a normalized data structure where tables contain only data fields with related attributes. The challenge here will be to approach this task without imposing constraints and assumptions required by earlier versions of FileMaker Pro but no longer valid. In the past, data models needed to be coerced into a relatively low, fixed number of tables. For all practical purposes those constraints have now been eliminated and our architecture should reflect this.

Once we have established our table structure we can address the placement of tables into files/databases. This step was unnecessary in previous versions as files were, by definition, equated to tables. In FileMaker Pro 8 we have the possibility to create one file, or database, which contains all of the data tables our solution requires. (The actual limit of tables allowed in a file is 1,000,000.)

Taking the approach of placing all tables in one database will simplify greatly the creation of our solution. One advantage is the security schema for our data model will need only to be created and maintained in this one database. Another advantage is the creation and maintenance of our relationships graph(s). Relationships graphs will be necessary in the data file(s) to provide context for any required calculated fields. Relationships graphs are also required to enforce referential integrity, though as we will see later in this document, there is no requirement that this graph be placed in the same file/database as the tables themselves.

However, there are several factors which may recommend more than one file/database be used in a data model. For example, if you intend to deploy a modular solution in which your clients have the option of purchasing certain modules, the solution might be better served to modularize the data files as well.

If a solution will be shared by multiple users and employ the features available in FileMaker Server 8 or FileMaker Server 8 Advanced be aware that backups are performed on a file-by-file basis. So, if your solution contains highly volatile data that requires frequent backups, as well as relatively static data that may suggest a periodic backup, it may be preferable to divide these tables into different databases in order to meet the needs of the solution more appropriately. Also, a time may come when we wish to update this file by replacing it with a new version. That update process will require that all data tables be imported into the new file.

Understanding that it is possible, but not necessarily preferable, to have all data tables in one file we are presented with the challenge of determining the appropriate number of files for our solution. While our inclination is that it will make sense for our data files to include groups of tables, there is also a school of thought which suggests we may, in the end, prefer to assign each table to its own file. Time and experience shape best practices in this area. The limit of 125 files hosted on a single FileMaker Server 8 may be a factor in some circumstances. Regardless of the number of files in the model we select, data housing is the primary purpose of the files.



Once we have determined the appropriate table and file/database structure for our solution we must determine the appropriate location of the relationships graph(s) that will enforce referential integrity. This is the graph(s) that imposes the delete rules for your data structure. As these rules and associated graph(s) are a key component of your solution, careful thought and consideration should be given to placement. Currently there is no 'best practice' recommendation however, we encourage you to consider the following:

- A relationships graph can be composed of table occurrences (TOs) from local tables, external tables, or any combination of the two.
- Regardless of the location of relationships graphs that contain the delete rules for your solution, these rules will be imposed if that file is open.
- One way to require that the file containing the rules for referential integrity be open is to place a TO of a table* from that external file into each of your interface files and reference that TO with a blank layout. (* This table need not have any defined fields).

Bearing in mind these points, we like the idea of placing all TO groupings (TOGs) that enforce referential integrity in one location so that we can get the 'big picture' and review and maintain this in one place.

Phase 3: Select Interface Model

It is possible to create a FileMaker Pro 8 database that contains no locally defined tables. The relationship graph in this file can be populated with tables based in external files. The TOs on this graph function just as TOs based on locally defined tables. (The only indication that a TO is attached to an external file rather than the local file is the font style of the name of the TO; regular = local, italicized = external.) Relationships can be established between these TOs that enforce functional dependency and referential integrity as well as specifying a sort order.

When layouts are created you are required to specify a TO which will provide the context for the layout. Context defines the tables that will be available to that layout for field and portal placement, as well as the path that will be used in order to view related data. It is important to note that layouts (and layout elements such as fields, portals and buttons) attached to TOs associated with an external file will behave in exactly the same manner as TOs associated with a locally defined table. This means list view, table view and find mode, in addition to all menu commands such as new, delete, duplicate, sort etc. So now we can have a file that contains all of the user interface elements native to FileMaker Pro 8 files without the file containing any data.

In the past it was common in a SM solution to keep all interface interaction in one file. This decision was driven in large part by constraints in previous versions of FileMaker Pro that are no longer applicable. Previously it was extremely time intensive and tedious for a developer to port functionality from one file to another. In FileMaker Pro 8 imports (of scripts) and pastes (of layout objects) are now resolved by name, simplifying the process of transferring complex layouts and scripts from one file to another.

What does this mean to our interface model? It is now feasible to consider using multiple interface files which interact with the same data file(s). One might choose to create separate interface files for different departments, i.e. accounting, sales, and shipping might each have their own unique file(s). Or a developer may



choose to implement different versions or feature sets in various interface files, for example demo, light, full, and web. Each of these options represents a different interface model. The common theme among these models is that in every case data is stored in file(s) outside the file(s) provided for user interaction.

It will be up to the developer and the business requirements to dictate how many interface files should be used for any given solution. There are pros and cons associated with using multiple interface files or attempting to constrain all user interaction to one file. For example the relationships graph created in one file is not accessible by another file. So, if you have several interface files that require the same table occurrence groupings you will need to create and maintain these in both files. On the other hand, if you have a complex solution you may find your relationships graph extending beyond your organizational threshold. Creating additional interface file(s) to address specific areas of your solution may provide you with a greater ability to manage and control your solution. Therefore it is imperative that the developer considers carefully the advantages that multiple interface files might offer.

Phase 4: Select Business Logic Model

Microsoft's definition of the business layer, or business logic, seems quite simple to grasp upon first blush; "The business layer implements business rules by checking limits, validating data, and providing calculated or summarized data, etc."¹ However, those new to implementing the SM with FileMaker Pro often find it challenging to understand this layer. This is because much of what would be classified as belonging to this layer in a traditional RDBMS is part of the functionality built in to FileMaker Pro. Because many of these features are 'simply there,' we as FileMaker developers often tend to mentally group elements from this layer into either the data or interface layer.

You may not think of it as such, but business logic already exists in your database solutions. Field validation, value lists, summaries and summarized reports are elements of the business layer. Calculations used to derive useful data as well as the rules governing how users will access and view data are all business logic. The FileMaker Pro find mode is another example of business logic within the application. Developers may enhance this logic by applying business rules to layout objects, field behavior options for example.

If we understand that the business logic layer includes those elements that will enforce the structure, rules and order required by organization, and that many of the features we love about FileMaker Pro roll these elements into the file structure, why would we want to even consider this as a separate layer?

We separate business logic in order to re-purpose code, enhance modularity, provide a more unified understanding of a process, and to ease maintenance and future development efforts. Business logic elements inherent in FileMaker Pro features that provide no true benefits if separated are not candidates for explicit separation.

The extent of separation of business logic from the data layer and presentation layer depends upon many factors. First, there are certain inherent FileMaker Pro provisions that typically are not separated. Field validation being the topmost feature and would only be separated for the most regulated environments. Calculated data may well reside in the data layer, even though it is more appropriately separated. Calculated, or derived data, fields are an example of where explicit separation may not be recommended, but we can imply



separation by utilizing naming conventions and field headings in our table structure. See the final section of this document for an advanced treatment of separating derived data.

Some value lists are more appropriately defined in the presentation layer while others might benefit by separation into the business logic layer. For example, the value list for a contact salutation (Mr., Mrs., Ms, Dr) is not a list that changes or requires frequent modification. As such, it is a safe bet to define this list in the presentation layer. A value list for departments within a company, on the other hand, is in constant flux. Departments are phased out, renamed and combined. This list will be better managed as a table of information, separated in the business layer.

Moving value lists from the FileMaker Pro application feature into a table is one example of business logic stored in table format. There are many others such as: user preferences, graphic resources used by interface elements, localized text for dialogs stored in tables, and text strings used by plug-ins. It is important that we begin to recognize and categorize these types of tables as part of the business layer rather than objects belonging to the data layer.

The “Evaluate” function in FileMaker Pro 8 allows a calculation to reference a textual representation of the formula that resides in a text field in a table. Using this feature it is possible for a table to store some of your calculation logic. This formula would be available to fields using auto-entered calculations, calculation fields and script steps. By placing a formula in the business logic layer you can programmatically modify it or allow users access to define or modify this field under very controlled conditions.

FileMaker Pro 8 provides convenient methods for searching on any layout. Since the developer has the ability to lock fields down for entry or for searching, the rules of business are embedded in the presentation layer. Your solution may benefit by keeping the search process in the presentation layer. On the other hand, the business rules of searching may be more easily managed when separated from the presentation layer.

How do we decide when to explicitly separate business logic elements? Solution upgrades and maintenance drive the most often-separated business logic elements. You can reduce the amount of development time and impact on the user by separating functions that are prone to frequent development changes.

The business layer element most often separated is the reporting file. Reporting files are popular for SM developers as well as more traditional FileMaker developers. Reporting files keep complex layout structure out of your data files. You can update a reporting file and easily swap it out without disrupting the use of the database system. You can provide a reporting file for ad hoc reporting which allows users to create their own reports without giving them access to your more sensitive files.

Display elements are often contained in separate files. For example, the film production company might create badges that display a blue border for the international crew, a red border for the local crew, and a green border for the company staff. These are interface elements that follow guidelines of the business and typically are separate from the presentation layer for ease of maintenance.

Even searching is becoming a more popular function to separate from the data and presentation layer. Providing the user with a convenient one-stop shop for searching a list of fields over a complete set of tables is only one reason to separate the query process. It also reduces the amount of logic you need to maintain throughout



the presentation layer. You do not have to ensure that value lists are accessible on every layout in which a field might be searched or that certain fields are shut down for searching. You can create more easily maintained rules for personnel access. All searches are created from a single screen, searches can be saved, edited and viewed in a more readable format. The query engine can port from one solution to the next. And most importantly, the query process is available for modular solutions.

In order to create modular solutions or a solution that is re-purposeable, the business logic must be separated to a greater degree than the single-use custom solution. Imagine creating a solution for a clothing manufacturer and by changing a handful of field values in a business logic file the solution is on its way to becoming useful for an interior designer. This is the concept behind re-purposing a solution. A contact manager / project management / invoicing system is a prime example of a solution that ought to be re-purposeable. The basic functions are the same but the names and rules change. By isolating categories of contacts, discount rules, and project types in a business logic file, you can literally change the names to re-purpose a solution. This is a bit of oversimplification, as there is always a level of customization required in re-purposing a solution, but your core work evolves to benefit all solutions. This is especially attractive for the upgrade path of your existing customers.

A modular solution utilizes an architecture that is highly separated. A number of presentation files access the same core set of data files and similarly a core set of business logic files. Modular solutions are attractive for the large workgroup situation in which multiple departments, each with its own required functionality, access and manipulate data, but for which the data rules are not changed across departments. Maintenance is simplified, upgrade paths are more easily managed and budgeted, user loads are leveled and teams of developers can work more independently.

There are also special cases in which business logic will be separated to a great extent. Consider the regulated industries that answer to the U.S. Environmental Protection Agency (EPA). The EPA puts out regular updates for rules and regulations regarding, among other things, pollution control limits. The rules for validation of field input are changing all the time. The rules will never apply across the board for every record in the database. To put all that logic in the field validation area of FileMaker Pro 8 is unwieldy. In this case, the business rules for validation would be better managed outside the inherent FileMaker Pro 8 field validation area.

A careful review of the needs for field validation, value lists, reporting, calculation, and the flexibility within each of those areas over time, as well as predicting the need for upgrade and enhancement, all factor into your decision as to how much business logic separation you will employ in your solution. No fast and hard rules apply and the level of business logic separation you employ in a solution today may differ significantly from the level of the next solution you deploy.

Phase 5: Create your Data file(s) and tables

You will utilize your updated data dictionary, prepared in Phase I, to create the requisite tables and fields. At first, when modeling and initially implementing the solution architecture, only those key fields necessary for referential integrity should be created. In the file(s) in which you have decided to locate the functional dependency relationships, implement the relationships graph, adding all necessary file references, table occurrences, and relationships.



Phase 6: Create Interface and Business Logic Elements

This is the substantial phase when we create fields, table occurrences, relations, layouts, and scripts that will be used by the individuals in the organization to interact with the system.

Phase 7: Import Data and Test

Once you have migrated all elements of your existing solution into the new FileMaker Pro 8 format you will want to import your existing data into your new solution and thoroughly test functionality.

Features that Support the Model

Some of the key concepts inherent in the new architecture and functionality are:

- a) visible and editable file references
- b) external table occurrences
- c) multi-table relationship joins – multiple predicates, relative operators
- d) script parameters
- e) security / privilege set features
- f) locking and committing records
- g) portal ranges
- h) the “Evaluate” function and custom functions
- i) multiple windows

Understanding and utilizing these new FileMaker Pro 8 features effectively will produce solutions that come much closer to meeting the goals of separation and with less development time than in previous versions of FileMaker Pro.

As we have seen in the previous section, external table occurrences, a new concept introduced in FileMaker Pro 8, provide a huge step forward in supporting the separation model.

The features of the new relational model eliminate a myriad of previously necessary FileMaker Pro elements such as calculated match keys, multi-keys, and external calls to the data file to search and return results. This is a huge step forward for developers who have used the SM in the past and solves some of the stickiest issues encountered by those developers.

In the past SM developers have created highly controlled and scripted solutions. While some of this may no longer be required, many scenarios will persist that necessitate solutions to rely on a heavily controlled and scripted environment. Script parameters, a new feature in FileMaker Pro 8, allow more script re-use and if used effectively, will reduce the number of scripts in your solution. The script parameter feature allows you to pass a calculated parameter to any ‘Perform Script’ step – either directly from a button or within a script. For example, if you have a string of buttons, each linking to a letter in the alphabet, instead of creating 26 scripts, you can create one script and have each button pass the appropriate letter of the alphabet to that script.



The new account based security model and increased granularity provided by the new model will allow developers of highly controlled solutions to rely on built-in FileMaker Pro 8 security to control many actions which previously would have required complicated scripting and UI development.

In FileMaker Pro 8 record locking and committing has moved closer to a transactional model. A user can now enter a field (for example to copy a name to the clipboard) without establishing a record lock. Record locks are not established until data has been edited. The process of committing records has also been modified and reduces the need to use globals and additional layouts for data entry in highly controlled environments.

With the ability to set both the initial portal row to be drawn and the number of subsequent portal rows, developers should be able to eliminate much of the wrangling done in earlier versions with match keys in order to achieve certain UI elements. A good example where this feature simplifies the interface is the calendaring interface, where the user wants to quickly page down to the next day/week/month.

The new 'evaluate' function as well as the ability to create custom functions using FileMaker Pro 8 Advanced will extend the possibilities for separating business logic from the data and presentation layers. These features are also key players in separating derived data that is discussed at the end of this document for those who want to take separation a step further.

The ability to have multiple open windows for a file is the crowning glory for eliminating the passing of found set results, data entry, and a universe of possibilities upon which we may not have a complete grasp.

Before you can identify how these elements best enhance the structure and functionality of your solution a careful review of the architecture and presentation needs of your system is necessary.

Further Thought: Derived Data

Working in FileMaker Pro, developers are often lulled into believing that the best (often only) place to store derived data is within the same file/table as the data upon which the data is based. For example a LineItems table might contain the fields 'quantity' and 'selling price'; the question of where to place the 'extended price' field is rarely made. The FileMaker Pro application interface implies that the best solution is a calculated field in the LineItems file. However, there are various situations that suggest that a more appropriate storage space for this type of data is a separate table. This concept existed in the FileMaker Pro development arena for several years, most notably represented as a reporting file.

The new relational model in FileMaker Pro 8 allows us to separate derived data from data tables more effectively. So when we discuss selecting a data and business logic model we need to bear in mind that we are making a determination about placement of derived data fields.

Reasons for separating the inherent FileMaker Pro 8 business logic from the data layer (e.g. calculations, summaries, auto-enter features, and field validation) will depend on solution requirements and may include any of the following; table and logic updates, maintenance, modularity, solution re-purposing, performance and



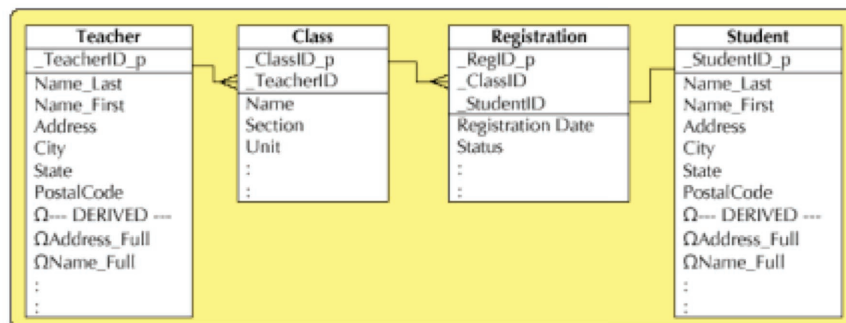
security. We can't imagine a solution that doesn't use field validation and calculation to some extent but there are compelling cases where separation proves useful.

Two examples are presented below, these are neither exhaustive nor individuated. Rather they are presented in order to illustrate and describe various methods which FileMaker Pro 8 enables developers to consider when architecting new solutions. We will discuss some of the pros and cons inherent to each method. It is assumed the reader will select those elements from each which apply most appropriately to their specific situations.

Combined Data and Derived Data

This structure will be most familiar to developers and places data and derived data fields in the same table within the data file(s). Derived data is achieved using calculation fields or posted to fields (text/number/date/time etc.), in which case, it is presumed that the business logic layer is performing the requisite calculations and posting the results in the data table.

While this is the most straightforward model, and for that reason may be the most commonly employed, we suggest that the appropriate type of derived data to be included in this method are those of an enduring nature. If the business rules applied in the calculation are not likely to be modified throughout the use of the solution, full name for example, we would categorized this as enduring. Please note in the illustration below that even though we have combined elements of the business layer (derived data) into the data layer we are imposing implied separation on the fields by utilizing field headers and naming conventions.



PROS:

This method is most easily and quickly understood.

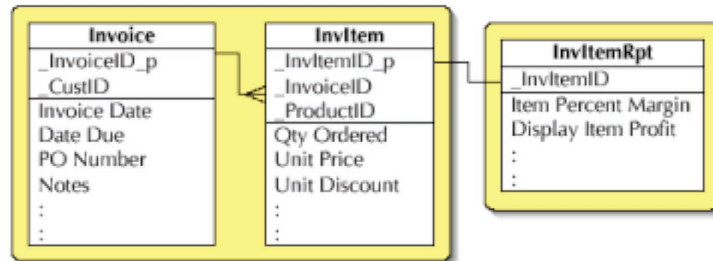
CONS:

In order to allow someone to edit a calculation field, access must be given to the 'define database' dialog.



Separation of Derived Data

This structure places data and derived data in separate tables that exist in separate files (databases). The illustration below shows an example of derived data that might be calculated and stored in a separate table, in this case the `InvItemRpt` table that is located in another database file.



This model also has two variations; one that uses traditional calculation, or auto-entered calculation, fields to store the derived data, and one that relies on functionality available in FileMaker Pro 8 to store the calculation equation as data in a FileMaker Pro 8 database. This second method makes extensive use of the “Evaluate” function to store calculation equations in a table of business logic.

Utilizing the “Evaluate” function to store calculation equations will certainly carry with it restrictions and limitations. It is still too early to make specific recommendations with regard to when this feature will be most appropriately used; however, it appears to be a very valuable feature.

PROS:

If ‘average’ users will have access to sort and export data fields, this structure could simplify these dialogs.
Separation of business layer elements from data layer.
Ability to programmatically change business logic.

CONS:

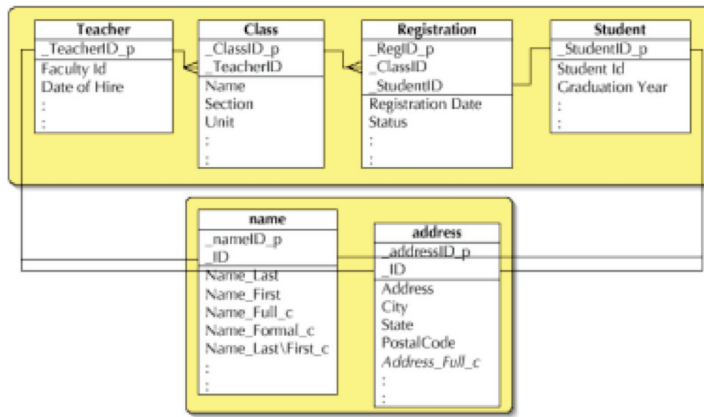
More thought required to set up.

Further Thought: Data Modeling for Modularity and Re-Purposing

The concept of re-purposing a solution ranges from the re-use of core files and code from one project to another to a complex solution that is so incredibly flexible in its design that by changing content, it can be deployed for a variety of clients. This is not the same as a vertical market solution, but reaps some of the same benefits – upgrade path, increased profits. The more business logic is isolated into its own layer, the closer the solution becomes to becoming re-purpose able. Modular solutions also employ a core set of data and business logic files (value lists, search engines, derived data and navigation pointers) that are accessed by modular presentation/business logic elements.



One example of this structure isolates a core grouping of data fields that form the backbone of other solutions the developer intends to create. By isolating these fields in each solution the developer can easily and quickly re-purpose them for a variety of different solutions for different clients.



PROS:

- Highly re-purposeable.
- Upgrade path across client base.
- Ability to create demos quickly.
- More rapid development.

CONS:

- Not as easily understood 'at a glance', more complex relations graph.

Conclusion

The new relational model in FileMaker Pro 8 along with a host of new features, contribute to new discussions and possibilities for achieving separation in FileMaker Pro 8 solutions. The Separation Model, an architecture that isolates the data layer from the business and presentation layers provides developers with the ability to create modular and re-purposeable solutions that reduce development time and hypothetically increase our profits. Separation can also provide the database administrator with a more flexible environment in which to maintain solutions.



About the authors

Colleen Hammersley is the founder and director of DataWaves, a FileMaker Solutions Alliance Partner Member. DataWaves is located in Suttons Bay, Michigan, where Colleen specializes in using FileMaker Pro to provide practical solutions to everyday chaos.

Wendy T. King, president of Database Mechanic in San Francisco, has developed custom FileMaker solutions since the days of FileMaker 2. The variety of her clients includes wastewater treatment facilities, cosmetics and clothing manufacturers, event planners, and trucking companies. Wendy is a long time member of the FSA and a contributor to FileMaker Advisor Magazine.

(Footnote)

¹ Microsoft SQL Server 7.0 Server System Administration Training Kit (Microsoft Press, 1999).



Bridging .fp5 and .fp7

Worlds Apart

FileMaker® Pro 8 is a revolutionary step forward, but it is a different piece of software than its previous versions. FileMaker Pro 8 databases are not compatible with FileMaker Pro 5, FileMaker Pro 5.5 or FileMaker Pro 6 databases. They have different file formats. FileMaker Pro 8 cannot open .fp5 files directly nor can earlier version open FileMaker Pro 8 files directly. You can still host FileMaker Pro 5, FileMaker Pro 5.5 and FileMaker Pro 6 and FileMaker Pro 8 databases on the same network, but you won't "see" the hosted .fp5 files in FileMaker Pro 8, nor will you be able to "see" .fp7 files through earlier versions.

This article is a high-level discussion of how we might "bridge" .fp5 and .fp7 databases. By high-level, I mean that I will provide some background knowledge and talk about concepts and strategies, but I won't go into the details of how you would implement these strategies. My goal is to provide information that might be helpful towards the larger task of thinking and planning for conversion or migration.

Why bridge? In another context, it might be simpler to say "share" instead of bridge, but I think that the word bridge is useful because it reinforces the point that .fp5 and .fp7 databases live in separate worlds. At its most basic level, bridging means sharing data by exporting and importing data in a format that is comprehensible to FileMaker Pro. At a more advanced level, bridging refers to the process of reading and writing to FileMaker Pro databases in a controlled fashion to avoid duplication and error.

Before you go about building bridges between .fp5 and .fp7, I would encourage you fully to explore migration and conversion options outlined in other articles. That said, I can see some possible scenarios where bridging might be a consideration. For example, you might want to continue to use a legacy FileMaker Pro 6 database solution that is being phased out even as you phase in its replacement in FileMaker Pro 8; or you might have a FileMaker Pro 6 solution that is currently serving a mission-critical function such as a database back end for a CDML web-form that needs to stay in-place until it can be migrated.

Ultimately, whether you want to have .fp5 and .fp7 databases coexist is a business decision. This is an important decision, but how we arrive at this decision is beyond the scope of this article. However, I'd encourage you to consult the various articles on methodology and cost-benefit assessments if you are interested in these discussions.

Definitions



Distributed database...

Multiple copies of a database (in whole or in part) installed at various locations

Synchronization...

The process by which two different copies of the database are made to conform to each other

Replication...

The process of creating an exact copy of data or a database in its entirety

Reconciliation...

A decision-making process in which two different copies of data or a database in its entirety are made to conform or allowed to be non-conforming to each other

Race condition...

A condition where two or more edits are being made simultaneously to the same data with the possibility of one or more edits being lost

Data collision...

The loss of data due to a race condition

Irreconcilable Differences

To begin, let's lay out the principal challenges in sharing data between .fp5 and .fp7 databases:

1. *Incompatible file formats.* The FileMaker Pro 8 file-format is different from its predecessors. FileMaker Pro 8 cannot read FileMaker Pro 5 or FileMaker Pro 6 file-formats natively and vice-versa. You can convert a FileMaker Pro 5 or FileMaker Pro 6 file to FileMaker Pro 8, but you cannot go the other way.
2. *No direct connectivity via relationships.* You can host FileMaker Pro 8 and FileMaker Pro 5-6 databases on the same network, but you cannot create relationships between the two. They can coexist, but they cannot share data directly.
3. *No cross-version imports.* At the time of this writing, you cannot directly import a .fp5 file into a .fp7 database. If you try to import a .fp5 file into a .fp7 database, FileMaker Pro 8 will tell you the .fp5 file needs to be converted first. Additionally, you cannot import .fp7 data into an earlier version's database.
4. *Changes to ODBC/JDBC in FileMaker Pro 8.* In FileMaker Pro 5 or FileMaker Pro 6, you set up your FileMaker database as an ODBC/JDBC data source through the FileMaker Pro client. Other applications that want to access FileMaker Pro data via ODBC/JDBC must go through a FileMaker Pro workstation with the proper data-access companion plug-ins installed. In FileMaker Pro 8, you can import data with FileMaker Pro 8 but you cannot set it up to be an ODBC/JDBC data source. FileMaker Server 8 Advanced can be set up as an ODBC/JDBC data source. Note: as of the publication of this document, FileMaker Pro 8 and FileMaker Server 8 Advanced can not be used as an ODBC or JDBC data source on Mac OS X.



5. *Changes to web connectivity with FileMaker Pro 8.* In FileMaker Pro 5, FileMaker Pro 5.5, or FileMaker Pro 6, you can use the web-companion plug-in with FileMaker Pro (the client) to make your databases web-accessible. That is, you can publish data using CDML or export data via XML over the web. In FileMaker Pro 8, CDML is no longer available. Furthermore, all other custom web-publishing abilities are now supported only in FileMaker Server 8 Advanced but not by the FileMaker Pro 8 client. Bottom line: FileMaker Pro 8 is no longer a data source that is accessible by plug-ins and APIs using the Web Companion. You can import XML data with FileMaker Pro 8 via the web but you cannot export XML data. Note that I am strictly talking about using FileMaker Pro 8 to export data via the web. Instant Web Publishing through FileMaker Pro 8 is very much alive and well. In fact, IWP in FileMaker Pro 8 is vastly improved and significantly more powerful than it was in previous versions.

Bridging Strategies

If we cannot connect .fp5 and .fp7 files directly, what can we do? In a sense, since .fp7 and .fp5 files are not entirely incompatible with each other, we have to approach the problem as if they were different databases platforms. In general, there are three ways to bridge .fp5 and .fp7 databases. You can export and import sets of data between the two. Second, you can establish some kind of direct database connection that will allow you to push bits of data between them. Finally, you can choose to synchronize or replicate your different data sources. I'll discuss these below roughly in the order of increasing sophistication.

Convert to FileMaker Pro 8 and import

The simplest way to get data from FileMaker Pro 5, FileMaker Pro 5.5, or FileMaker Pro 6 databases to FileMaker Pro 8 databases is to convert existing .fp5 files to .fp7 before importing them into FileMaker Pro 8. Since you cannot open a .fp7 file in FileMaker Pro 6 (or earlier) this isn't an ideal approach for anything but the most basic tasks of uploading data to FileMaker Pro 8 databases from .fp5 files. At the time of this writing, it doesn't appear as if you can programmatically (with a script) convert .fp5 files and save them, so if total-manual operation isn't sufficient, keep on reading.

Note

Actually, this isn't quite true. You can create a script in FileMaker Pro 8 to convert an existing fp5 file but there doesn't seem to be a way to control where the converted files are saved. You'll still be prompted with a dialog to save the converted file to some location and there doesn't seem to be a way as of this writing to save without a dialog box.

Exporting and importing via a compatible file format

Data from .fp5 and .fp7 databases can be exported in the following compatible file formats: tab separated, comma-separated, SYLK, DBF, WKS, BASIC, Merge, HTML and XML. In the simplest scenario, data that need to



be shared between a .fp5 and .fp7 database can be exported from one database in one of those formats and then imported on the other end. If needed, this process can be automated by scripts and scheduled to run periodically.

Import and export via XML/XSLT

The strength of the standard import and export approach is its simplicity. If the data you wish to share have identical structures that is if they have same tables and fields, imports and exports can be an effective way of exchanging data. Here's a typical case:

Table 1. Students.fp5 file

Field	Type
StudentID	Text
Name	Text
Sex	Text
Grade	Num
ClassOf	Num

Table 2. Students table in MySchool.fp7 database

Field	Type
StudentID	Text
Name	Text
Gender	Text
Grade	Text
YearOfGraduation	Text

In the scenario above, importing and exporting between the Students.fp5 database and the Students table in the FileMaker Pro 8 database, MySchool.fp7, is a relatively straightforward matter. They have identical table structures even if they don't have exactly the same data types. We could run imports manually in both .fp5 and .fp7, line up the fields, select the appropriate import order and save our imports as scripts that can be reused.

However, it is another matter all together if we are trying to exchange data between different relational structures. Consider an import between these two relational structures:

Structure 1. Flat file in .fp5

Table 3. Students.fp5

Field	Type
StudentID	Text



Name	Text
Sex	Text
Grade	Num
ClassOf	Num
Class1	Text
Class2	Text
Class3	Text

Structure 2. Relational database in .fp7

Table 4. Students table in MySchool.fp7 database

Field	Type
StudentID (pk)	Text
Name	Text
Sex	Text
Grade	Num

and

Table 5. StudentClasses table in MySchool.fp7 database

Field	Type
CourseNumber	Text

Here, simply saving our exports in a compatible merge file or tab-separated file isn't sufficient. We need a way to *transform* the structure of the exported data from one database to another. This is where XML and XLST come into play.

If you are new to XML, I'd recommend visiting *XML Central* at www.filemaker.com/xml. XML is a markup language that describes data and its structure. You can save exported data from .fp5 files and .fp7 files as XML files. FileMaker creates these XML files in a format that is readable by both FileMaker Pro 6 and FileMaker Pro 7/8.

Note

If you are familiar with the concept of grammars, note that FileMaker Pro can export XML data in either FMPXMLRESULT or FMPDSORESLT. However, FileMaker Pro 5-8 will only import XML data that has been formatted with FMPXMLRESULT. FMPDSORESLT has been deprecated in FileMaker Pro 8. It exists to support legacy functionality, but you should avoid using FMPDSORESLT when possible.



The beauty of the FileMaker Pro XML import feature is that we can tell FileMaker Pro to transform a particular XML data file before importing it. We can tell FileMaker Pro to rename fields or even to restructure the data with transformation stylesheets, otherwise known as XSL. This business of transforming data with XSL stylesheets is also called XSLT.

Using XML and XSLT, we can take data from one database structure and coerce it into a suitable structure on import. Thus, to solve the problem above of importing data between different database structures, we might write a stylesheet to “flatten” the exported data into the StudentCourses so that it can be imported nicely into the Students.fp5 database file. The catch is that these transformation scenarios are likely to be unique to your .fp5-.fp7 bridging needs. You, or someone who understands what you want to do has to write these XSL stylesheets for your particular transformation scenarios.

ODBC

ODBC is an application programming interface (API) for executing SQL statements. SQL stands for structured query language and is the common “language” for communicating with most relational databases. The ODBC approach to bridging .fp5 and .fp7 databases offers a little more power over file-based import/exports at the cost of some added complexity.

You can find additional instructions on setting up FileMaker for ODBC sharing in FileMaker Pro 8 through the application help system.

In FileMaker Pro 6, you must configure a copy of FileMaker Pro (not FileMaker Server) to be the ODBC data source by installing the FileMaker ODBC driver and the Data Access Companion plug-ins on the computer running FileMaker Pro 6. You cannot configure FileMaker Server directly to be the ODBC data source. There is a subtle difference between the concept of setting up FileMaker Pro to access other ODBC data sources and that of setting up a FileMaker Pro database to be accessed as an ODBC data source. To connect FileMaker Pro to other ODBC data sources, you need to make sure you have the proper database drivers and a properly configured data source name (DSN). DSNs are typically configured using your operating system’s ODBC control panel or driver manager. Here is table to illustrate the ODBC capabilities among FileMaker Pro 6, FileMaker Server 5.5, FileMaker Pro 8 and FileMaker Server 8:

Table 6. FM ODBC Capabilities

	FMP 6	FMS 5.5	FMP 8	FMS 8
Can be configured to be an ODBC Data Source?	Y	N	N	Y
Can connect to remote data sources via ODBC?	Y	N	Y	N

One advantage of using ODBC as a bridge between .fp5 and .fp7 is that you can import directly from the FileMaker data source without first going through the intermediate step of exporting the data to an external file. This eliminates the need for maintaining a shared file location between the two as well as the additional scripts required to export the data.



Assuming that you have configured your .fp5 and .fp7 to be proper data sources, you can use the File>Import>ODBC Source... command to launch the ODBC import dialog box. Unlike standard imports between FileMaker Pro databases, you don't have to find the correct set of records in the remote FileMaker source before performing the import. You can issue a SQL statement to return the proper records for import entirely within the import file itself. Moreover, you can write data to an external database in an incremental fashion and with more granular control. With ODBC, it is possible to create a script to execute SQL statements to save data to another FileMaker Pro database. Moreover, the anticipated improvements to ODBC services in FileMaker Pro 8 promises to make this approach practical in a real world setting.

JDBC

JDBC stands for Java Database Connectivity. Like ODBC, it is an API that provides connectivity between many SQL databases. Unlike, ODBC, FileMaker Pro 6 doesn't let you import data from a "JDBC source" from within FileMaker Pro itself--there is no Import...>JDBC command. Another potential impediment is that JDBC connections to FileMaker Pro are made using the HTTP protocol via the Web Companion plug-in. You have to turn on sharing via Web Companion in FileMaker Pro 6.

FileMaker Pro 8 lets you import data via JDBC, but it can't export data via JDBC. To do that, you need FileMaker Server 8 Advanced. You can execute SQL statements in FileMaker Pro 8, but these are really ODBC SQL calls, not JDBC calls.

At this time, as far as bridging .fp5 and .fp7 databases is concerned, JDBC appears to be a useful "high-end" technology, if you want to write your own replication or synchronization engine.

XML interchange between .fp5-.fp7 via the web

When FileMaker introduced XML in version 5.5, it offered a compelling way of creating lightweight XML based web-services through the web-companion plug-in. The idea is that one could set up a .fp5 database using the web-companion plug-in to listen for instructions sent to it over the web (HTTP). FileMaker Pro could then process those instructions and return the results in nicely formatted XML. Thus, instead of saving and picking-up export files from some shared location, we could, in theory, script two different FileMaker Pro databases to import as well as write data to one another over the web.

FileMaker Pro 8 can query other web services and import XML data. To get XML data out of FileMaker Pro 8 databases, you'll need to use FileMaker Server 8 Advanced. That said, there are some benefits to this approach. Unlike ODBC, XML over HTTP is supported on both Mac OS X and Windows. As of this writing, ODBC connectivity support will only be available in FileMaker Server for Windows 2000 Server and Windows 2003 Server.

More importantly, with XML interchange over the web, it is also possible to establish smaller, but more frequent, exchanges between two different databases and to do so over the wide-area-network. You can write data directly to specific records and in a more incremental fashion. For example, it would be possible to write your



FileMaker Pro 8 application to selectively copy data to a remote FileMaker Pro 6 database when the user clicks an “update remote” button.

The actual mechanics of doing data interchange over XML/HTTP deserve much greater consideration than I am able to give in this article. There are also other considerations, such as record locking, size limitations of the HTTP request, HTTP GET vs. POST, and SSL, that aren’t covered here. Just because it is possible, doesn’t mean it is necessarily a good idea. This approach requires that you have sufficient understanding of querying various versions of FileMaker Pro databases over the web, a solid grasp of XML, the ability to apply XSLT where necessary, and the knowledge to manage web-based security as well as a clear understanding of the limitations of data privacy and authenticity in FileMaker Pro 6 or FileMaker Pro 8.

There are caveats to the import-export strategies above. For one, in all scenarios the contents of container fields are not supported. This can be overcome with the ExportFM plug-in from New Millennium www.newmillennium.com which supports scripted export and import of container field contents.

A bigger issue has to do with merging changes between databases that are being used simultaneously. In a one-way import scenario, the import-export strategies outlined above might be sufficient to share data between .fp5 and .fp7 databases. The problem gets trickier if these databases need to be synchronized. Let’s say you have a Students database in .fp5 and another in .fp7. Both co-exist peacefully in your school, but both databases are being used by different groups of people, and records in both databases can be modified independently of each other. If Abdul in The Department of Student Records updates a student, John Smith’s address in his .fp7 Students table but Homer in Accounts updates the phone number in the John Smith’s record in his .fp5 database, we now have a situation where changes from both might need to be merged. John’s Smith record in both Abdul’s .fp7 database and Homer’s fp5 database might need to have the latest phone number and address info. As such, the import/update approach alone may not be sufficient to reconcile these changes.

This is a classic synchronization issue with few easy solutions. The crux of the issue is that if the same record that is stored in two data stores has been modified separately we want to be able to reconcile the changes. There are two possibilities. You can write your own synchronization/reconciliation application for your specific database needs or you could adopt a third-party solution such as SyncDeK www.SyncDeK.com.

Replication with SyncDeK

SyncDeK is not a FileMaker Pro application. It is currently a Java-based solution that allows users to replicate data between different database nodes. It supports FileMaker Pro versions 5 through 8 and will work over nearly all standard network protocols such as SMTP (email) and HTTP (web). SyncDeK is a cross-platform set of databases and FileMaker Pro plug-ins that allows different copies of databases to be synchronized via automated email messages or the web, all from within a FileMaker Pro solution.

One of the most important features of SyncDeK is that it does not require an always-on Internet connection, allowing offices (known as “nodes” in SyncDeK parlance) to synchronize at their own convenience. Because the synchronization exchange is fully automated, the only interactions required from the user are initiation of the synchronization and reconciliation in the case of race conditions.



SyncDeK uses industry standard DES encryption, so data transferred over the Internet is secure from casual observers and other forms of data interception. SyncDeK provides a method of capturing not only record creation and edits, but also record deletion, and a method of propagating those changes to other databases. Unlike the traditional “update import” in FileMaker Pro, SyncDeK allows data to be merged together, capturing field-level changes and combining them in the record. While it will not prevent race conditions, it provides field-level reconciliation. This allows the individual node to decide whether or not to accept a particular edit if a race condition has occurred in order to avoid the data collision side effect.

SyncDeK introduces the concept of record ownership to each “node” of the database. For example, records created by Abdul who is using a FileMaker Pro 6 database are owned by Abdul. For such records, Abdul can decide to whom modifications are propagated and from whom and for which fields it will accept modification of those records. This gives the Abdul the ability to prevent accidental modification or deletion of key data while allowing other offices access to the records and the ability to change those records. The nodes that don’t own the record are free to modify and/or delete the record as they see fit, however those modifications will not necessarily propagate back to Abdul’s database or to other nodes.

Records can be created in any node, whether in FileMaker Pro 6 or FileMaker Pro 8. Those records are owned by their originating nodes. As the owner node, Abdul has the ability to determine whether those records are propagated out to other nodes, and in turn, what changes it will accept - if any. In the default configuration, records that are deleted by the owner node are deleted from other nodes. If a node deletes a record it does not own and propagates that deletion back to the owner, the owner has the option to accept, decline, or defer the decision. If the owner declines the deletion, any future modification to that record and ensuing synchronization will cause that record to be recreated at the node that initially deleted it.

The beauty of SyncDeK is that the actual mechanics of exporting and importing between FileMaker Pro 6 and FileMaker Pro 8 can be made transparent to the user. However, this interchange is not “live”. When a user modifies a record in one SyncDeK node, the same record in other copies is not locked automatically. In other words, inherent in a distributed database system is the possibility of a race condition. SyncDeK is not able to prevent a race condition.

However, it provides a way of dealing with it through its conflict resolution process. When a record is edited at multiple nodes and subsequently synchronized, SyncDeK will provide each node with the option to accept or decline the conflicting changes on a field-by-field basis. This process will allow the various copies of the database to remain out of sync however only at the explicit desire of each location.

Choosing the right strategy

Choosing the right bridge depends on several factors. There are no easy answers. You should first consider your overall database needs and decide whether it makes sense for you and your users to be using .fp5 and .fp7 databases at the same time. What are your overall goals? Do you have to maintain two FileMaker Pro platforms? Is it possible to convert all of your existing database solutions to .fp7? If you have to migrate incrementally, what is your migration time frame? What is your budget?



If you decide that bridging is necessary, it is worth thinking through your bridging strategies in the context of your overall migration plan. Which applications and databases are mission-critical? How are these databases connected to other systems? How tightly are they connected? Is it possible to rely on imports and exports to exchange data? Do you need these exchanges to be automated or is it sufficient for them to be relatively manual. What sort of technical expertise can you count on to maintain these bridges?

I offer these questions as starting points for discussion and planning. My personal opinion is that you should turn to bridging as a last resort. It may ultimately be necessary, but FileMaker Pro 8 promises to open a whole new dimension of database experience to users and developers alike and from this vantage point, the sooner we all cross the bridge, the better.

About the author

Ernest Koe is the executive VP and Chief Methodologist at inRESONANCE, Inc. a FileMaker Solutions Alliance Partner based in Northampton, Massachusetts. inRESONANCE is a strategy and technology consulting firm serving schools and non-profit organizations; iR provides customizable FileMaker Pro solutions, FileMaker Pro training as well as web-design and integration services to clients worldwide. See www.inresonance.com for more information about inRESONANCE.

Please Note: The following Appendix was created to capture FileMaker 7 conversion issues and has not been updated to reflect any minor FileMaker 8 differences. Please refer to the “Converting FileMaker Databases from Previous Versions” pdf document included on the CD with FileMaker Pro 8 or FileMaker Pro 8 Advanced for updates.

