SECURITY SCHEMA CHANGES IN FILEMAKER® PRO 11 AND FILEMAKER® PRO 11 ADVANCED

By:

Steven H. Blackwell
President & CEO, Management Counseling Services

Platinum Member, FileMaker Business Alliance
FileMaker Authorized Trainer
FileMaker 10 Certified Developer
FileMaker 9 Certified Developer
FileMaker 8 Certified Developer
FileMaker 7 Certified Developer

March 2010.

TABLE OF CONTENTS

—THANKS AND ACKNOWLEDGEMENTS—

A special thanks to FileMaker Business Alliance Members Barbara R. Levine, Anne Verrinder, and Wim Decorte for their review of versions of this manuscript and for their many helpful suggestions to improve the clarity of these descriptions and the understanding of these concepts.

The newly released versions of FileMaker Pro, FileMaker Pro Advanced, and FileMaker Server Advanced all have significant changes in their security features. The desktop client products have a new File Access Protection feature and new calculation functions related to Extended Privileges and Privilege Set Names. Additionally there is a change in the default Menu Commands option when developers create a new Privilege Set. FileMaker Server Advanced has some new features related to role-based administration of the FileMaker Server Advanced product; that is beyond the scope of this paper however.

This paper will examine the new security features in FileMaker® Pro 11 and FileMaker® Pro 11 Advanced, with specific emphasis on the File Access Protection enhancements.

Why were these changes needed? Why were they implemented? What are they? And, how do they work?

—INTRODUCTION—

Since the advent in March 2004 of FileMaker® Pro 7, the data and business logic in files have been at risk for external discovery and manipulation in ways neither authorized nor (often times) contemplated by developers of FileMaker solutions. Additionally since FileMaker® Pro 4, other internal information, including possibly sensitive data, was at risk of discovery in much the same unplanned and not-contemplated fashion. The core purpose of File Access Protection is to close these vulnerabilities.

An external file can be pointed at a second (usually hosted) data file[1] to which a user has some access rights, but not [Full Access] rights, or, in many instances, even *significant* access rights. This is accomplished by having the data file be an *external data source* in the external file. With such a configuration, ***and irrespective of the major privilege bit settings in the hosted data file***, data from that file can be printed, exported, manipulated, and extracted. Prior to the newly released version, the only really effective counter to this was extensive use of Record Level Access constraints. Even with that, a user authorized to access and to view data could perform actions on that data in the external file that the user could not perform from *within* the User Interface (UI) of the data file itself. This includes printing and exporting data; it also could include changing data in ways not contemplated by the UI or business logic of the original data file.

Additionally, FileMaker scripts in the hosted data file could be called in a variety of ways from an external file, either by a button action or by the *Perform Script* script step. Unless the script was marked as <<No Access>>, it could be called and performed in unexpected, and often times incomplete, ways. Developers have attempted to manage

[1] See the FileMaker Tech Info Note at http://thefmkb.com/5671 for additional details.

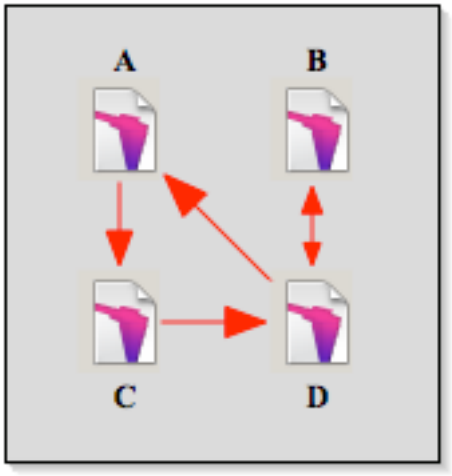
this vulnerability by use of "secret" parameters with varying degrees of success. In many instances the concept worked; however, in other instances, the parameter wasn't really "secret" and thus the intended process was thwarted.

Finally, even before the introduction of the .fp7 format, Design Functions could be used in unexpected ways to extract both data and metadata from hosted files. As an example, the *LayoutNames, ValueListNames, ScriptNames,* and *ValueListItems* functions are powerful tools; however they can have unexpected consequences. For example, if a Value List is defined to be the contents of a particular field, the *ValueListItems* function can extract the unique values for that specific field to an external unauthorized file. Similarly the *ScriptNames* function will return a list of all Scripts that are either *Executable Only* or *Modifiable* for the Account that opened the hosted data file. Further manipulation of those scripts can then occur.

—WHAT IS NEEDED—

What is needed to address these issues? How can developers assure that their FileMaker Pro solutions are not subject to access from unauthorized files? What we need here is a process wherein **all authorized files that are part of a specific solution have a *trusted relationship* each one with all the others and operate within a *security bubble* that prevents other files' becoming part of that trusted group.**

Thus, to establish a trust relationship between two files should require knowledge of a *[Full Access] set of credentials* in both files. *These need not necessarily be the same credentials; however, both sets of credentials should be for [Full Access].* This will help to assure that persons authorized to use those trusted files cannot manipulate them to access information or metadata in other trusted files in ways not intended by the developer.



Schematic 1. Trust relationships among files.

As an example, suppose we have four files called *A, B, C,* and *D* respectively, as shown in the schematic. Unidirectional or bidirectional trust relationships can exist between and among these files in a variety of fashions. *A* can trust *C*; *C* can trust *D*; and, *D* can trust *A*—all without *D* trusting *C* or *A* trusting *D*. Alternatively, *B* and *D* have a bidirectional trust relationship; each trusts the other.

As another example, in solutions utilizing The Separation Model™ the UI files would have a trust relationship with the data files. Data files might trust one another for purposes of the data model. But extraneous UI files would not be part of the trust; hence they would fall outside of the security bubble.

—How To Address This—

So, given all these items, how does the newly released FileMaker Pro version attempt to manage these issues? *Define Accounts & Privileges…* or *Manage Accounts & Privileges…* has been renamed ***Manage Security…*** as shown in Figures 1 and 1A below, and a new tab named **File Access** has been added as shown in Figures 2 and 2A below:

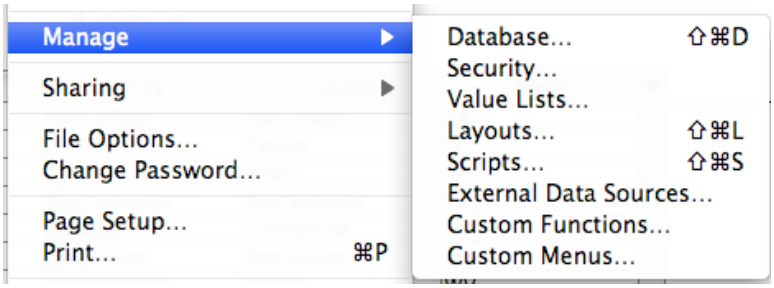


Figure 1. Manage Security… on Macintosh OS.

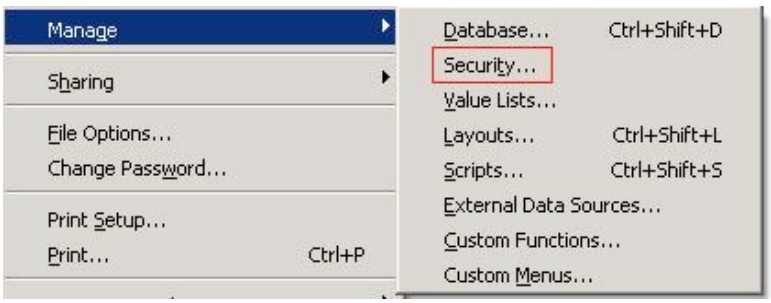


Figure 1A. Manage Security… on Windows OS.

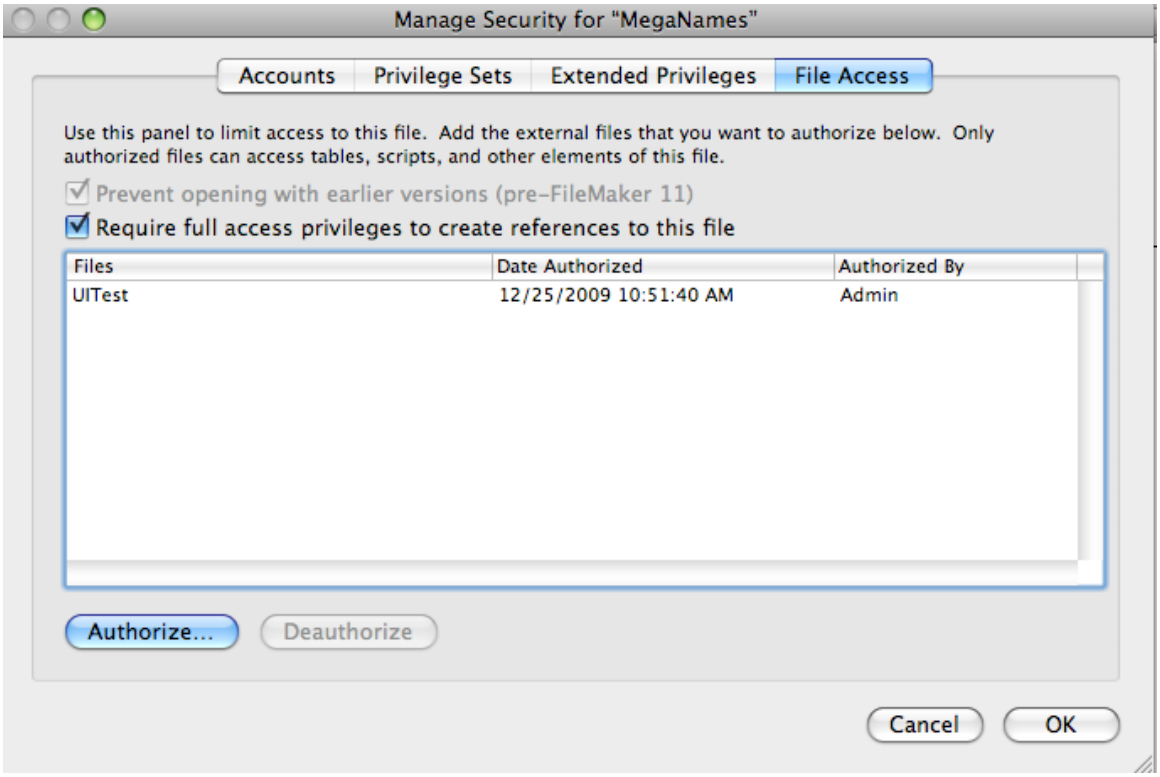


Figure 2. File Access Tab on Macintosh OS.

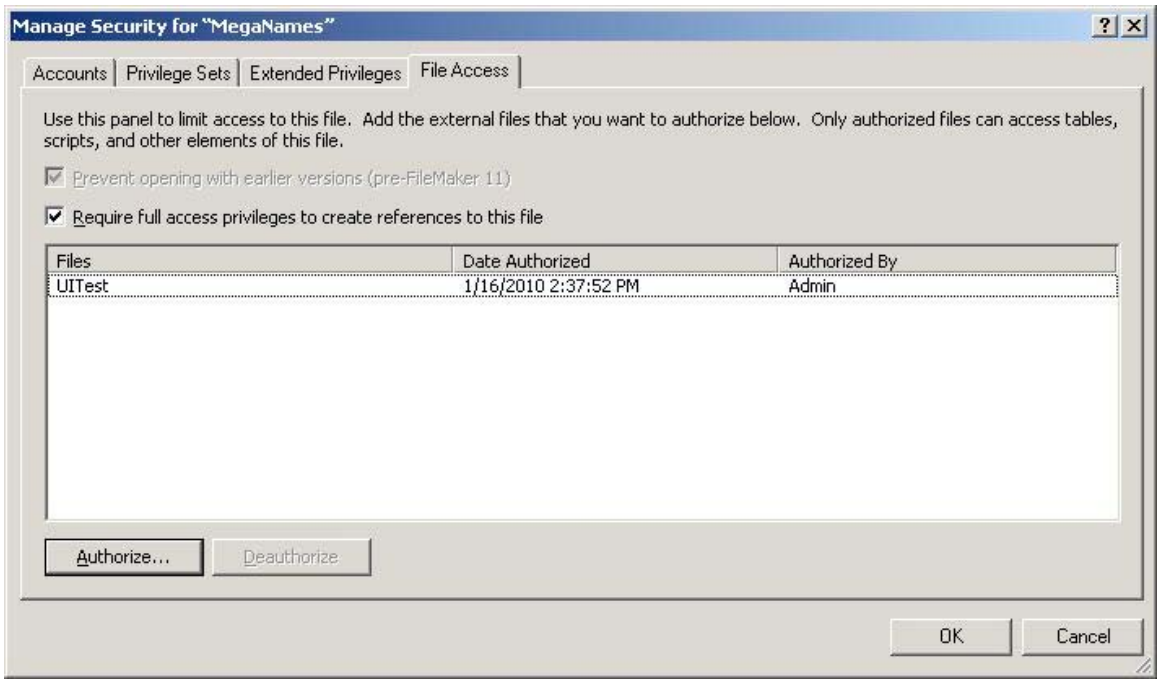


Figure 2A. File Access Tab on Windows OS.

For purposes of this White Paper's discussion, we have two FileMaker Pro files: a data file called ***MegaNames.fp7*** *hosted on FileMaker Server and a file called*

UITest.fp7 *that a user has created fresh on his or her workstation.* Developers now have the ***capability to require*** that anyone wishing to link to a protected file must provide [Full Access] credentials to that file (MegaNames.fp7) before any of the following actions can occur:

1. Placing a table alias to a table in *MegaNames* on the Relationship Graph in *UITest*;
2. Creating a Value List in *UITest* based on fields in *MegaNames* or using a Value List already created in *MegaNames*;
3. Calling a Script from *MegaNames* either by a Perform Script step or a button action in *UITest*; and,
4. Extracting information to an external file from *MegaNames* by use of Design Functions.

If the file attempting the access has been created by the person seeking access, as with UITest.fp7, then that person knows the [Full Access] credentials for the file seeking access. If, on the other hand, the person seeking access is using a file already created by someone else as the vehicle for gaining access to MegaNames.fp7, then that person must provide [Full Access] credentials *for both files before the trust relationship can be established.*

How does all this work? There are a number of moving parts here. What is required to implement these new functionalities? First, developers must take *two specific steps* to implement the new process. Refer again to Figures 2 and 2A. Note the two options:

Prevent opening with earlier versions (pre-FileMaker 11), and,

Require full access privileges to create references to this file.

Developers can enable the first option related to earlier FileMaker Pro versions without implementing the second related to references to the file. **However, to enable the *create reference* protection, the first option related to earlier FileMaker Pro versions must also be enabled.** This is the scenario shown in the two illustrations. A principal reason for this is to prevent thwarting the intent of the protection by opening the files in an earlier .fp7 format version of FileMaker Pro.

Here is an important note: *This will also mean that only FileMaker® Server 11 or FileMaker® Server 11 Advanced can host the file. Earlier Server versions will be unable to host such a file.* Earlier versions FileMaker Server will return an error message when they encounter a protected file.

—THE IMPACT OF THIS—

So, once enabled, what does this protection do? What benefits and functionalities does it offer? Consider the following scenario. The developer has placed the file

MegaNames.fp7 on FileMaker Server for hosting. A user has restricted privilege access to this file, but decides he or she wants to see what other information can be gleaned. So that user creates a new file on the desktop of the workstation called UITest.fp7 and points it by use of *Manage External Data Sources* at MegaNames.fp7. If MegaNames.fp7 has been protected, when the user attempts to place a Table Alias of a table in MegaNames.fp7 on the Graph in UITest.fp7, the user will be challenged as shown in Figures 3 and 3A.

Figure 3. Access Challenge for Protected File on Macintosh OS.

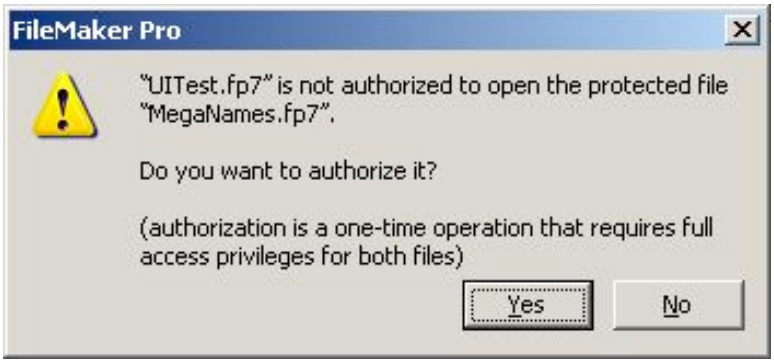


Figure 3A. Access Challenge for Protected File on Windows OS.

Further, examine the Graph in MegaNames.fp7 as shown in Figures 4 and 4A. Note the description on the Table Alias related to its protection status. File Access Protection is Enabled.

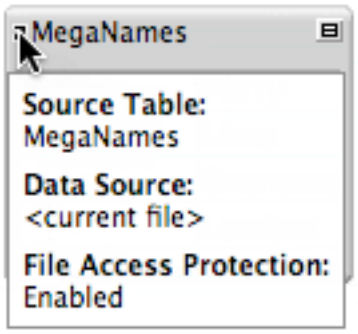


Figure 4. File Access Protection Enabled on Macintosh OS.

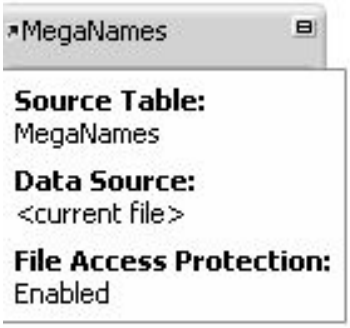


Figure 4A. File Access Protection Enabled on Windows OS.

In order to create Value Lists in UITest.fp7 based on either field values in MegaNames.fp7 or on existing Value Lists in MegaNames.fp7, users might go to Manage Value Lists… in the external file as shown, Figures 5 and 5A.

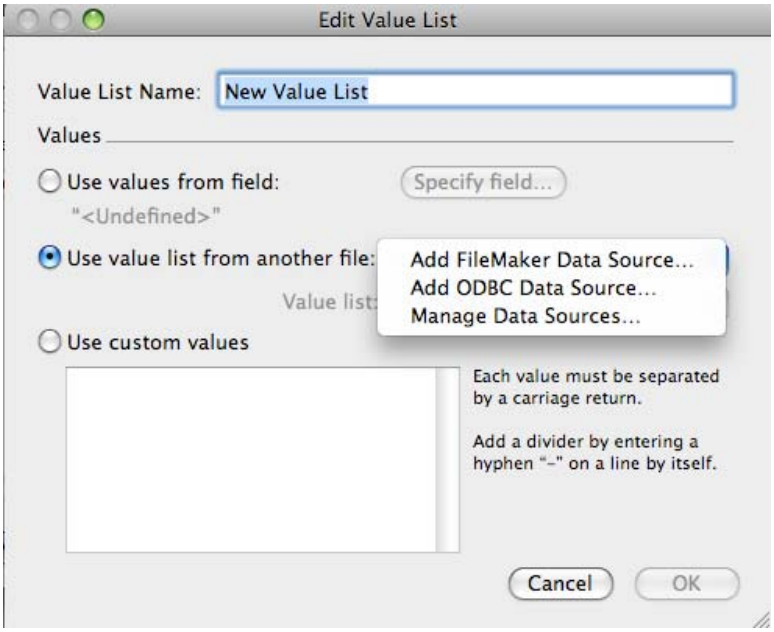


Figure 5. Add Value List on Macintosh OS.

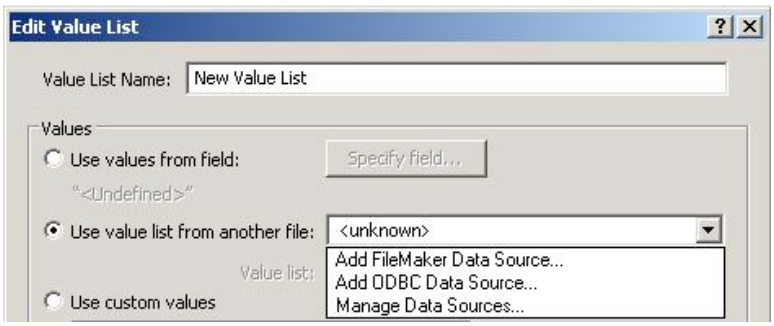


Figure 5A. Add Value List on Windows OS.

In each of these instances, users will be challenged as before for access to protected files. See Figures 6 and 6A. *Important Note: If any item, such as a Script or Value List, is set to be <<No Access>> for the Account that opened MegaNames.fp7,* ***that item will not appear even if the external file is authorized to access MegaNames.fp7.***

Figure 6. Credentials Challenge on Macintosh OS.

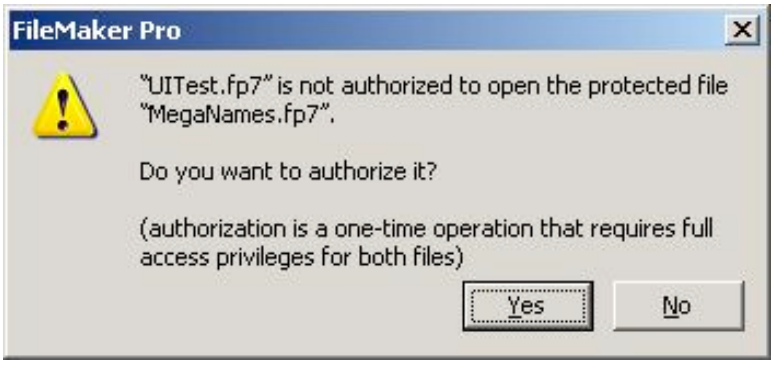


Figure 6A. Credentials Challenge on Windows OS.

The same process applies to scripts in MegaNames.fp7 that a user might wish to run from UITest.fp7. as noted in Figures 7 and 7A.

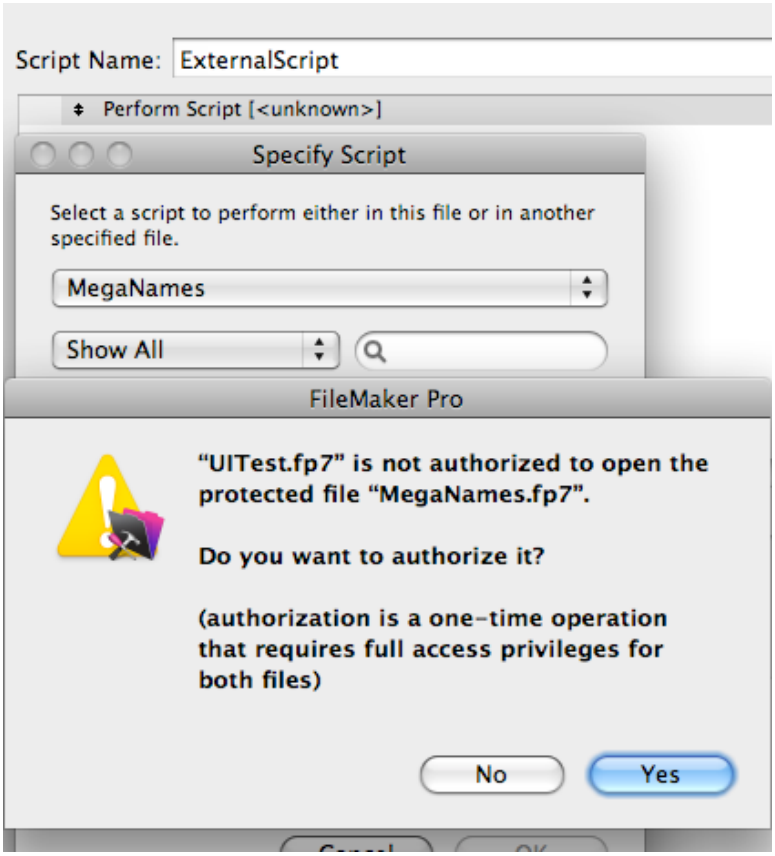


Figure 7. External Scripts Challenged on Macintosh OS.

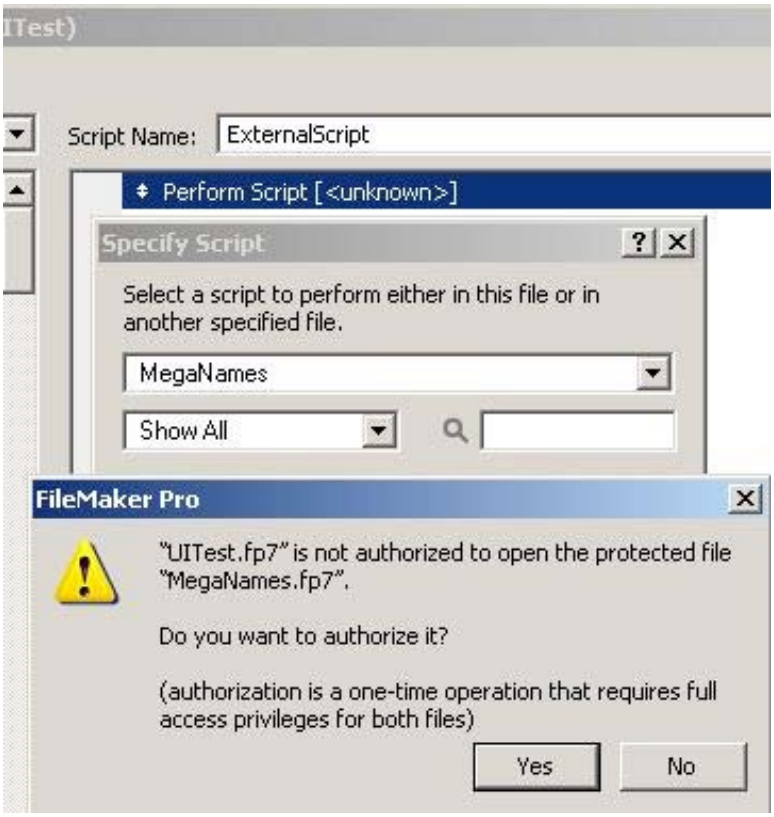


Figure 7A. External Scripts Challenged on Windows OS.

—DEAUTHORIZATION—

Developers may ***deauthorize*** an external file that was previously authorized if they desire. Refer again to Figures 2 and 2A. Note the ***Deauthorize*** button in the User Interface. By selecting a file's name in the list and clicking the *Deauthorize* button, that file's access permission is removed. The next time the previously authorized external file is opened and seeks access challenges will occur as shown in Figures 8 and 8A.

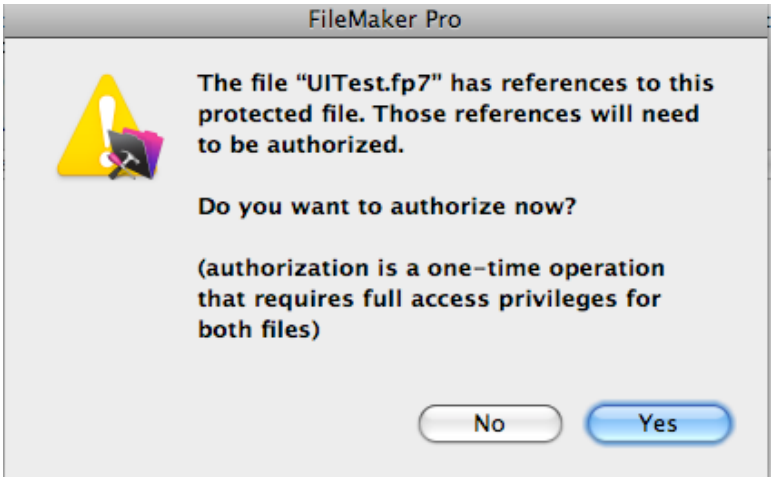


Figure 8. Authorization Challenge on Macintosh OS.

Figure 8A. Authorization Challenge on Windows OS.

—INTERACTION WITH DESIGN FUNCTIONS—

As part of File Access Protection, Design Functions will no longer return values ***from*** a file ***to*** an external file unless the external file is authorized to access the original file. FileMaker Pro returns no specific error; however the functionality is disabled. To restore the ability of the external file to receive Design Function information, a developer must establish the authorized trust relationship described previously.

As a result of this new protection unauthorized external files cannot extract data and metadata about other files through the Design Functions.

All these are the functions affected by this behavior except for *DatabaseNames*, because it does not reach into the specific file:

DatabaseNames	ScriptIDs
FieldBounds	ScriptNames
FieldComment	TableIDs
FieldIDs	TableNames
FieldNames	ValueListIDs
FieldRepetitions	ValueListItems
FieldStyle	ValueListNames
FieldType	WindowNames
GetNextSerialValue	ScriptIDs
LayoutIDs	ScriptNames
LayoutNames	TableIDs
LayoutObjectNames	TableNames
RelationInfo	ValueListIDs

Another important note: Design Functions still operate normally when called from within the file where they are targeted. Design Functions will work ***inside*** MegaNames.fp7 irrespective of File Access Protection settings.

—NEW CALCULATION FUNCTIONS—

Also new in FileMaker Pro 11 and FileMaker Pro 11 Advanced are functions designed to return information about the privileges associated with the Account that has opened the file. If files have been created in earlier versions of the .fp7 family, then the names of some functions have changed. These changes will be evident when the file is opened in version 11.

1. **Get(PrivilegeSetName)** converts to **Get(CurrentPrivilegeSetName)** and functions as in earlier versions.

2. **Get(ExtendedPrivileges)** converts to **Get(CurrentExtendedPrivileges)** and functions as in as in earlier versions.

3. **Get (AccountPrivilegeSetName)**, new in version 11, returns the actual Privilege Set of the Account that opened the file *even if* in a script that is set to *Run script with full access privileges*. This functionality would therefore have to be added to any converted files or it can be selected in newly built files in version 11. This obviates the need for passing a parameter into the script with the information associated with the Active Account.

4. **Get (AccountExtendedPrivileges)**, new in version 11, returns the Extended Privileges of the Account that actually opened the file *even if* in a script that is

set to *Run script with full access privileges*. This functionality would therefore have to be added to any converted files or it can be selected in newly built files in version 11. This too obviates the need for passing a parameter into the script with the information associated with the Active Account.

5. **Get (AccountPrivilegeSetName)** and **Get (AccountExtendedPrivileges),** both newly created in version 11, when opened in earlier versions, return ***<FunctionMissing>.***

—DEFAULT MENU COMMANDS OPTIONS—

In earlier versions (7 through 10) when developer created a new Privilege Set, the default setting for the Menu Commands Privilege Bit was *All.* This was inconsistent with behavior of other Privilege Bits that all follow the ***Rule Of Least Privilege*** concept, wherein a specific privilege is *disabled by default* (or is least permissive) and must be enabled for the particular role defined by the Privilege Set. In FileMaker Pro 11 and FileMaker Pro 11 Advanced, this has been corrected. Now the default Menu Commands option will be *Minimum* as shown below.

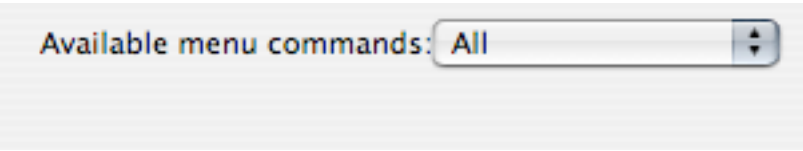


*Figure 9. **Old** default setting on Macintosh OS.*

*Figure 9A. **Old** default setting on Windows OS.*

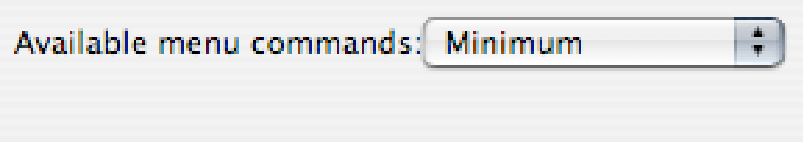


*Figure 9B. **New** default setting on Macintosh OS.*

*Figure 9C. **New** default setting on Windows OS.*

Here are a couple of important caveats about this feature. First, the two FileMaker-created default subordinate Privilege Sets, [Data Entry Only] and [Read-Only

Access] still use the *All* Menu Commands option. This is another reason developers should avoid using these defaults. These default Privilege Sets also have surprising high levels of privileges despite their names. Second, when creating a new Privilege Set with the (now new) default Menu Commands setting of *Minimum*, a developer can still change that to *All* or to *Editing only* if required.

—SUMMARY—

1. We are establishing a *trust relationship* between two files that allows access *into* one file *from* another file.
2. To establish such a trust relationship requires knowledge of [Full Access] credentials for both files in the trust relationship.
3. All the files that share one or more trusted relationships run in a *security bubble* that prevents external unauthorized files from establishing table aliases to them, prevents value list manipulation through the FileMaker Pro UI, and prevents extraction of data and metadata by use of the Design Functions.
4. When we create a new Privilege Set, the default setting for the Menu Commands option is now *Minimum*, not *All* as was previously the case.
5. Several new calculation functions allow for better testing of the current Privilege Set conditions attached to the Account that opened the file.