Copyright © 2010 New Millennium Communications, Inc.

SecureFM 11
Plug-In for FileMaker Pro

User Guide

SecureFM's core security functions let you entirely remove FileMaker menus or selectively disable menu commands, keyboard shortcuts, toolbars, and checkboxes. Additional functions give you the ability to rename menus and menu items, and override normal FileMaker menu functions with your own scripts. You can also automatically run a script whenever the front file window changes or when a user enters data into a field.

With the security functions, you can prevent inadvertent data loss or corruption by selectively disabling individual FileMaker menu commands such as Delete Record, Delete All Records, Replace, Close, or Quit, while keeping other essential commands available to the user. With the additional menu functions, you can create highly customized menus for your solutions.

This is the best solution available to protect your data and customize FileMaker Pro's menus!

Overview of SecureFM

New Millennium Communications first created SecureFM 1.0 for FileMaker Pro 3.0 in 1997, originally as a system extension on the Macintosh platform. We released a radically enhanced and more flexible version in 1999, SecureFM 3.0 for FileMaker Pro 4.x, as a cross-platform FileMaker plug-in, and added the "Configurator" (now called the Menu Editor). We have maintained it since then to be compatible with each new release of FileMaker Pro and we have added features to improve the degree of security offered.

The "MenuMagic" features are by far the greatest enhancement that we have made to the plug-in since 1999. The features that are now included are part of the original vision that we had for SecureFM, but it has taken years to get this technology ready for market.

SecureFM gives you all of the following capabilities:

1. Disable individual menu items and their keyboard equivalents
2. Remove entire menus
3. Hide the toolbars
4. Hide the status bar (that useless little bar at the bottom on Windows)
5. Assign a password to your registration so that only the developer can change these menu definitions
6. Create a default menu definition to disable selected menu items as soon as FileMaker is launched, before a file is open
7. Rename menus and menu items
8. Call your own scripts from menu items instead of the FileMaker function
9. Automatic changing of menu item names and their actions when the active table occurrence changes (such as when a new window or layout comes to the front... No script required!)
10. Call your own scripts from FileMaker's keyboard shortcuts, contextual menus, and toolbar icons!
11. Assign custom hot keys which will call your scripts, using any modifier keys (or no modifiers)
12. Trigger scripts from a calculation, for instance when tabbing from a field
13. Trigger scripts by the event of a window coming to the front (regardless of how it is brought to the front)
14. Disable, rename, and script menus whenever the file, table occurrence, layout, or window changes.

15. Use variable file, table occurrence, layout, and window names when defining menus. For example, you can apply a menu configuration any time the active table occurrence name begins with a word like “Contacts.”
16. Completely remove the entire FileMaker menubar on both Windows and OS X!

You can now create custom controlled user-interfaces in FileMaker Pro with dynamic menus linked to specific file, table occurrence, layout, and/or window names, without having to control navigation by script. Think about that for a minute...

> "New Record" can be "New Customer" or "New Widget’ and run your custom script.
> "Delete Record" can become "Delete Customer" or "Delete Widget", and can be properly controlled.
> "Show All Records" can be "Show My Records"
> Control-F can run your custom find script
> The toolbar icons and contextual menus can call your custom scripts

AND (trumpet sounds...):

> To change files or layouts, you don’t need to have the user click a button to change your menu definitions. As soon as the file, table occurrence, layout, or window changes, the menu definitions change automatically!

Ease of Deployment with the “Menu Table”

You can easily create your custom menus for each file with the “Menu Editor”, and you can easily deploy your custom menus with the “Menu Table”.

The SecureFM Menu Table file, which is provided with the SecureFM demo files, has no dependencies on other files. You can rename it and put it into your own solution. It is designed to be called whenever you need it and then closed. Most commonly, it is opened during a startup script; selected “menu definitions” are applied (stored in the plug-in’s memory) by looping through a found set of records; and then the file is closed.

The Menu Table file is open so you can customize it however you wish.

Contents

Installing SecureFM

Installation on Macintosh:
Place the SecureFM plug-in into the Extensions folder inside your FileMaker Pro folder.

Installation on Windows:
Place the SecureFM.fmx plug-in into the Extensions folder inside your FileMaker Pro folder.

Remove old versions: Do not forget to remove any older versions of **SecureFM** that were previously installed. The old plug-in may not be automatically replaced. Having more than one version of **SecureFM** can cause problems.

Restart FileMaker after installing the plug-in.

Make sure SecureFM is selected in the Plug-Ins section of the Application Preferences (found in the Edit Menu).

Installation Note: The plug-ins for Windows and Macintosh OS X are different. Please make sure you are using the correct version for your platform.

Registering SecureFM

Registering SecureFM in the Demo File

To register SecureFM enter your user name and license key string into the Registration String field provided in the registration area of the SecureFM Menu Editor (included with demo). If you have not yet downloaded the current version of the plug-in and associated files from our website, please do so.

http://www.newmillennium.com

Once you have entered your user name and license key, click the Register button on SecureFM Menu Editor's Register layout. You will know that the plug-in has registered properly if the Registration Pass|Fail field contains the word: "Registered."

Registering SecureFM in Your Solutions

You can register SecureFM in your own FileMaker Pro solutions by using the external "Menu-Register" function in a script. The registration script step can be incorporated into any script in your solution, as long as it is called before you intend the menus to be modified. Therefore it is commonly put into an "On Open" script that is automatically run every time your solution’s main file is launched.

You can use a simple Set Field script step to register SecureFM. In the example below we are performing the registration Set Field step into a text field named SFM Response. The registration syntax is:

External ("Menu-Register", "My Company|MACCODE|WINDOWSCODE|Optional Password")

The code for the Macintosh platform must appear after the first pipe separator. The code for the Windows platform must appear after the second pipe separator. Even if you are only using the registration code for a single platform, you must still include both pipe separators in the parameter - but only the code for the current platform is checked.

Please note that the user name and license key are case-sensitive.

For a more complete discussion of how to incorporate the registration process into your solutions, see the Menu-Register function discussion of the SecureFM Documentation.

Important Registration Note:

Registering SecureFM once does not mean it is permanently registered! It is only registered until FileMaker is shut down. **Each time FileMaker is launched SecureFM needs to be re-registered.** This function enables you to test in your solution to make sure that SecureFM is installed.

SecureFM expires after 60 minutes if not registered. After that point, FileMaker menus are no longer disabled, and the user is warned that an unregistered version of SecureFM is in use.

Menu-Register

External ("Menu-Register", parameter)

Menu_Register

Menu_Register (parameter)

The parameter must include the user name and the registration code for either the Mac or Windows platform (or both). You can also add an optional password for additional security (more about this function later). Each of the pieces of information are separated by a pipe "|" character:

"My Company|MACCODE|WINDOWSCODE|Optional Password"

The code for the Macintosh platform must appear after the first pipe separator. The code for the Windows platform must appear after the second pipe separator. Even if you are only using the registration code for a single platform, you must still include both pipe separators in the parameter - but only the code for the current platform is checked.

Please note that the user name and license key are case-sensitive.

SecureFM expires after 60 minutes if not registered. After that point, FileMaker menus are no longer disabled, and the user is warned that an unregistered version of SecureFM is in use.

Important Registration Note:

Registering SecureFM once does not mean it is permanently registered! It is only registered until FileMaker is shut down. **Each time FileMaker is launched SecureFM needs to be re-registered.** Calling the Menu-Register function when your solution launches enables you to test to make sure that SecureFM is installed.

You can register SecureFM in two ways:

1) You can use the "Menu-Register" function every time your solution is opened, such as in an On Open script, or

2) You can use the "Menu-SetStartup" function to store the registration as part of FileMaker's default configuration. To do this, you must first register SecureFM (as well as make sure any other "Menu-Disable" settings you want are also active) and then call the "Menu-SetStartup" function. The registration code as well as the default

Menu-Disable setting will be stored in the "SecureFM.ini" file, and SecureFM will automatically re-register itself every time FileMaker is started up.

Registering with the "Menu-Register" function

You can register SecureFM by calling the external "Menu-Register" function, usually from a script, though it can also be done from a calculation field or within a validation by calculation. You can include a registration script step in any script in your solution, as long as it is called before you intend the menus to be modified. Therefore it is commonly put into an "On Open" script.

The syntax in the script is:

```
External ("Menu-Register", "My Company|MACCODE|WINDOWSCODE|Optional
Password")
```

To set a script as the file's On Open script, go into FileMaker's File Options (in the File menu). Make sure "Perform Script" is selected under 'When Opening "FILENAME"' in the Open/Close tab, and select your On Open script in the pull-down menu to the right.

If you have a multiple file solution, you must make sure during the opening process that at least one of your files will open and properly register SecureFM in this manner.

Auto Registering Using SetStartup

This method of registering SecureFM is helpful if you want the menus to be modified as soon as FileMaker is launched, but you can't be sure which file will be opened first. It can also be essential if you don't want to allow access to modifying a script in a file. It is not as secure as putting the Register step into a script, however.

To use this method, register SecureFM once via script using the correct Licensee Name and Registration Codes. (This can be done from the SecureFM Menu Editor file.)

Create a default SecureFM preference file using SecureFM's "Menu-SetStartup" command. (Again, this can be done using the SecureFM Menu Editor file. See the section below regarding SecureFM's "Menu-SetStartup" for more details.)

In the new preference file ("SecureFM.ini"), SecureFM will store all registration information (as well as the default Menu-Disable string) and will re-register itself automatically when FileMaker is started up. The preference file has the suffix ".ini" and must be placed in the FileMaker Extensions file for Mac or FileMaker's System directory for Windows. Place this file into the appropriate folder, along with SecureFM itself, on each computer.

If your Registration String contains both the Mac and Windows Registration Codes, the preference file will be cross-platform and can be copied to either type of platform across a network, as long as your Menu-Disable string has no differences between platforms (see "Menu-SetStartup" below for more details).

Detecting if SecureFM is Installed

The "Menu-Register" call can be used to detect whether SecureFM is installed, as it always returns one of the errors or confirmations below:

- "Registered SecureFM" & the version number,
- "Invalid Registration SecureFM" & the version number,
- "Expired Registration SecureFM" & the version number,
- "Incorrect Password SecureFM" & version number, or
- Question Mark ("?") if the plug-in is not installed.

Question Mark vs. Blank Registration Response

In past versions of FileMaker, if a plug-in was not loaded and active or if an unknown External function was called, no response was returned to the response field. In FileMaker 7 and later, if an unknown External command is called -- because the plug-in is missing, not active, or even if the function name is misspelled -- FileMaker now returns "?" to the response field.

If you try to register SecureFM with an empty string, it returns the registration status, without altering the current registration. This can be helpful for checking whether SecureFM has already been registered.

Thus, if you have already registered SecureFM in the same FileMaker session, and then send the command:

```
External ("Menu-Register", "")
```

SecureFM will NOT alter the registration status. Instead, it will return the current registration status.

If, however, you include any data in the registration string, SecureFM will evaluate the new registration string.

Checking for Plug-In Version

See the documentation on the Menu-Version function for a discussion of how to easily check the plug-in’s version number.

Important Registration Security Note

Any invalid registration string will void the current registration UNLESS the original registration uses a password. When registered with a password, SecureFM will not upgrade, downgrade, or unregister if an invalid registration string is sent or if a registration string without the correct password is sent. (See below for information or registering with a password.)

If security is an important part of your SecureFM implementation, then you should always register with a password!

Password Registration

SecureFM's Password is an advanced feature that gives you an added layer of security, allowing you to ensure that only the solution creator can control FileMaker's menus and toolbars.

Using a password is OPTIONAL (but strongly recommended for security). This makes SecureFM backward compatible with plug-in commands designed to run on earlier versions of SecureFM. It also means you can simply choose not to add a password to your solution if you feel that this advanced level of security isn't necessary for your solution.

Password for Added Security

Registering SecureFM with a password prevents users from re-configuring the menus unless they know the password. When SecureFM is registered with a password, that password must also be used when setting all menu configurations and toolbar access.

If you don't use a password, a FileMaker-skilled user who has script access to any FileMaker file and understands the syntax of SecureFM could enable all menu items. Even a novice user who has access to the SecureFM Menu Editor file (which can be downloaded by anyone from New Millennium's web site) can simply hit the "Restore All Menus" button. Registering SecureFM with a password prevents these possibilities.

Ex: Registering With a Password

Registering with a password uses the following format, with the optional password as the fourth part of the parameter:

```
External ("Menu-Register", "My Company|MACCODE|WINDOWSCODE|Optional
Password")
```

The password parameter can be up to 100 characters long.

Using SecureFM with a Password

After SecureFM has been registered with a password, all subsequent commands which have a password parameter (Menu-Disable, Menu-Rename, Menu-Script, Menu-Toolbar, etc.) will require the password as well; otherwise, an "Incorrect Password" error will be returned.

This requirement also applies to the Menu-Register function. Once SecureFM is registered with a password, you can only re-register using the same password. This prevents a user with script access from removing or changing the password. Once set, the only way to remove or change the password is by restarting FileMaker.

Password Protection

In our SecureFM demo files, we store the password in a field because that is the simplest way to demonstrate how the function works. However, storing sensitive data like passwords and license codes in FileMaker fields is not very secure. For the greatest security, this sensitive data should be manually entered into the script steps themselves, and not held in fields.

Menu-Version

External ("Menu-Version", “”)

Menu_Version

Menu_Version (parameter)

The Menu-Version function returns just the version number of SecureFM without any other text. This makes it easy to determine the version of SecureFM currently in use.

External ("Menu-Version", "")

No parameter is used.

The response returns only SecureFM's version number, such as:

11.0.2

Version # in aabbccdd Format

To get the plug-in version number in the format aabbccdd (eg. version SecureFM 11.0.2 returns 11000200):

Menu_Version (“number”)

[This capability is only available with Menu_Version. It is not available for the old style function External (“Menu-Version” ; ...).]

Menu-Disable

External ("Menu-Disable", parameter)

Menu_Disable

Menu_Disable (parameter)

This is the primary SecureFM command. It allows you to specify which FileMaker menu items (or entire menus) you want to disable. Menu items can be disabled for all files by using a “default” configuration or different menu items can be disabled for different files and layouts by using a “table occurrence-specific” configuration.

Disabling Menus and Menu Items

SecureFM recognizes specific menus and specific items in those menus, not by name, but by number. Menus are counted left to right, from File (1) to Help (9). (The Arrange menu, which is available only in Layout mode, is considered to be menu number 10.) The individual menu items are counted top to bottom.

A simple disable string such as--

<3,B,<3>>

--is read as--

1 - FileMaker menu affected. Counting left to right, the View menu is the third FileMaker menu.

B - Mode affected. In this case, we are affecting the Browse mode. You can specify one of the following five options for this middle clause:

B = Browse mode
F = Find mode
L = Layout mode
P = Preview mode
A = All modes (see notes below)

3 - Menu item(s) to be disabled. In this case, the third item from the top in the View menu is the "Layout Mode" command.

Each Menu-Disable string must be preceded by a prefix that tells SecureFM when to apply the menu configuration.

Applying to All Files

You can apply a menu configuration globally to all files by using an asterisk by using the default context prefix of “<*>:”

```
<*>:
<3,B,<3>>
```

By using a default menu definition, you can enforce security globally on new files.

Context-Specific Prefix Syntax

You can also specify Disable configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The new format used in the string’s prefix is:

```
<file/table/layout/window>:
```

Context-specific prefixes add significant power and flexibility to your control of FileMaker’s menus!

(For a complete explanation of how to work with Context-Specific Prefixes, see the section on String Prefixes and Different Types of Menu Definitions.)

If you want to change the previous example to only apply to a file named “Widgets”, you string would become:

```
<Widgets///>:
<3,B,<3>>
```

If you want the string to apply only to layouts named “Entry”, you would use:

```
<//Entry/>:
<3,B,<3>>
```

Default vs. Specific Precedence

If you have both a default context and specific context string that applies, the specific string will apply.

Menu Item Numbers and Separators

Be aware that the SEPARATOR LINES appearing in the menu pull-downs ARE COUNTED as menu items. Thus, if you want to disable the "Cut" command in the Edit menu, you must disable item THREE, not item two, because the separator is item two:

```
<*>:
<2,B,<3>>
```

A Simple Example - Disabling a Single Item

The parameter (or "configuration string") is enclosed in <> ("greater than" and "less than" brackets). Here is a very simple example:

```
<*>:
<1,B,<2>>
```

This configuration string tells SecureFM to disable the "Open..." command in FileMaker's File menu when in Browse mode.

Disabling Keyboard Commands

When you disable a menu item that can be called by keyboard command -- for example "Close" and Command-W on Mac or Ctrl-W on Windows -- SecureFM will also automatically disable the associated keyboard command(s). Go ahead and try:

```
<*>:
<1,B,<7>>
```

Another Example - Disabling Multiple Items

If you want to disable more than one item within a menu, simply separate the item numbers by commas. For example:

```
<*>:
<1,B,<1,2,3,4,5,6,7>>
```

SecureFM interprets this as: In the File menu (the first menu from the left), when in Browse mode, disable "New Database...", “New From Starter Solution”, "Open...", “Open Remote…”, “Open Recent”, “Open Favorite”, and "Close".

Meaningless Items in Strings

There are several menu items which FileMaker does not enable in certain modes (they are always gray). You can configure your string to disable them with SecureFM, but it will have no additional effect since they aren't selectable in the first place. You can even configure your string to disable separators, but it will also have no effect.

A More Complex Example - Disabling Multiple Menus

If you want to disable items in more than one menu, simply create a different configuration string for each menu. (You may want to separate the string for each menu with a paragraph return to make the string easier to read and edit, though this is not required). For example:

```
<*>:
<1,B,<1,2,3>>
<7,B,<1>>
```

With the above string, SecureFM will disable the first three items of the File menu and also the first item of the Script menu ("ScriptMaker...").

You can add as many items to the string as necessary:

```
<*>:
<1,B,<1,2,3>>
<3,B,<3>>
<6,B,<1,2,3,5>>
<7,B,<1>>
```

Free Form String Formatting

You can freely add spaces, tabs, and returns to your strings for the purpose of readability.

Ex: Disabling for More than One Mode

Use the same format above to disable menu items in more than one mode. For example, suppose you want to disable the first three items in the File menu for both Browse and Layout modes, your string will look like this:

```
<*>:
<1,B,<1,2,3>>
<1,L,<1,2,3>>
```

Also note, the items disabled do NOT have to be identical in different modes. You can, for example, disable all three of the above items in Browse mode, but only disable the "New Database..." command in Layout mode. Your string would then look like this:

```
<*>:
<1,B,<1,2,3>>
<1,L,<1>>
```

Ex: Disabling the Same Item in ALL Modes

If you want to disable the same item or items in ALL four modes, instead of four lines in the string (one for each mode), you can use the “A” clause instead. For example:

```
<*>:
<7,A,<1>>
```

This string will disable the "ScriptMaker..." command in the Script menu in all four modes. It has exactly the same effect as:

```
<*>:
<7,B,<1>>
<7,F,<1>>
<7,L,<1>>
<7,P,<1>>
```

Important Note About the "A" Clause:

Be cautious when using the disable in all modes "A" command. SecureFM will disable that item number in all modes, EVEN IF THE ITEM IS DIFFERENT IN ANOTHER MODE. For example, in the Records/Requests/Layouts menu, item number 7 is "Show All Records" in Browse and Preview, but it is "Perform Find" in Find mode and it is a divider in Layout mode. Using the string—

```
<*>:
<6,A,<7>>
```

—will disable ALL of those menu items!

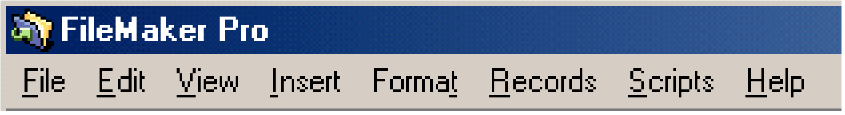
Ex: Disabling an Entire Menu

If you want to disable all items in a menu, rather than individually listing each item number, you can use a zero ("<0>") in the items brackets. For example, suppose you want to completely disable the Window menu (in all modes) to prevent the user from bypassing your solution's navigation buttons. Your string would look like this:

```
<*>:
<8,A,<0>>
```

The Window menu will still pull down but all items will be grayed out.

Ex: Removing an Entire Menu



If you want to completely remove the menu from FileMaker's menu bar, you can use a negative one ("<-1>") in the items brackets. (Note: this function is not available on OS X.) For example--

```
<*>:
<8,A,<-1>>
```

--will cause the Windows menu to disappear from FileMaker's menu bar, until you choose to re-enable it.

To remove the Window menu in all modes except Layout mode, use:

```
<*>:
<8,B,<-1>>
<8,F,<-1>>
<8,P,<-1>>
```

This can be an especially useful configuration, particularly when combined with an On Open or Startup string which disables the Layout Mode menu item, and a hidden button which re-enables Layout Mode. This enables a developer to work on a solution at a user’s desk without having to quit and relaunch with a different password.

Individual Menus Can Only be Disabled, Not Removed on Mac OS X

Attempting to remove a menu on OS X will result in the menu being disabled instead. You can still use the remove command, for example "`<8,B,<-1>>`", but the menu will be grayed out rather than removed, as if you were using the string "`<8,B,<0>>`".

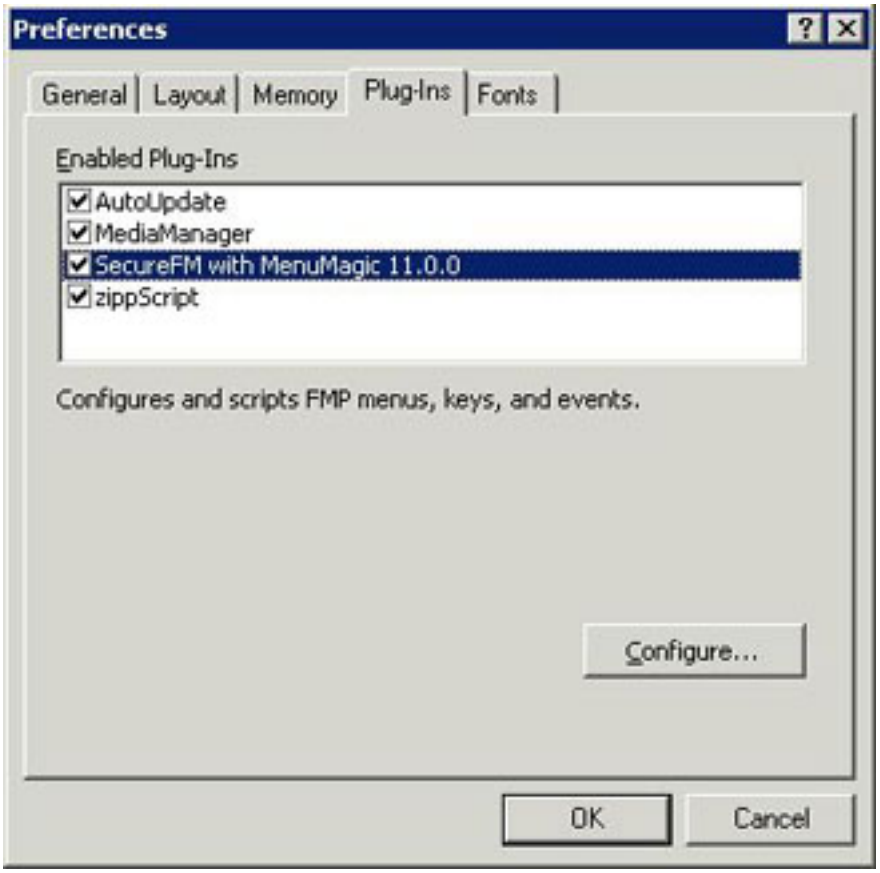
You can, however, use the Menu-Menubar command to completely remove the FileMaker Pro menu bar, if you wish. (See the documentation on Menu-Menubar for more information.)

Controlling Specific Functions

Very Important: Disabling FileMaker's Preferences Command

FileMaker gives users a very simple way to disable a plug-in: In the Edit menu (or the “FileMaker Pro” Application menu in OS X), select Preferences: Application: Plug-Ins; then uncheck whichever plug-ins you want. If a user does this, the plug-in immediately ceases to function. **If this is done with SecureFM, all disabled FileMaker functions are IMMEDIATELY RE-ENABLED!**

You can prevent a user from doing this by disabling Preferences in the Edit menu.



Enabled Plug-Ins

☑ AutoUpdate
☑ MediaManager
☑ SecureFM with MenuMagic 11.0.0
☑ zippScript

Configures and scripts FMP menus, keys, and events.

The cross-platform string to do this in FileMaker:

```
Case (
Abs (Get (SystemPlatform)) = 1,
"<*>:
<-1,A,<3>>",

Abs (Get (SystemPlatform)) = 2,
"<*>:
<2,B,<19>>
<2,F,<19>>
<2,L,<18>>
<2,P,<19>>")
```

There may be times, however, when you, as a developer, or other system managers will need access to the Preferences menu. You can still access Application Preferences or Document Preferences via script. So, for example, you can have an Open Application Preferences script that is conditional based on User Name, or create a button that is available on a management-only layout.

You may want to disable the Preferences menu item in all but Layout mode (and, of course, make certain that Layout mode isn't available to most users).

An additional protection is to use the “SFMDisable” Menu-Event function available with a MenuMagic license, which can be used to automatically quit FileMaker Pro if SecureFM is disabled. See the Menu-Event documentation for more information.

Disabling Exit / Quit

The Exit / Quit command can be disabled by using menu item number "40" in the File menu. If you want to disable Exit / Quit in Browse mode on Windows, you can use:

```
<*>:
<1,B,<40>>
```

This format for disabling Exit / Quit minimizes confusion over the numerical location of Exit / Quit within the File menu. With the potential for different positions of the Exit/Quit command in different versions of FileMaker, position number "40" has been arbitrarily selected to specify the last menu item, no matter what its actual position number.

Disabling Window Closeboxes

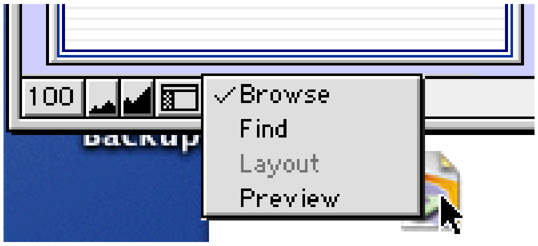
SecureFM will automatically disable the file window closebox when you disable the File: Close command:

```
<*>:
<1,A,<7>>
```

The application window closebox on Windows is, in effect a Quit button, so SecureFM disables it when the Exit / Quit command is disabled in the File menu.

Disabling the Mode Footer Pop-Up Menu

When you disable access to any of the four modes (Browse, Find, Layout, Preview) in the View menu, access is also disabled in the mode pop-up menu that appears in FileMaker's footer bar.



Important: Hide and Lock Status Bar in FileMaker 10 & 11

Beginning with FileMaker 10, the old Status Area has been redesigned, moving it from the left to the top, and renaming it the Status Bar. In addition to reconceived mode buttons, this is also now a customizable toolbar, allowing users to choose which buttons to display in the Status Bar for each file. For many uses, this new design is very useful, but it also creates potentially serious security loopholes, giving users easy access to functionality like Manage Database, Delete Record, etc. SecureFM does not control the new Status Bar. **If you are creating a secure solution in FileMaker 10 or 11, in addition to using SecureFM to disable or repurpose potentially dangerous menu items, you must also hide and lock the Status Bar using the Show/Hide Status Area script step – in every file you want to protect.**

Controlling the Minimize Window Command

Disabling Minimize Window item in the Window menu ("<8,A,<4>>") disables the Minimize Window menu item, its associated keyboard command (Cmd-M or Ctrl-M). On OS X, the yellow Minimize button in the FileMaker title bar is also disabled, as well as the ability to minimize the window by double-clicking on the title bar.

Disabling Manage Scripts (ScriptMaker)

If you want to disable the Manage Scripts menu item (formerly called ScriptMaker), remember that the menu item appears in two different locations in the newer versions of FileMaker, and both menu items must be disabled:

In the Scripts menu, disable Manage Scripts ("<7,A,<1>>"), but it also appears as a sub-menu item under Manage in the File menu ("<1,A,<7>>"). To use SecureFM to prevent access to scripts, you must disable both menu items.

Disabling the Restore Window Widget

You can disable the Restore Window widget (the green "gumdrop" on OS X) by disabling the first divider in FileMaker Pro's Window menu ("<8,A,<5>>").

Locking the File Window Position

To lock the file window position (its upper left corner), disable Move in the File System menu ("<0,A,<2>>").

Although the Mac platform does not have a File System menu, use the same string ("<0,A,<2>>") to lock the window position. When locked, users can still drag the window aside, but when the mouse button is released, the file window will return to its original position.

Locking the File Window Size

To lock the file window size, disable Resize in the File System menu ("<0,A,<3>>").

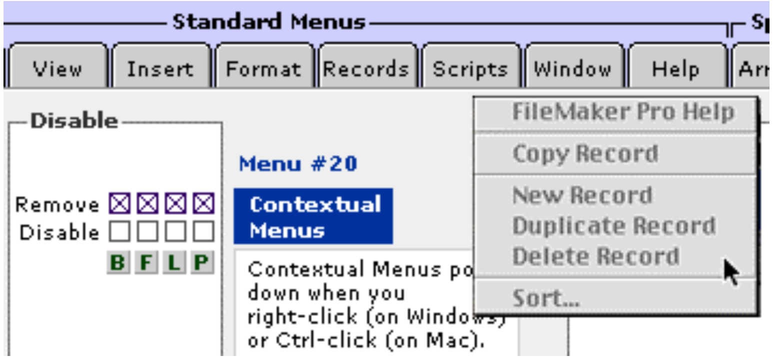
The Mac platform does not have a File System menu, but you can still use the same string ("<0,A,<3>>") to lock the window size.

Help Menu Cannot be Disabled on Mac

Attempts to modify the Help menu – with Menu-Disable, Menu-Rename, or Menu-Script – will have no effect on the Help menu. No error is returned if you attempt to modify the Help menu but no changes will occur.

Disabling Contextual Menus

Contextual Menus are the pop-up menus that appear in FileMaker when you do Ctrl-Mouse Click (on the Mac) or Right Mouse Click (on Windows).



Even though these menus may not be widely used by the average user, through them FileMaker Pro offers access to powerful FileMaker commands, such as Delete Record and Duplicate Record. EVEN IF YOU HAVE DISABLED THESE COMMANDS IN FILEMAKER'S MENU BAR, THEY ARE STILL AVAILABLE TO THE USER IN THE CONTEXTUAL MENUS! (SecureFM Menu Editor will, however, warn you when you have checked a menu item to be disabled but have not disabled the appropriate Contextual Menu.) For complete security, you must also disable the Contextual Menus.

Contextual Menus have been assigned a menu number of twenty ("<20>"), and SecureFM only allows you to disable ("<0>") or remove ("<-1>") them; you cannot disable individual items in the Contextual Menus.

To be certain you haven't left any backdoors in your SecureFM configuration, it is usually a good idea to include the following in your configuration string:

<20,B,<-1>>
<20,F,<-1>>
<20,P,<-1>>

Disabling New Database in FileMaker's Initial Open Dialog

If you want to prevent users from being able to create new FileMaker databases under all circumstances, in addition to disabling the “New Database...” command in the File menu, you must also disable the ability to create new databases in FileMaker's initial open dialog.

This can be done using SecureFM’s Menu-SetStartup command. See the documents on Menu-SetStartup for a discussion on how to do this.

Disabling Copy All Records

The Copy All Records keyboard command on Macintosh, “Cmd-Opt-C” is disabled when either Copy (“<2,A,<4>>”) or Export Records (“<1,A,<19>>”) is disabled.

On OS X, “Copy All Records” appears in place of “Copy” in the Edit menu when the Option key is held down. This too is grayed out when either Copy or Export Records is disabled. If Export Records is disabled and not Copy, then the Copy command remains available in edit. When the option key is hit, Copy becomes Copy All Records but is grayed out.

Platform Menu Differences

Platform Differences Between FileMaker Menus

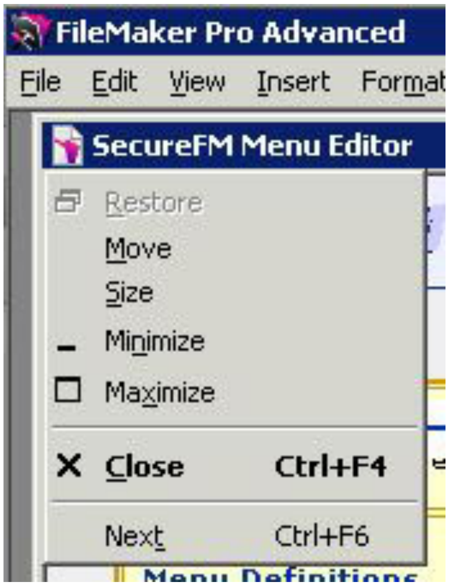
If you plan on using SecureFM for a cross-platform solution, there are a few key differences you must be aware of between FileMaker's menus for the various platforms.

This is essential to keep in mind when creating SecureFM configuration strings since SecureFM disables menu items by position number, not by name. The same menu item may have a different menu position on different platforms.

The SecureFM Menu Editor file makes these differences easy to recognize. For example, the File menu is identical on both Mac OS X and Windows except that on Mac OS X, the Quit command has been completely removed and placed in the "FileMaker Pro" Application menu.

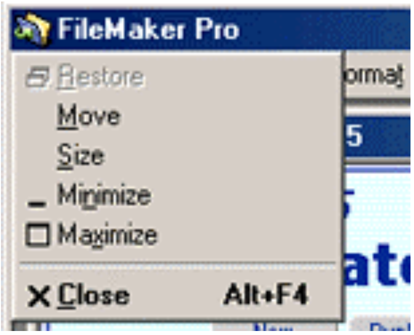
File System and Application System Menus on Windows

SecureFM allows you to disable two additional menus that appear only on the Windows platform: the File System menu and the Application System menu.



The File System menu appears when you click on the FileMaker file icon to the far left of the file window title bar.

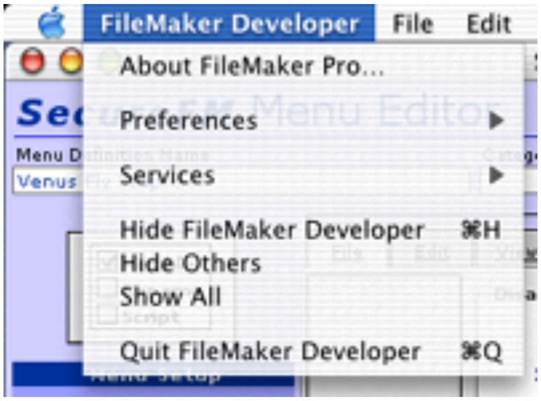
File System menu <0>



The Application System menu appears when you click on the FileMaker application logo to the far left of the application window title bar.

Application System menu <-1>

“FileMaker Pro” Application Menu on Mac OS X



In FileMaker for Mac OS X, there is a “FileMaker Pro” Application menu that appears immediately to the left of the standard File menu. The two most important items in the Application menu to be aware of for security purposes are Quit and Preferences (which appear in the File and Edit menus, respectively, on other Windows). The menu number corresponds with the Windows Application System menu:

Application menu <-1>

FileMaker 11: On Mac, any time your string attempts to disable any individual menu item in the “FileMaker Pro” Application menu, the entire menu will be disabled. This change in behavior only applies to FileMaker 11 on the Mac.

Dynamic Menus on OS X

Holding down the Option key when a menu is pulled down alters certain menu items. The dynamic menu item is generally related to the standard menu item. For example, Paste becomes Paste Text Only. One dynamic menu item change to be aware of is that Copy becomes Copy All Records (see Controlling Copy All Records elsewhere in the Menu-Disable documentation).

When SecureFM disables a particular menu item, it also disables its associated dynamic menu item and keyboard command.

Creating Cross-Platform Strings with Get (SystemPlatform)

Using the FileMaker function Get (SystemPlatform) allows you to check which platform is currently running FileMaker Pro, allowing you to specify which variation of the configuration string you want to activate.

Get (SystemPlatform) -- called Status (CurrentPlatform) in FMP 6 -- returns the following results for each specific platform:

Windows 95, 98, and ME= 2
Windows 7, Vista, XP, 2000, NT = -2
Mac Classic = 1
Mac OS X = -1

You can use Get (SystemPlatform) to create CONDITIONAL configuration strings.

If you need to be aware of each platform in your configuration string, the following conditional statement will work:

```
Abs (Get (SystemPlatform))  = 1,
MAC OS X STRING,

Abs (Get (SystemPlatform)) = 2,
WINDOWS STRING)
```

Note that the use of the absolute value function ("Abs") will return 2 for either Get (SystemPlatform) = 2 or -2.

This technique can be used for any discrepancy in the menus between Mac and Windows versions of FileMaker Pro. You can also use this technique if you want different configurations based on platform.

Let's take a look at some examples of how this works.

Ex: Two Platform Conditional Strings

As a simple example, the Windows Edit menu has two more items ("Paste Special..." and "Object") that are not found in the Macintosh Edit menu. This causes the same item to have a different number depending on the platform. For example, the "Clear" command is Edit item number 7 on the Mac OS X, but number 8 on Windows.

If you use the string,

```
"<*>:
<2,B,<7>>"
```

SecureFM will disable "Clear" on Mac solutions, but will disable "Paste Special..." if you try to use that solution on a Windows machine!

The solution to this problem is to create a conditional configuration string. To tell SecureFM to disable the "Clear" command correctly for both Macintosh and Windows, your string will need to look like this:

```
Case (Abs (Get (SystemPlatform)) = 1,
"<*>:<2,B,<7>>",
"<*>:<2,B,<8>>")
```

SecureFM will then disable the Edit menu's item number 7 if on Mac OS X (Abs (Get (SystemPlatform)) = 1); otherwise, on Windows, it will disable item number 8.

SecureFM Menu Editor Automatically Generates Cross-Platform Configurations

The ‘Menu Editor’ within the SecureFM Menu Editor file automatically generates cross-platform configuration strings. For most uses, the strings created in the Menu Editor can be used without further editing or even a clear understanding of what the string means.

For further exploration of how to construct cross-platform strings, see the fully-commented ‘EXAMPLES’ scripts within the SecureFM Menu Editor file.

Creating Strings for Different Platforms with the Menu Editor

When creating a new menu configuration in the SecureFM Menu Editor file’s ‘Menu Editor,’ the 'Calculate String for' preference defaults to 'Selected Version of FMP.' When you click the 'Create String' button, the Menu Editor will calculate the string for only the selected version of FileMaker Pro. To change this preference, click on the ‘View Strings’ button to go to the Strings Palette. You can then change the radio button to 'All FMP Versions'.

To see the configuration string for a different version of FileMaker Pro, select the version you want to see in the FMP Version # popup menu in the upper left of the Menu Editor layout, or Strings Palette layout.

For further exploration of how to branch your scripts based on the version of FileMaker in use, see the fully-commented 'EXAMPLES' scripts within the SecureFM Menu Editor file.

Errors

Errors in the String

When you call the Menu-Disable command in a Set Field script step, SecureFM returns an empty string if successful, or an error message if the parameter is incorrect in some way.

Error Position

When SecureFM detects a syntax error in a function parameter, the error response attempts to locate the point at which the error occurs by listing the character position where the unrecognized code begins. This can help when debugging complex strings.

Configuring Menus with a Password

If you have registered SecureFM with a password (see Menu-Register), the same password is required to change the menu configuration or remove toolbars. To set a menu configuration with a password, you must add an additional pipe character "|" and the password to the end of the Menu-Disable string.

Ex: Password in a Simple String

For example, if your original Menu-Disable configuration was:

```
<*>:
<7,A,<-1>>
```

The string would become:

```
<*>:
<7,A,<-1>>|Password"
```

Ex: Password in a Multi-Line String

This format works the same when the string is multiple lines:

```
"<*>:
<7,B,<-1>>
<7,F,<-1>>
```

```
<7,P,<-1>>|Password"
```

Ex: Password in a Cross-Platform String

If you are using a string that branches based on platform, it is important to include the password at the end of each platform's string:

```
Case (Abs (Get (SystemPlatform)) = 1,
"<*>:
<-1,A,<3>>|Password",

Abs (Get (SystemPlatform)) = 2,
"<*>:
<2,B,<14>>
<2,F,<14>>
<2,P,<14>>
<2,L,<15>>|Password")
```

Password Error

If the password is missing or wrong in the Menu-Disable string, SecureFM returns "Incorrect password" and does not change the currently active configuration.

Menu-Menubar

External ("Menu-Menubar", parameter)

Menu_Menubar

Menu_Menubar (parameter)

The Menu-Menubar function gives you the ability to instantly hide FileMaker Pro's entire menu bar, disable all command keys, and disable all widget buttons in the window title bars (such as close window, maximize window, etc). This function also lets you prevent the file window from being moved or resized on Windows.

This is a simple function that only uses two parameters, marked 1 for on or 0 for off:

1|1 = Normal state

0|1 = Remove FMP menu bar. The entire menubar is hidden. No contextual or application popup menus will show (but user-defined field popup menus will still work). All FMP and OS keyboard commands are disabled; only Menu-HotKeys will work.

1|0 = Disable all window widget buttons: Close Window, Maximize Window, Minimize Window, Restore Window. (On the Windows platform, the Application System and File System menus are disabled with this parameter.) This parameter also prevents the file window from being moved or resized.

0|0 = Remove FMP menu bar and disable widget buttons.

When SecureFM has been registered with a password, this function will also require the use of that same password (using the format "0|0|password").

Warning: Be Careful Not to Get Trapped!

Because this function globally removes all FMP menus while disabling all keyboard commands, you must be very careful not to get trapped! Once the menu bar is removed, you cannot open ScriptMaker or call any scripts, except by button -- even to restore the menu bar. If you also disable window widget buttons, you cannot quit FileMaker Pro or even close file windows unless you have scripted buttons on an accessible layout to perform those functions or restore access to the menus and widgets. When testing, always have a button on the layout that restores the menubar to the normal state.

You must also be careful not to close all files or minimize all file windows with the menu bar removed.

Example - Remove the Menu Bar and Disable Window Buttons

The command --

```
External ("Menu-Menubar", "0|0")
```

-- will remove the entire menu bar, disable all keyboard shortcuts associated with FileMaker menu commands, and disable all buttons in the window title bar, including close, minimize, and restore.

Example - Restore the Menu Bar and Window Buttons

The command --

```
External ("Menu-Menubar", "1|1")
```

-- will restore the menu bar, keyboard shortcuts, and the window title bar buttons.

Configuring with a Password

If you have registered SecureFM with a password (see Menu-Register), the same password is required to use this function, as well. To use Menu-Menubar with a password, you must add an additional pipe character "|" and the password to the end of the Menu-Menubar string.

Example - Password in a Simple String

For example, if your original Menu-Menubar string was:

```
External ("Menu-Moolbar", "0|0")
```

It would become:

```
External ("Menu-Menubar", "0|0|Password")
```

Password Error

If the password is missing or wrong in the Menu-Menubar string, SecureFM returns "Incorrect password" and does not modify the menu bar or window buttons.

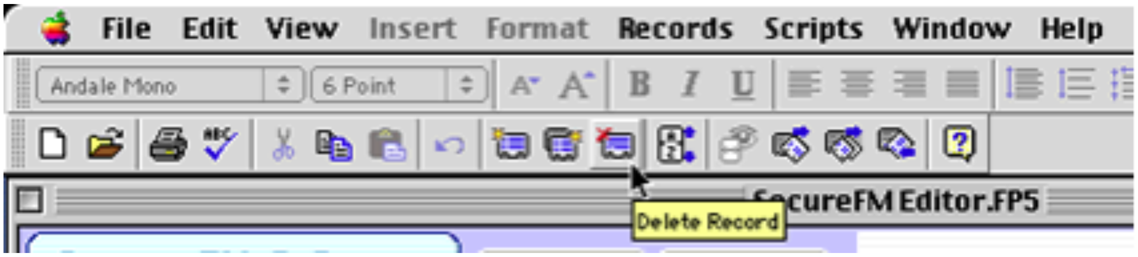
Menu-Toolbar

External ("Menu-Toolbar", parameter)

[This function only applies to older versions of FileMaker that used individual toolbars, prior to the FileMaker statusbar.]

SecureFM is able to easily remove FileMaker Pro’s toolbars: the Standard Toolbar (which includes such commands as New Record and Delete Record) and the Text Formatting Toolbar (which includes font sizes, text alignment, etc.).

Disabling the toolbars, especially the Standard Toolbar, can be very important to create a secure environment for your FileMaker solution. Even if you disable a function like Delete Record in FileMaker's Records menu, IT IS STILL AVAILABLE IN THE TOOLBAR! You must ALSO disable the Standard Toolbar.



To have a truly secure environment, you may need to disable FileMaker's toolbars. SecureFM gives you that ability.

Ex: Removing Both Toolbars

The following command:

External ("Menu-Toolbar", "0|0")

disables both the Standard Toolbar and the Text Formatting Toolbar in ALL modes.

Toolbar Command's Syntax

The zero on the left tells SecureFM to remove the Standard Toolbar. The zero on the right removes the Text Formatting Toolbar. Restated:

Left of pipe divider - Standard Toolbar
Right of pipe divider - Text Formatting Toolbar

There are three possible commands you can send to SecureFM for each of the toolbars:

0 - Remove toolbar
1 - Display toolbar
(blank) - No change to toolbar status

Note that the pipe character divider ("|") must always be used in the parameter, even if only one of the toolbars is being modified.

Ex: Removing Just One Toolbar

If you just want to remove the Standard Toolbar and leave the Text Formatting Toolbar alone, use:

```
External ("Menu-Toolbar", "0|")
```

Remember that the pipe character divider ("|") must be used, even though only one of the toolbars is being modified.

Why Have a 'No Change' Option?

Suppose you want to remove only the Standard Toolbar, but it doesn't matter to you whether the Text Formatting Toolbar is visible or not. If you use the command:

```
External ("Menu-Toolbar", "0|0")
```

the Text Formatting Toolbar will also be removed. And if you use the command:

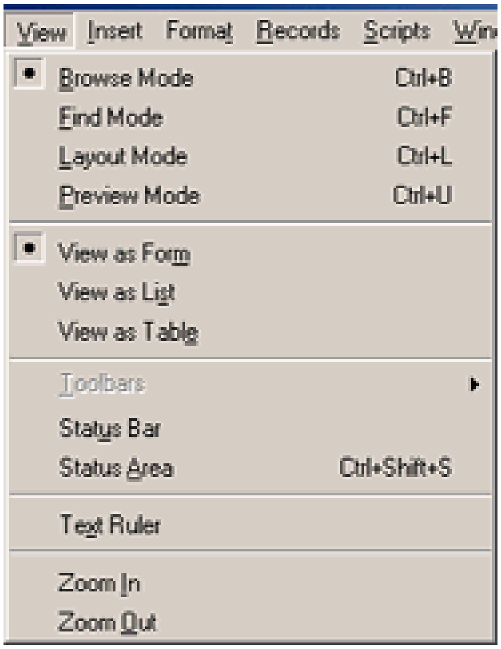
```
External ("Menu-Toolbar", "0|1")
```

the Text Formatting Toolbar will automatically appear. By leaving the Text Formatting Toolbar's command blank:

```
External ("Menu-Toolbar", "0|")
```

the Text Formatting Toolbar is left however it was previously set by the user or by a previous Menu-Toolbar command.

Important: Disabling the Toolbars Menu Item



It is very important that you also disable the Toolbars menu item in the View menu!

```
"<3,B,<10>>"
```

If you don't, users will be able to manually re-enable the toolbars at any time. (SecureFM Menu Editor will warn you if you have checked one or both of the toolbars to be disabled but have not disabled the View: Toolbars menu item as part of your configuration string.)

Toolbar Command and FileMaker's Modes

Removing (or displaying) toolbars with the SecureFM Menu-Toolbar command affects ALL modes (Browse, Find, Layout and Preview). You cannot control the toolbars in individual modes.

FileMaker's Other Toolbars

SecureFM only gives you direct control of the Standard Toolbar and Text Formatting Toolbar. FileMaker's other toolbars (Arrange and Tools) are available only in Layout mode and are not controlled by SecureFM. You can, however, restrict access to those other toolbars by disabling the Toolbars item in the View menu for Layout mode.

Re-Enabling Toolbars

The "Restore All Menus" button in the Menu Editor file does NOT cause disabled toolbars to reappear. It will, however, re-enable the Toolbars item in the View menu, which allows you to manually activate the toolbars. In order to make both toolbars automatically reappear, you must use the following command:

```
External ("Menu-Toolbar", "1|1")
```

Configuring Toolbars with a Password

If you have registered SecureFM with a password (see Menu-Register), the same password is required to change the menu configuration or toolbars. To configure toolbars with a password, you must add an additional pipe character "|" and the password to the end of the Menu-Toolbar string.

Ex: Password in a Simple String

For example, if your original Menu-Toolbar string was:

`External ("Menu-Toolbar", "0|0")`

It would become:

`External ("Menu-Toolbar", "0|0|Password")`

Ex: Configuring Without a Password

If SecureFM was registered without a password, the password part of the parameter and the last pipe character must be omitted:

`External ("Menu-Toolbar", "0|0")`

Important!: In SecureFM versions 5.06 to 6.0x, the pipe character could be included even when no password was present – hence this was valid:

`External ("Menu-Toolbar", "0|0|")`

In SecureFM 6.1 and later, this is no longer valid! You cannot include the last pipe character unless a password is included.

Password Error

If the password is missing or wrong in the Menu-Toolbar string, SecureFM returns "Incorrect password" and does not change the currently active toolbar configuration.

Menu-Statusbar

External ("Menu-Statusbar", parameter)

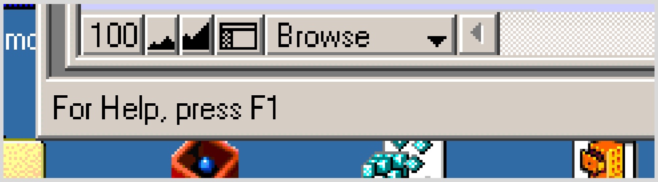
Menu_Statusbar

Menu_Statusbar (parameter)

The Menu-Statusbar function allows you to remove the Status Bar at the bottom of the FileMaker window on the Windows platform. This gives you better control of the application window's size and appearance from within your scripts.

The Status Bar appears only on the Windows platform. There is no parallel to the Status Bar on the Macintosh.

Ex: Removing the Status Bar

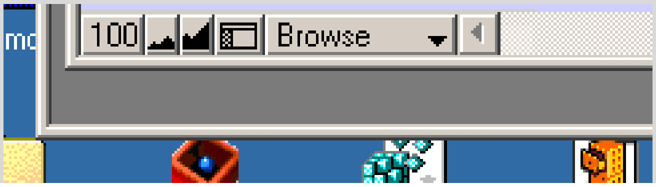


For Help, press F1

The following command--

External ("Menu-Statusbar", "0")

--removes the Status Bar in ALL modes.



Statusbar Command's Syntax

There are two possible commands you can send to SecureFM for the Status Bar:

0 - Remove Status Bar
1 - Display Status Bar

Disabling the Status Bar Menu Item

It is important that you also disable the Status Bar menu item in the View menu in your Windows platform configurations.

```
If[Abs(Get(SystemPlatform)) = 2,
External(“Menu-Disable”,
    "<3><B><11>"
    "<3><F><11>"
    "<3><P><11>"
    "<3><L><17>"
End If
```

If you don't, users will be able to manually re-enable the Status Bar at any time. (SecureFM Menu Editor will warn you if you have marked the Status Bar to be disabled but have not disabled the View: Status Bar menu item as part of your configuration string.)

Statusbar Command and FileMaker's Modes

Removing (or displaying) the Status Bar with the SecureFM Menu-Statusbar command affects ALL modes (Browse, Find, Layout and Preview). You cannot control the Status Bar in individual modes.

Re-Enabling the Status Bar

The "Restore All Menus" button does NOT cause the disabled Status Bar to reappear. It will, however, re-enable the Status Bar item in the View menu, which allows you to manually activate the Status Bar. In order to make the Status Bar automatically reappear, you must use the following command:

```
External ("Menu-Statusbar", "1")
```

Statusbar and Set Startup

The Menu-SetStartup function will not save the Status Bar's position. If you want the Status Bar to be either displayed or hidden when your solution is launched, you must call the Menu-Statusbar command in a startup script.

Configuring the Status Bar with a Password

If you have registered SecureFM with a password (see Menu-Register), the same password is required to use the Statusbar command. To configure the Status Bar with a password, you must add an additional pipe character "|" and the password to the end of the Menu-Statusbar string.

Ex: Password in a Simple String

For example, if your original Menu-Statusbar string was:

`External ("Menu-Statusbar", "0")`

It would become:

`External ("Menu-Statusbar", "0|Password")`

Ex: Configuring Without a Password

If SecureFM was registered without a password, the password part of the parameter and the last pipe character must be omitted:

`External ("Menu-Statusbar", "0|")`

is not valid. You must use:

`External ("Menu-Statusbar", "0")`

Password Error

If the password is missing or wrong in the Menu-Statusbar string, SecureFM returns "Incorrect password" and does not change the Status Bar.

Menu-SetStartup

External ("Menu-SetStartup", "")

Menu_SetStartup

Menu_SetStartup ("")

Understanding this command is necessary if you want a SecureFM menu configuration to apply to FileMaker on startup. This method of initializing SecureFM is useful if you want to have a default configuration of SecureFM which modifies the menus as soon as FileMaker is launched, regardless of which file is opened.

Because the file created by the SetStartup command is modifiable by a knowledgeable user, this method of configuring FileMaker's menus should only be used to control the initial launch process. It is always a good rule of thumb to configure FileMaker's menu's in your solution's On Open script, as well.

SetStartup Parameter is Not Used

Note that, although FileMaker's format for External commands assumes that there will be a parameter string, the parameter is not used by SecureFM for this function. The startup configuration is set based on the SecureFM Menu-Disable and Menu-Toolbar settings that are active when the SetStartup command is called.

Rather than look to a parameter in the External "Menu-SetStartup" command to set the default configuration, SecureFM sets the default based on:

1) The currently ACTIVE menu configuration (set using the "Menu-Disable" command),
2) The currently ACTIVE toolbar configuration (set using the "Menu-Toolbar" command), and
3) The currently ACTIVE registration (set using the "Menu-Register" command).

(Menu-Rename and Menu-Script configurations are not stored by Menu-SetStartup.)

Any of the above configurations you want set into the default must first be made active BEFORE using the "Menu-SetStartup" command.

The syntax for this function is always:

External ("Menu-SetStartup", "")

The parameter should always be empty.

Passwords and SetStartup

While encoded registration information, as well as Menu-Disable and Menu-Toolbar settings, are stored in the default preferences file ("SecureFM.ini"), no passwords are saved. This means that if you are relying on the default preferences to register SecureFM, it will do so with a blank password.

If you want SecureFM to be encoded with a password, you must re-register with the password (in a script, field calculation, etc.) after startup. For more details, see the section on Password Registration.

Using SetStartup in the SecureFM Menu Editor

In order to create a default configuration from the SecureFM Menu Editor file, you must create the SecureFM configuration string you want, activate it by clicking the 'Apply Now' button, and then click the Set Startup button.

SecureFM then creates a text file called "SecureFM.ini" that stores the default configuration data. This file is saved in the FileMaker Extensions folder (Mac) or FileMaker's System directory (Windows). When FileMaker is started, the default Menu-Disable settings will immediately take effect without the need to run any script or even have a file be open.

The SecureFM.ini file can be copied directly to workstations, making it very easy to configure a default configuration throughout a network.

SecureFM.ini for Different Platforms

The configuration string stored by SecureFM in the default preference file ("SecureFM.ini") is platform-specific. Even if the string you activate is conditional based on platform (see above), the configuration stored in the SecureFM.ini file will only contain information for the platform you are using.

If you are setting the default preference for a cross-platform solution -- and if the string is different for the two platforms -- you must create two separate SecureFM.ini files (one for Mac, one for Windows).

If the string is different for the two platforms, be aware that you must call the SetStartup function to create the SecureFM.ini file for each platform on the CORRECT platform. That is, you must create a Windows SecureFM.ini file on a Windows machine, and a Mac SecureFM.ini file on a Mac.

Disabling New Database Creation in FMP’s Initial Open Dialog

If you want to prevent users from being able to create new FileMaker databases under all circumstances, in addition to disabling the New Database... command in the File menu, you must also disable the ability to create new databases in FileMaker's initial open dialog.

This 'initial open dialog' occurs when a user opens the FileMaker application directly, rather than opening a database file created by FileMaker.

On the Mac, the initial dialog contains a New button.

On Windows, the dialog's Open button can be used to create a new database.

To disable new database creation in the initial open dialog, disable the New Database... command ("<1,A,<1>>") along with any other commands you want disabled as a default setting, and then use the SetStartup command to set the default.

The New button will be completely removed on the Mac. On Windows, the Open button will be disabled, however users can still open FileMaker files by double-clicking them in the selection window.

Disabling New Database Bypasses FileMaker's Initial Open Dialog on OS X: Using the Set Startup command to disable New Database creation in the .ini file results in OS X completely bypassing the initial Open dialog when FileMaker Pro is launched.

Important Note -- Initial 'Templates' Dialog: When opening FileMaker directly, you can actually get one of two different initial dialogs. One is the open dialog mentioned above. The other is a 'templates dialog' that also allows for the creation of new databases using radio buttons (as well as opening existing FileMaker files and templates). SecureFM DOES NOT DISABLE THE NEW DATABASE COMMAND IN THE 'TEMPLATES' DIALOG. To make sure only the open dialog appears, you must uncheck the 'Show Templates in New Database dialog' checkbox in the Application Preferences (in the Edit menu) for all copies of FileMaker on the network. This can be done automatically with New Millennium’s DialogMagic plug-in: http://www.newmillennium.com.

Menu-Rename

External ("Menu-Rename", parameter)

Menu_Rename

Menu_Rename (parameter)

The Menu-Rename function gives you the ability to rename menus and menu items.

Rename Parameter Format:

```
<Context Prefix>:
<Menu#,Mode,
        <
        <Item#,ItemName >,
        <Item#,ItemName >
        >
>
```

Applying to All Files

You can apply a menu configuration globally to all files by using an asterisk by using the default context prefix of “<*>:”

For example, in order to rename the Records menu (menu 6) in Browse mode (B) to "Widgets" and to rename the first few items as well – and apply that change to all contexts as the default – you can use a Menu-Rename string, such as:

```
<*>:
<6,B,
        <
        <0,Widgets>,
        <1,New Widget>,
        <2,Duplicate Widget>,
        <3,Delete Widget>
        >
>
```

By using a default menu definition, you can enforce security globally on new files.

Context-Specific Prefix Syntax

You can also specify Rename configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The format used in the string’s prefix is:

```
<file/table/layout/window>:
```

Context-specific prefixes add significant power and flexibility to your control of FileMaker’s menus!

(For a complete explanation of how to work with Context-Specific Prefixes, see the section on String Prefixes and Different Types of Menu Definitions.)

If you want to change the previous example to apply only to a file named “Widgets”, your string would become:

```
<Widgets///>:
<6,B,
	<
	<0,Widgets>,
	<1,New Widget>,
	<2,Duplicate Widget>,
	<3,Delete Widget>
	>
>
```

Default vs. Specific Precedence

If you have both a default context and specific context string that applies, the specific string will apply.

Tabs and Returns in Strings

SecureFM ignores tabs and returns in the menu definition strings. Tabbed indents are used for clarity, but they are not required. Spaces are also ignored, except when part of a menu or script name.

FMP Disabled Items

If an item is normally disabled by FileMaker Pro (based on mode, field selected, etc.) that item will still be disabled, even if you have modified it with SecureFM functions. SecureFM does not override FileMaker Pro's normal disabling process. If you have renamed an item which FileMaker Pro has disabled, it will appear renamed to the user, but the item will be grayed out.

Similarly, if you have used FileMaker Pro's access privileges to limit access to menu items by defining a password's available menu commands as “Editing Only” or “None”, the limitations of that password privilege will apply and the appropriate items will be grayed out.

Shortcut Keyboard Commands

You can set shortcut keys (Alt-keys) when renaming a menu or menu item by using the ampersand "&" character. The character immediately following the ampersand "&" in the new name will be used as the shortcut key.

To rename the Records menu "Widgets" and define the keyboard shortcut Alt-D to pop down the menu:

```
<Widgets///>:
<6,B,
        <
        <0,Wi&dgets>
        >
>
```

To rename the “New Record” menu "New Widget" and define the keyboard shortcut “N” to select it once the menu has been opened:

```
<Widgets///>:
<6,B,
        <
        <1,&New Widget>
        >
>
```

Duplicated Shortcut Keys

Because SecureFM gives you the freedom to set any keyboard shortcut you want, it is possible to give the same shortcut key to more than one menu or to more than one item within the same menu. Be aware that FileMaker Pro will ignore a keyboard shortcut if it is duplicated among the menu names or among the items within a particular menu.

Scripts Menu Can be Renamed, but Not Scripts Menu Items

You can safely rename the Scripts menu (menu #7), but attempting to rename menu items in the Scripts menu will cause SecureFM to respond with an error.

Windows AppSys and FileSys Menu Items Cannot be Renamed

Menu-Rename cannot be used with the Application System (AppSys) and File System (FileSys) menus (menus 0 and –1, respectively, on the Windows platform). Only menus 1 through 10 (File, Edit, View, etc.) can be renamed by SecureFM.

Items in the AppSys and FileSys menus can still be disabled, however, using the Menu-Disable command.

You can, however, rename the “FileMaker Pro” Application menu on Mac OS X.

X Help Menu Cannot be Renamed on Mac

Attempts to modify the Help menu – with Menu-Disable, Menu-Rename, or Menu-Script – will have no effect on the Help menu. No error is returned if you attempt to modify the Help menu but no changes will occur.

Dividers Cannot be Renamed

You can send a Menu-Rename string that attempts to rename a divider (based on its numerical position in the menu), but that line of the string will have no effect. If you are using the Menu Editor, it will not allow you to attempt to rename dividers.

Sub-Menu Items Cannot be Renamed

Menu items with sub-menus can be renamed, but the sub-menu items themselves cannot be renamed or scripted. If you want to prevent user access to a particular sub-menu item you must disable the menu item that contains it.

Renaming Contextual Menu Items

Contextual Menus are the popup menus that appear when a user does a right-click (on Windows) or Ctrl-click (on Mac).

In the Contextual Menus (menu 20), renaming of items automatically occurs when the corresponding menu items have been renamed in FileMaker Pro’s main menus.

For example, if we use the Widget rename example already mentioned, and then pop down a Contextual Menu in Browse mode when clicking on a layout background, instead of New Record, Duplicate Record, and Delete Record, we get New Widget, Duplicate Widget, and Delete Widget among the Contextual Menu items.

Note: The toolbar icon tool tips, activated by hesitating when mousing over an icon, are not altered by SecureFM.

Passwords

When the plug-in is registered with an optional password, the Menu-Rename string must include the same password for it to work. At the very end of the string, add a pipe character "|" and then the password. This is true for all SecureFM commands that affect menus, including Menu-Disable and Menu-Script.

Menu-Script

External ("Menu-Script", parameter)

Menu_Script

Menu_Script (parameter)

Menu-Script gives you the ability to override FileMaker Pro's normal action when a particular menu item is selected and, instead, run one of your own scripts. With this function, you can make normal FileMaker Pro functions conditional or, used in conjunction with Menu-Rename, you can repurpose FileMaker Pro's menus.

Script Parameter Format:

```
<Context Prefix>:
<Menu#,Mode,
        <
        <Item#,ScriptName >,
        <Item#,ScriptName >
        >
>
```

Applying to All Files

You can apply a menu configuration globally to all files by using an asterisk by using the default context prefix of "<*>:"

For example, to set the first three items in the Records menu to call your own custom scripts – and apply that change to all contexts as the default – you can use a Menu-Script string, such as:

```
<*>:
<6,B,
        <
        <1,My Create New Widget Script>,
        <2,My Duplicate Widget Script>,
        <3,My Delete Widget Script>
        >
>
```

By using a default menu definition, you can enforce security globally on new files.

Context-Specific Prefix Syntax

You can also specify Script configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The format used in the string's prefix is:

```
<file/table/layout/window>:
```

Context-specific prefixes add significant power and flexibility to your control of FileMaker's menus!

(For a complete explanation of how to work with Context-Specific Prefixes, see the section on String Prefixes and Different Types of Menu Definitions.)

If you want to change the previous example to apply only to a file named "Widgets", your string would become:

```
<Widgets///>:
<6,B,
	<
	<1,My Create New Widget Script>,
	<2,My Duplicate Widget Script>,
	<3,My Delete Widget Script>
	>
>
```

Default vs. Specific Precedence

If you have both a default context and specific context string that applies, the specific string will apply.

Tabs and Returns in Strings

SecureFM ignores tabs and returns in the menu definition strings. Tabbed indents are used for clarity, but they are not required. Spaces are also ignored, except when part of a menu or script name.

FileSys and AppSys Menus Cannot be Scripted on Windows

Menu-Script cannot be used with the FileSys and AppSys menus on the Windows platform.

Sub-Menu Items Cannot be Scripted

Sub-menu items cannot be renamed or scripted. If you want to prevent user access to a particular sub-menu item you must disable the menu item that contains it.

X Footer Mode Menu Disabled Rather than Scripted on Mac

If you use Menu-Script to script any of the mode items in the View menu – Browse Mode, Find Mode, Layout Mode, Preview Mode – the corresponding item(s) in the mode menu at the foot of the FileMaker window will be disabled, rather than scripted.

X Help Menu Cannot be Scripted on Mac

Attempts to modify the Help menu – with Menu-Disable, Menu-Rename, or Menu-Script – will have no effect on the Help menu. No error is returned if you attempt to modify the Help menu but no changes will occur.

Scripting Contextual Menu Items

Contextual Menus are the popup menus that appear when a user does a right-click (on Windows) or Ctrl-click (on Mac). Items in Contextual Menus (menu 20) are automatically set to call custom scripts when the corresponding items in FileMaker Pro’s main menus have been scripted.

For example, if we set the Widget script string example above, and then pop down a Contextual Menu in Browse mode, selecting Delete Record (or Delete Widget, if you have renamed the item) will cause the script named “My Delete Widget Script” to run instead of FileMaker’s normal Delete Record process.

Scripting Closeboxes

You can attach scripts to the menu items in the File menu for Exit/Quit and Close, and the corresponding close boxes will then also trigger these scripts.

If you attach a script to the Exit/Quit menu item in the File menu <1,B,<<40,ScriptName>>>, that script will also be called if you click on the application window closebox. Mapping a script to the Close menu item in the File menu <1,B,<<5,ScriptName>>> will call that script when the file window close box is clicked.

Scripted Menu Items in Layout Mode

Be aware that, when FileMaker Pro is in Layout mode, running a script will switch FileMaker Pro into Browse mode.

With the Menu-DoFMPMenuItem function (see below), you can, however, switch back to Layout mode as the final step of your script.

Scripting the Same Menu Item in All Modes

While it may seem convenient to attach a single script to a menu item for all modes, this is often not a good idea. The same menu item may have a different function in different modes, or certain script steps may behave differently in different modes. There is also the issue of running scripts in Layout mode mentioned above. If you assign a script to the same menu item for all modes, make sure you have tested it and thought out all of the consequences. In most circumstances, it makes more sense to define scripts separately for each mode in which you want to override FileMaker Pro's normal menu functions.

SecureFM Script Calls Do Not Exit/Commit the Record

Previously, when SecureFM called a script with one of its functions (Menu-Script, Menu-HotKey, Menu-PerformScript, etc.), all fields were exited as if an Exit Record/Request script step had first been executed. This old behavior caused problems with scripts that were supposed to affect the currently selected field, such as a scripted paste, etc.

Runtime Note: This does not apply to Windows runtime files. The technique used by MenuMagic to prevent FileMaker from exiting or committing the record when triggering a script is not supported by the Windows runtime engine.

Specifying a File Source for Individual Scripts

If you want to trigger a script located in another file, you can specify a file location for an particular script by placing the full file name (with extension) between square brackets just before the script name.

For example, the following string will trigger the “My Exit Script” in a file named “Widgets.fp7” when a user attempts to Exit FileMaker Pro – no matter which file is active when the script is triggered:

```
<*>:
<1,B,
     <
     <40,[Widgets.fp7]My Exit Script>
     >
>
```

If no file source is listed prior to the script name, the plug-in assumes the script exists in the currently active file.

Passwords

When the plug-in is registered with an optional password, the Menu-Script string must include the same password to work. At the very end of the string, add a pipe character "|" and then the password.

Menu-HotKey

External ("Menu-HotKey", parameter)

Menu_HotKey

Menu_HotKey (parameter)

The Menu-HotKey function gives you the ability to trigger a custom script through any keyboard command, such as Alt-E, Ctrl-I, Cmd-Shift-Z, function keys, etc. These hot key combinations can be completely independent of the keyboard commands normally associated with FileMaker Pro's menus. If you define a hot key combination that is also associated with a FileMaker menu function (such as Ctrl-E, which normally calls Delete Record), the hot key script will run instead.

External ("Menu-HotKey", Hot Key Parameter)

Hot Key Parameter Format:

```
<Context Prefix>:
<Mode,
        <
        <Modifier Key #, Primary Key, Script Name >,
        <Modifier Key #, Primary Key, Script Name >
        >
>
```

Modifier Keys and Primary Key

A modifier key doesn't result in text appearing when it is held down; instead, it modifies the primary key being typed. For example, in the key combination Alt-z, "z" is the primary key, and Alt is the modifier key.

Important: Use Lower Case or 'Base' Key

Use lower case or the 'base' key when defining the primary character in the string. In other words, use "b", not "B"; "2", not "@"; "/", not "?"; etc.

Modifier Key Number

FileMaker Pro assigns a number to each of the modifier keys:

1 = Shift
2 = Caps Lock
4 = Control
8 = Alt / Option
16 = Command (Mac)

Menu-HotKey uses the same modifier key numbers.

Command Key Note: The Command key (16) is a Mac-only modifier key.

Caps Lock Key Note: The Caps Lock key (2) is ignored by the Menu-HotKey command in the most recent versions SecureFM.

(For further discussion of modifier key numbers, see the FileMaker Pro documentation on the Get (ActiveModifierKeys) function.)

Applying to All Files

You can apply a menu configuration globally to all files by using an asterisk by using the default context prefix of “<*>:”

By using a default menu definition, you can enforce security globally on new files.

A simple example

If you want to trigger a script named "Go to Utility" whenever Alt-z is typed in Browse mode in a table occurrence named Widgets, the string would look like this:

```
<*>:
<B,
        <
        <8,z,Go to Utility>
        >
>
```

Context-Specific Prefix Syntax

You can also specify HotKey configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The format used in the string's prefix is:

```
<file/table/layout/window>:
```

Context-specific prefixes add significant power and flexibility to your control of FileMaker's hot key commands!

(For a complete explanation of how to work with Context-Specific Prefixes, see the section on String Prefixes and Different Types of Menu Definitions.)

If you want to change the previous example to apply only to a file named "Widgets", your string would become:

```
<Widgets///>:
<B,
        <
        <8,z,Go to Utility>
        >
>
```

If you want it to apply only to a layout named "List", your string would become:

```
<//List/>:
<B,
        <
        <8,z,Go to Utility>
        >
>
```

Multiple Modifier Keys

If you want to trigger a script with a primary character typed in combination with more than one modifier key, the Menu-HotKey function follows the convention used by the Status (CurrentModifierKeys) function and lets you add the two modifier key numbers together.

5 = Control (4) + Shift (1)
9 = Alt / Option (8) + Shift (1)
12 = Alt / Option (8) + Control (4)
17 = Command (16) + Shift (1)
20 = Command (16) + Control (4)
24 = Command (16) + Option (8)
32 = Enter key extended bit (see Enter/Return note below)

Another example

To modify the simple example above to trigger the hot key script with Alt-Shift-z, all we need to do is change the modifier key number to 9:

```
<Widgets///>:
<B,
        <
        <9,z,Go to Utility>
        >
>
```

Hot Keys with No Modifier Keys

The Hot Key function can also trigger a script based only on a primary key being typed without an accompanying modifier key.

You can, for example, repurpose the i, j, k, and m keys to call up, left, right, and down scripts in a FileMaker-based game. Just use a zero “0” for the modifier key section of the string:

```
</Game Table Occurrence//>:
<B,
        <
        <0,i,Move Up>,
        <0,j,Move Left>,
        <0,k,Move Right>,
        <0,m,Move Down>
        >
>
```

Of course, a Hot Key string without modifier keys should be used carefully since it will trigger scripts during normal typing activities.

If you leave the modifier key blank, SecureFM will return an error.

Special Primary Keys

You can also define the primary key with a keycode number. This makes it possible to attach Hot Key scripts to keys that do not normally produce text characters, such as arrow keys, function keys, etc.

The keycode must always be a three-digit numeral. For example, 97 must be written as 097.

The Menu-HotKey format can be rendered in either format below:

```
External (“Menu-HotKey”, “<*>:<B,<<4,a,ScriptName>>>”)
External (“Menu-HotKey”, “<*>:<B,<<4,097,ScriptName>>>”)
```

To find out the correct keycode value for a specific key, use the Menu-GetUserAction function or consult the table below.

Keycode Numbering Not Cross-Platform

The keycode numbers are unique to each operating system. Many keys, such as function keys, have different keycode numbers on different platforms. For example, F1 = 222 on OS X, but F1 = 112 on Windows. When using the three-digit keycode, you may have to create different Menu-HotKey strings for each platform.

Function Keys on OS X

SecureFM uses the standard keycode numbering for the function keys on OS X. Be aware that these numbers do not follow a linear pattern. For example, on OS X F1 = 222, F2 = 220, F3 = 199, etc. Use Menu-GetUserAction or consult the table below to make sure you are using the correct number for each function key.

On Panther versions of OS X, the F10 and F11 function keys will still call Expose functions, even if you have assigned Hot Key scripts to those keys.

On Windows, even if set to call a Hot Key script, the F1 key will still call up the Help Topics window, but the Hot Key script will run in the background.

Special Key Codes

Key	Windows Key Code	OS X Key Code
Clear	144	208
Delete	008	008
Del x>	046	127
Enter	013	003
Return	013	013
Esc	027	027
End	035	004
Help	--	005
Home	036	001
Page Down	034	012
Page Up	033	011
Space	032	032
Tab	009	009
Up Arrow	038	030
Down Arrow	040	031
Left Arrow	037	028
Right Arrow	039	029
F1	112	222
F2	113	220
F3	114	199
F4	115	218
F5	116	196
F6	117	197
F7	118	198
F8	119	200
F9	120	201
F10	--	209
F11	--	203
F12	--	211
F13	--	205
F14	--	207
F15	--	213
F16	--	206
Keypad 1	097	049

Keypad 2	098	050
Keypad 3	099	051
Keypad 4	100	052
Keypad 5	101	053
Keypad 6	102	054
Keypad 7	103	055
Keypad 8	104	056
Keypad 9	105	057
Keypad 0	096	048
Keypad .	110	046
Keypad =	--	061
Keypad /	111	047
Keypad *	106	031
Keypad -	109	045
Keypad +	107	043

Differentiating Enter and Return Keys

The Menu-HotKey function has always been able to distinguish between the "Return" key (on the main keyboard) and the "Enter" key (on the number pad). The ability to distinguish between Return and Enter allows developers to exercise greater control over user behavior in their solutions.

Example #1: When a script is paused, the Enter key (but not the Return key) will trigger the Continue button.

Example #2: When a FileMaker record is being edited, the changes aren't recorded until the record is committed. One way to commit a record is to press the Enter key (but not the Return key).

In both examples, it may be desirable to prevent the Enter key from performing its usual behavior by remapping it using Menu-HotKey.

While it may be desirable in certain circumstances to prevent the user from entering paragraph returns, it should be a conscious decision on the part of the developer, not the unintentional by-product of remapping the Enter key!

On the Mac platform, you can easily distinguish between the Enter and Return keys because they have unique key code values:

```
<*>:
<B,
        <
        <0,003,Enter Script>,
        <0,013,Return Script>
        >
>
```

The Windows system assigns the same key code to both keys, however. The Enter and Return keys are distinguished, instead, by the extended modifier key 32. The Windows Hot Key string would look like this:

```
<*>:
<B,
        <
        <32,013,Enter Script>,
        <0,013,Return Script>
        >
>
```

Important: On Windows, the Hot Key string must list the line for the Enter key script first. If you reverse the order and list the Return key script first, both keys will trigger the Return script.

Clearing Hot Key Settings

As with Menu-Disable, Menu-Rename, and Menu-Script, you can clear all Hot Key settings by using SecureFM's 'clear all' parameter ":-"

```
Set Field [Response Field, External ("Menu-HotKey", ":-")]
```

Specifying a File Source for Individual Scripts

If you want to trigger a script located in another file, you can specify a file location for an particular script by placing the full file name (with extension) between square brackets just before the script name.

For example, the following string will trigger "My Script" in a file named "Widgets.fp7" when a user types Ctrl-a – no matter which file is active when the script is triggered:

```
<*>:
<1,B,
     <
     <4,a,[Widgets.fp7]My Script>
     >
>
```

If no file source is listed prior to the script name, the plug-in assumes the script exists in the currently active file.

Passwords

When the plug-in is registered with an optional password, the Menu-HotKey string must include the same password to work. At the very end of the string, add a pipe character "|" and then the password.

String Prefixes and Different Types of Menu Definitions

Menu definition strings require a prefix in addition to the actual code that defines which menu items are altered. This new prefix gives you the ability to create default menus that are applied globally to all files, while other menu definitions are active only for specific files, table occurrences, layouts, and/or windows in your solution. This also gives you the advantage of being able to load several different menu definitions all at once when your solution is launched to be applied only under certain circumstances.

Menu definition strings generally follow the format:

```
<Menu Context String Prefix>:
<Menu Definition String>
```

Default Menu Definition

The default menu definition is globally applied to each file when no context-specific menu definition applies. You may commonly want to use this definition to limit availability of functions for all files in your solution without needing to set specific rules per table occurrence or if you are a system administrator and you want to universally disable certain features for your users (New Database, Find/Replace, Replace Contents, Relookup Contents).

By using a default menu definition, you can enforce security globally on new files, without needing to know the name of every file that might be opened. The “default” prefix is “<*>”.

Default String Format:

```
<*>:
<Menu Definition String>
```

Disable Example:

```
<*>:
<1,B,<1,2,3>>
<2,B,<0>>
```

For security reasons, it often makes sense to create a default disable menu definition.

On the other hand, you may not want to use a default at all. This would be the case if your users could be opening files which are not part of your solution and you want them to have full FileMaker Pro functionality in any files or table occurrences other than ones which you specifically want to control. In this situation, to have a “universal setting” for all of your files only, you would use a “table occurrence-specific” menu definition, listing ALL of your files’ table occurrences in the prefix (see below for more info about table occurrence-specific menu definitions).

Rename Example (the following definition would affect all files (and therefore may or may not be desirable):

```
<*>:
<6,B,
        <
        <0,Widgets>,
        <1,New Widget>,
        <2,Duplicate Widget>,
        <3,Delete Widget>
        >
>
```

In many solutions, it makes sense to rename menus and menu items uniquely for each table occurrence. For that reason, you may not want to set a default rename menu definition and, instead, use only table occurrence-specific rename menu definitions (see below).

Script Example:

```
<*>:
<6,B,
        <
        <1,My New Record Script>,
        <2,My Duplicate Record Script>,
        <3,My Delete Record Script>
        >
>
```

For a script menu definition to work as a default, you must have scripts with the same script names in each file – even if the scripts themselves work differently from to file. Otherwise, you may prefer to only use table occurrence-specific script menu definitions (see below).

Context-Specific Prefix Syntax

Beginning with SecureFM 7.1 (and all subsequent versions) you can now specify Disable, Rename, Script, and HotKey configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The new format used in the string's prefix is:

<file/table/layout/window>:

This change adds significant power and flexibility to your control of FileMaker's menus!

Menus by File Name

If you want to automatically disable Layout Mode whenever a file named Contacts.fp7 is the frontmost file, your Menu-Disable string would be:

<Contacts///>:
<3,B,<3>>

If you want the same disable string to apply to more than one file, such as Contacts.fp7 and Invoices.fp7, just create additional context groupings separated by commas:

<Contacts///,Invoices///>:
<3,B,<3>>

Menus by Table Occurrence Name

In addition to the old TO-specific format described above, you can now link menu configurations to the table occurrences using the new syntax. For example, if you want SecureFM to disable Layout Mode whenever a table occurrence named "table occurrence 1" is active through the current layout, your Menu-Disable string would be:

</table occurrence 1//>:

```
<3,B,<3>>
```

If you want the same disable string to apply to more than one table occurrence, just create additional context groupings separated by commas:

```
</table occurrence 1//,/table occurrence 2//,/table occurrence 3//>:
<3,B,<3>>
```

Menus by Layout Name

To disable menus when a layout named “Entry” is active, your Menu-Disable string would be:

```
<//Entry/>:
<3,B,<3>>
```

If you want the same disable string to apply to more than one layout, just create additional context groupings separated by commas:

```
<//Entry,List/,//Detail/>:
<3,B,<3>>
```

Menus by Window Name

To disable menus when the active window is named “MySolution”, your Menu-Disable string would be:

```
<///MySolution>:
<3,B,<3>>
```

If you want the same disable string to apply to more than one window name, just create additional context groupings separated by commas:

```
<///MySolution,///SolutionWindow>:
<3,B,<3>>
```

Hierarchy Precedence

If you have activated several menu strings and the criteria of more than one string matches the current context, the string with the rightmost, more specific match is used.

For example –

</ViewInvoice/>:

-- will take precedence over –

<MyFile///>:

This allows you to set up a hierarchy of configurations, where the more specific ones take precedence over the more general.

Multiple Match Criteria

You can create very precise match criteria in your strings by defining more than one element in the prefix. For example:

<Contacts//Entry/>:
<3,B,<3>>

The above menu configuration will apply when the active file is Contacts.fp7 AND the active layout name is "Entry". **Remember that ALL the criteria must match.**

Variable Match Names

SecureFM incorporates support for regular expressions (RE's) in the match criteria of your string prefix. With RE code you can, among other things, set a menu configuration to be applied to variable contexts, rather than having to list each individual file name, table occurrence name, etc.

For example, if you want your menu configuration to be applied to three table occurrences named "Contacts Customers," "Contacts Vendors," and "Contacts Details," you can list each table occurrence name in the string –

</Contacts Customers//,/Contacts Vendors//,/Contacts Detals//>:
<3,B,<3>>

-- or you can use regular expressions to apply the menu configuration to all table occurrences that begin with "Contacts" --

```
</=Contacts.*//>:
<3,B,<3>>
```

The essential format options for variable names using RE's are:

match	Element name is "match"
=match.*	Element name begins with "match"
=.*match	Element name ends with "match”
=.*match.*	Element name contains "match"

There is extensive information available on the internet for advanced uses of regular expressions.

(Thanks to Henry Spencer for the regular expressions engine used by SecureFM. Copyright © 1998, 1999 Henry Spencer. All rights reserved.)

Restoring Menus and Special Settings

No Files Open Menu Definition

Finally, you can set up a “no files open” menu definition that applies when the FileMaker Pro application is running but no individual files are open.

You can use a “no files open” menu definition for Menu-Disable and Menu-Rename, but there is no reason to have a “no files open” Menu-Script definition since at least one file must be open to hold the necessary scripts.

If you have a default menu definition active but not a “no files open” menu definition, the default will apply. The prefix for “no files open” is “<~>”.

No Files Open String Format:

```
<~>:
<Menu Definition String>
```

Disable Example:

```
<~>:
<1,B,<1,2,3>>
<2,B,<0>>
```

Rename Example:

```
<~>:
<6,B,
        <
        <0,Widgets>,
        <1,New Widget>,
        <2,Duplicate Widget>,
        <3,Delete Widget>
        >
>
```

Since scripts are always associated with an open file, a Menu-Script definition for the situation when no files are open becomes meaningless.

Restoring Menus

The fundamental technique of restoring FileMaker Pro’s normal menus is to send a blank "" menu definition string with Menu-Disable, Menu-Rename, Menu-Script, and Menu-HotKey.

To accommodate the several types of menu definitions that can be active at the same time, however, SecureFM offers several levels of restoring menus which can be used:

Clear All Menu Definitions

If you set Menu-Disable, Menu-Rename, Menu-Script, or Menu-HotKey with a colon followed by a dash “:-” as the parameter, SecureFM will revert back to FileMaker Pro menus, clearing all default, table occurrence-specific, and “no file open” menu definitions.

Important Note: Be aware that the Clear All Menu Definitions command affects all SecureFM strings in all files.

Clear All Format:

External (“Menu-Disable”, “:-”)
External (“Menu-Rename”, “:-”)
External (“Menu-Script”, “:-”)
External (“Menu-HotKey”, “:-”)

Use FileMaker Pro Menus

You may want to specify FileMaker Pro’s normal menus for a specific table occurrence. You may do this temporarily (for instance as a developer doing some administrative or development work at a user’s computer) or in the instance when you have a default setting applied to all files but want to specifically have normal FileMaker Pro menus within a certain table occurrence.

To display FileMaker Pro menus, use a blank string “” – following the table occurrence list prefix.

It is important to understand that this is not the same as clearing menu definitions. Clear All Menu Definitions “:-” and Clear Current Setting “-” actually remove menu definitions from SecureFM's memory. Using a blank string "", on the other hand, sets a menu definition for the given context telling SecureFM not to override FileMaker Pro's normal menus; it does not clear any other menu definitions that may be set, such as the default setting.

Use FileMaker Pro Menus Format:

<Context Prefix>:

(Nothing follows the colon.)

To set a table occurrence-specific menu definition to use FileMaker Pro's menus:

External ("Menu-Disable", "</Table Occurrence 1//>:")
External ("Menu-Rename", "</Table Occurrence 1//>:")
External ("Menu-Script", "</Table Occurrence 1//>:")

Clear Current Setting

SecureFM stores an internal table of menu definitions. Sending a dash (think of it as a minus sign) "-" removes the currently stored menu definition for that setting (default, TO-specific, or no files open).

For example, "<*>:-" will remove the definition (if any) that had been set as the default.

The menu definition hierarchy (see above) determines the result when clearing existing settings.

Clear Current Setting Format:

<Context Prefix>:-

Backward Compatibility

For backward compatibility, the updated version of SecureFM still handles the old menu definition string format (used by pre-6.1 versions of SecureFM).

When no prefix is included with the disable string, it is assumed to be a default menu definition that applies to all files. This means that all old-style SecureFM strings are treated as default menu definitions.

Old TO-Specific Syntax

Beginning with SecureFM 7 (up until 7.0.3), SecureFM allowed for table occurrence-specific menu configurations using the following format:

<table occurrence 1, table occurrence 2, table occurrence 3>:
<3,B,<3>>

Using that format, the menu configuration (in this example, disabling Layout Mode – item #3 in menu #3) was immediately activated whenever table occurrence 1, 2 or 3 became active.

The current version of SecureFM still recognizes this old table occurrence-specific syntax for backward compatibility, but SecureFM offers you considerably more flexibility…

When all files are closed

If a user closes all file windows but keeps the FileMaker Pro application open, the "no file open" menu definition (see above) will be used. If you have not set a "no files open" definition, then the default definition will be used. If neither a default nor "no file open" menu definition has been set, then FileMaker Pro's normal menus will become active.

Other SecureFM Functions

Menu-StoreText

External ("Menu-StoreText", parameter)

Menu_StoreText

Menu_StoreText (parameter)

The Menu-StoreText function stores a block of text in an internal SecureFM dictionary. It can be used to store a string for use in SecureFM, or for storing any other value for any purpose. It can be useful for defining a variable for use later in a script or at any time during the user's session.

Store Text Parameter Format:

`External ("Menu-StoreText", Key|StoredText)`

The stored text can be anything and any length, though it may be limited by FileMaker's plug-in API or the total application memory available for FileMaker Pro. The key is case sensitive and can be up to 50 characters. The SecureFM dictionary is limited to 200 entries.

All text stored is globally available. You can store text by calling the Menu-StoreText command and then later retrieve the text using the Menu-GetText function (see below). You can store text in one file and retrieve it in another file. In other words, the SecureFM dictionary functions allows you to store truly global data without being dependent on a particular FileMaker file being open.

Menu-StoreText uses application memory to hold the text. You may need to be aware of memory allocated to FileMaker Pro if you are storing many large blocks of text.

When using this function, it is a good idea to also trap for errors returned by the plug-in.

The stored text dictionary is cleared each time FileMaker Pro is shut down.

Menu-GetText

External ("Menu-GetText", parameter)

Menu_GetText

Menu_GetText (parameter)

Menu-GetText retrieves text that has been stored by SecureFM using the Menu-StoreText function (see above). The key must match the corresponding key used in the original Menu-StoreText command.

Get Text Parameter Format:

External ("Menu-GetText", Key)

The Menu-GetText function does not return any errors. If no match for the key is found, SecureFM returns a blank response ("")

Creating a Menu Definition Dictionary

The Menu-StoreText and Menu-GetText functions have many possible uses, but one handy application is to create a menu definition dictionary, allowing you to store many menu definitions and activate them by sending a simple key word or phrase, rather than setting elaborate strings.

For example, suppose you want to work with two different disable menu definitions based on a logged in user’s privilege level.

For basic access, you might have a string like (oversimplified):

```
<*>:
<1,A,<0>>
```

For management level access, your string might be (also oversimplified):

```
<*>:
<1,B,<5,6,7,8,9,10>>
```

You can store them in the SecureFM dictionary:

```
External (“Menu-StoreText”, “Basic|<*>:<1,A,<0>>”)
External (“Menu-StoreText”, “Management|<*>:<1,B,<5,6,7,8,9,10>>”)
```

Then, if a user is logged in with “Basic” privileges, you can easily disable menus correctly without worrying further about menu definition string codes:

```
External (“Menu-Disable”, External (“Menu-GetText”, “Basic”))
```

On the other hand, if a user logs in with “Management” privileges, you can easily call up the correct menu definition with:

```
External (“Menu-Disable”, External (“Menu-GetText”, “Management”))
```

Note that there may be other, and in some cases preferable, ways to store multiple menu definitions, including in the SecureFM Menu Table file which comes with SecureFM.

Menu-GetUserAction

External ("Menu-GetUserAction", parameter)

Menu_GetUserAction

Menu_GetUserAction (parameter)

The Menu-GetUserAction function allows you to find out which FileMaker Pro menu item was last selected by a user and in which file. It also gives you the numeric key code of the last keystroke typed.

External ("Menu-GetUserAction", "")

No parameter is used.

Menu-GetUserAction returns five values in its response, each separated by a pipe character "|":

Menu #|Menu Item #|(unused)|File Name|Key Code

Menu Selection Example

If the last menu item selected by the user was the Open command (menu item #2) in the File menu (menu #1), and this was done in a file named "Customers.fp7", the Menu-GetUserAction response might be:

1|2|0|Customers|222

Key Code

The last value in the response is the keycode. The keycode is a numeric value, up to three digits long, that represents the last key typed.

This keycode value can be used by Menu-HotKey to attach hot key functions to keys that do not produce text characters, such as arrow keys, function keys, etc.

When checking for keycode values, be aware that not all keycodes are identical on all platforms. Some keys do indeed have the same value on different platforms, while others, especially the non-text keys, quite often have different values on different platforms.

Unused Third Value

The third value in the response is always zero and is currently unused.

Menu-DoFMPMenuItem

External ("Menu-DoFMPMenuItem", parameter)

Menu_DoFMPMenuItem

Menu_DoFMPMenuItem (parameter)

This function gives you the ability to call any FileMaker Pro menu item directly from within a script.

Parameter Format:

External (“Menu-DoFMPMenuItem”, Menu#|Item#)

Script Entering Layout Menu

Many FileMaker Pro menu items already have an equivalent script step available, but not all. For example, there are script steps already available to enter Browse, Find, and Preview modes, but there is no script step to enter Layout mode. The Menu-DoFMPMenuItem function lets you directly call the Layout Mode item in the View menu.

Using our Layout mode example above, the View menu is menu # 3 (counting from the left) and Layout Mode is item # 3 (counting from the top). The command would then be:

External (“Menu-DoFMPMenuItem”, “3|3”)

Preceding Script Step

To work reliably, this function must be preceded by another script step. You can use a Pause/Resume Script step set to zero seconds, but that may cause problems if you have other scripts queued up to run. The solution we recommend is to precede the Menu-DoFMPMenuItem command with a Go to Layout script step:

```
Go to Layout [original layout]
Set Field [Response Field, External (“Menu-DoFMPMenuItem”, “3|3”)]
```

Note that the Go to Layout script step will exit the record, causing the cursor to exit the currently selected field. If you want the triggered menu item to act on the selected field, such as with the Paste command, then the preceding script step should not be Go to Layout. A blank Set Field step works well:

```
Set Field [“”]
Set Field [Response Field, External (“Menu-DoFMPMenuItem”, “215”)]
```

Important: Does not work in FileMaker 11 on Mac
This function has been primarily used to script entry into Layout Mode. While this continues to work on Windows, it no longer works in FileMaker 11 on Mac. To provide an alternate way to script normally non-scriptable FileMaker actions, like Layout Mode, we have created a new function: Menu_DoFMPCommand. (See the documentation on Menu_DoFMPCommand.)

Menu_DoFMPCommand

Menu_DoFMPCommand (parameter)

The Menu-DoFMPMenuItem function, used primarily to enter Layout Mode by script, does not work in FileMaker 11 on the Mac (though it works with FM 11 on Windows). For solutions that depend on the ability to script access to Layout Mode, the Menu_DoFMPCommand function has been created as an alternate way to perform FMP commands. Rather than menu and item number in the parameter, this new function requires FileMaker's internal command number.

Parameter Format:

Menu_DoFMPCommand (command#)

The function always returns "0".

Layout Mode

To enter Layout Mode, use—

Menu_DoFMPCommand (49156).

Other Commands

To find the command number for other FileMaker actions, call Menu-GetUserAction function, and look at the last value returned.

Menu-Event

External ("Menu-Event", parameter)

Menu_Event

Menu_Event (parameter)

The Menu-Event function allows you to trap for certain events. There are three specific events which are available:

1. The active table occurrence changes (by changing layouts or file window)
2. When SecureFM is unable to call a script
3. When a user attempts to disable SecureFM in the Application Preferences.

Whenever one of those events occurs, SecureFM can be set to run a specified script.

The Menu-Event function is global and stays active until cleared by sending a new or blank "" event parameter.

External ["Menu-Event", Event Parameter(s)]

The event parameter format varies slightly depending on which types of events are being trapped:

TableOccurrenceChange
ScriptError
SFMDisable

Important: Using the Menu-Event command clears all previously set Menu-Event functions. If you intend to use more than one of these functions, you will most commonly want to set all of them at the same time, usually when your solution is launched (see below for more details).

TableOccurrenceChange

(Called “WindowActivate” in previous versions of SecureFM.)

TableOccurrenceChange will trigger a script whenever the active table occurrence changes, because of a change of layout, window, etc.

The TableOccurrenceChange script can be a script with the same name in every file within your solution, or you can call a single script in a central file.

In other words, this powerful function gives you the ability to detect whenever the active table occurrence has changed (because a new file window has become the front-most file or a layout has changed, etc. – even when done by manually navigation) and will then run any script you specify.

Event Parameter Format:

<TableOccurrenceChange,ScriptName>

So the syntax for the full function would be:

External("Menu-Event",<TableOccurrenceChange,ScriptName>)

ScriptError

When a SecureFM function triggers a script – such as when a user selects a menu item that has been modified to call a custom script with the Menu-Script function – but the expected script is not found, a generic dialog will appear explaining the problem. If, however, a "ScriptError" script has been set, and the initial script is not found, the ScriptError script will run instead.

Event Parameter Format:

<ScriptError,ScriptName>

For troubleshooting purposes, you may want your ScriptError script to make use of the Menu-GetScriptCaller function (see below) to find out which script failed and why.

The syntax for specifying a file name when referring to a script in SecureFM is to precede the script name with the file name in square brackets, hence:
<ScriptError,[FileName]ScriptName>

Failure of the ScriptError Script

If the ScriptError script itself fails, the generic error dialog will appear, showing the original script and which function called it.

Suppressing the ScriptError

There are times when you may not want a script error to be reported to the user, either as SecureFM's generic dialog or as part of a ScriptError script.

In order to avoid these situations, you can set the ScriptError to be suppressed by using a tilde character "~" in place of a script name:

<ScriptError,~>

This will prevent any error dialog from appearing.

SFMDisable

This function provides added security to ensure that a user can't disable SecureFM and still have access to your solution.

Event Parameter Format:

<SFMDisable,ScriptName>

The specified script will run immediately if any user attempts to disable SecureFM by unchecking it in the Application Preferences plug-ins list. This is important because unchecking SecureFM will immediately re-enable all menu items. If a user tries to turn off SecureFM, you may want a script to run to immediately quit FileMaker Pro. The specified script should probably be in a file which is guaranteed to be open if your solution is running. When that file closes, it should most likely close all files in your solution.

The syntax for specifying a file name when referring to a script in SecureFM is to precede the script name with the file name in square brackets, hence:

<SFMDisable,[FileName]ScriptName>

Another option you have to prevent a user from deactivating SecureFM in Application Preferences is to disable the Preferences menu item with the Menu-Disable command. However, if a savvy user creates a FileMaker file with a script which opens Application Preferences, they could still get around your disabling menu definition – so the SFMDisable option of the Menu-Event function will enable you to close that loophole. Airtight security will require careful thought regarding how your "On Open" and "On Close" scripts work in a primary file in your solution. At a minimum, the On Open script should most likely verify that SecureFM is registered, and otherwise not allow your solution to launch.

Menu-Event Resets All Previously Defined Events

Using the Menu-Event command clears all previously set Menu-Event functions. For example, if you use Menu-Event to specify a TableOccurrenceChange script and then, in a separate script step, use Menu-Event to set a ScriptError script, the TableOccurrenceChange script will be forgotten – unless you also respecify the TableOccurrenceChange script, as well.

The Menu-Event command can include any or all of the specifiers (TableOccurrenceChange, ScriptError, SFMDisable) in the same step. These specifiers are not separate commands that operate independently of each other. They can be thought of as sub-functions of the Menu-Event command.

For this reason, it usually makes sense to set all three Menu-Event values in the same step:

```
External("Menu-Event",
"<TableOccurrenceChange,ScriptName>
<ScriptError,ScriptName>
<SFMDisable,ScriptName>")
```

Under most circumstances, the Menu-Event specifiers are not functions you would set dynamically at different times. It makes sense for most solutions to set all applicable Menu-Event specifiers once in a single Menu-Event command when your solution is launched.

Configuring the Event Commands with a Password

If you have registered SecureFM with a password (see Menu-Register), the same password is required to use the Menu-Event command. To use a password, you must add an additional pipe character "|" and the password to the end of the Menu-Even string.

Ex: Password in a Simple String

For example, if your original Menu-Event string was:

```
External ("Menu-Event", "< TableOccurrenceChange,Disable Menus>")
```

It would become:

```
External ("Menu-Event", "< TableOccurrenceChange,Disable Menus>|Password")
```

Password Error

If the password is missing or wrong in the Menu-Event string, SecureFM returns "Incorrect password" and does activate the Event command.

Menu-PerformScript

External ("Menu-PerformScript", parameter)

Menu_PerformScript

Menu_PerformScript (parameter)

Menu-PerformScript immediately runs the specified script. With Menu-PerformScript you can trigger a script from any FileMaker Pro calculation - this could be a calculation field, a field validation, an auto-enter calculation, etc...

Parameter Format:

External ("Menu-PerformScript", ScriptName)

Performing a Script on Exiting a Field

You can use the Menu-PerformScript function to trigger a script when a user exits a field.

Working with FileMaker 7 or later, place the Menu-PerformScript function into the auto-enter by calculation. It is important to UNcheck 'Do not replace existing value of field.' You must also incorporate the field itself into the auto-enter calculation value; otherwise, it will be cleared each time a user exits the field:

Field Name & External ("Menu-PerformScript"; Script Name)

See the PerformScript on Exiting Field example within the Misc. Example file for more information.

Optional Parameter for Menu-PerformScript and Menu-GetScriptCaller

Menu-PerformScript can now pass a script parameter to the script it calls, giving you the ability to maintain data on the original file, layout, record #, etc. while the script runs!

An optional parameter can be submitted with the Menu-PerformScript function, separated from the Script Name by a pipe character “|”:

`External ("Menu-PerformScript", ScriptName|OptionalParameter)`

This optional parameter can contain any data up to 300 characters in length.

You can later retrieve that data using Menu-GetScriptCaller (see below). If the last SecureFM function to call a script was Menu-PerformScript and an optional parameter was included, Menu-GetScriptCaller will return the optional parameter as the third parameter in its response:

`[File Name]Script Name|SecureFM Function|Optional Parameter`

(For more information on Menu-GetScriptCaller, see the documentation below.)

Using FM’s Get (ScriptParameter) Function to Retrieve the Optional Parameter

You can also retrieve the script parameter, along with the file name and the name of the script that was performed, by using FileMaker's Get (ScriptParameter) function. The format for the data returned by Get (ScriptParameter) in this instance is:

`[File Name]Script Name|SFM Function Name|Script Parameter`

If you want to quickly isolate the value of the script parameter, you can use the Menu-ExtractParameter function:

`External ("Menu-ExtractParameter"; "3|" & Get (ScriptParameter))`

Example: Passing the Layout Name to Your Script

The optional script parameter can be the contents of a field or some other information, like the layout name:

```
External ("Menu-PerformScript"; "ScriptName|" & Get (LayoutName))
```

Why this is needed:

One common way to use the Menu-PerformScript function is to put it in a field auto-enter calculation (described above). In that situation, Menu-PerformScript gets triggered every time the field is modified, and allows the developer to modify other data in the same record based on the field value that was just modified. This works well most of the time.

However, there are times when a field can be exited by means other than simply tabbing to another field or clicking in the background of the layout. A field can be exited by an event which leaves the record – clicking on another record (if in list view), using the record rolodex in FileMaker Pro’s Status Area to switch records, running a script which leaves the record, etc. – and all of these events will trigger the script, but the script will operate on the data in the wrong record.

By adding an optional parameter, you can submit information about the status of the database when the script was triggered, such as the current record ID, which can then be used in the triggered script to evaluate if the record is still the same or if you must first return to the correct record.

Menu-GetScriptCaller

External ("Menu-GetScriptCaller", "")

Menu_GetScriptCaller

Menu_GetScriptCaller ("")

Menu-GetScriptCaller can be used as part of a ScriptError script (see Menu-Event) to find out which script was last triggered and which SecureFM function attempted to call the script. This function can also be helpful in debugging.

External ("Menu-GetScriptCaller", "")

No parameter is used.

Returns the following information:

[File Name]Failed Script Name|SecureFM Function|Table Occurrence|Menu #|Menu Item #|Modifier Key #|Key Character|Key Code

A simple example:

"[Widgets.fp7]Go to Menu|TableOccurrenceChange|Widget Customers|0|0|0|0|0"

-- means that the attempt to run a script named "Go to Menu" in the Widgets.fp7 file failed. The SecureFM function that attempted to call the script was the TableOccurrenceChange specifier for the Menu-Event function. In your ScriptError script, you can parse this result to provide an informative error message, either for a user, or, perhaps more importantly, for yourself when you are developing.

Example: Hot Key

"[Widgets.fp7]My Paste Script|HotKeyScript|Widget Customers|0|0|1|d|100"

-- means that the attempt to run a script named "My Paste Script" in the Widgets.fp7 file failed. The SecureFM function that attempted to call the script was the Menu-HotKey function, triggered by typing Shift (mod key #1) and “D” (key code #100).

Example: Scripted Menu Item

"[Widgets.fp7]My New Script|MenuScript|Widget Customers|6|1|0|0|0"

-- means that the attempt to run a script named "My New Script" in the Widgets.fp7 file failed. The SecureFM function that attempted to call the script was the Menu-Script function, triggered by selecting “New Record” (item #1) in the Records menu (menu #6).

In your ScriptError script, you can parse this result to provide an informative error message, either for a user, or, perhaps more importantly, for yourself when you are developing.

Menu-ExtractParameter

External ("Menu-ExtractParameter", parameter)

Menu_ExtractParameter

Menu_ExtractParameter (parameter)

Allows you to easily extract a value from a list separated by pipe characters "|". This is helpful when used in conjunction with Menu-GetScriptCaller, Menu-GetUserAction or any other plug-in result (not limited to New Millennium plug-ins), that returns multiple values separated by pipe characters.

Parameter Format:

```
External("Menu-ExtractParameter", Parameter#|ParameterList)
```

Menu-ExtractParameter views the first parameter ("Parameter#") on its own; all subsequent parameters divided by pipe characters "|" are seen as a separate "Parameter List" string.

The first parameter tells SecureFM which of the subsequent parameters to extract.

For example, if the Menu-ExtractParameter string is "3|Joel|Fred|Nancy|Susan", SecureFM will return "Nancy", the third parameter (after the parameter #).

If the Parameter # specifies a number higher than the number of parameters in the Test String, Export-ExtractParameter returns the last one. If the Parameter Number is zero "0" or a negative number, the number of parameters is returned instead.

Menu-ExtractParameter is even helpful for general use in parsing lists. If you have a list of values separated by return characters, you can first substitute pipe characters for the paragraph returns, and then easily extract the values in the list with this function. This is much easier than wrestling with FileMaker’s "Position" and "Middle" functions for multiple values.

Example: Extracting Parameters from Menu-GetScriptCaller

To return the name of the last script triggered by SecureFM, you can use:

```
External("Menu-ExtractParameter","1|"& External("Menu-GetScriptCaller", ""))
```

To get the function that called it:

```
External("Menu-ExtractParameter","2|"& External("Menu-GetScriptCaller", ""))
```

Example: Extracting Parameters from Menu-GetUserAction

To return the keycode value of the last key typed, you can use Menu-ExtractParameter in conjunction with Menu-GetUserAction:

```
External("Menu-ExtractParameter","5|"& External("Menu-GetUserAction", ""))
```

Counting the Number of Parameters

Setting the initial parameter to zero “0” tells Menu-ExtractParameter to count the number of parameters separated by pipe characters that follow. For example, sending—

```
External("Menu-ExtractParameter","0|apples|oranges|papayas")
```

—returns “3” to the response field.

When Parameters are Divided by a Different Character

If you have a string of values, but they are separated by a character other than the pipe character, you can still use the Menu-ExtractParameter function. All you need to do is use FileMaker Pro’s Substitute command.

If, for example, you have a field named “Shopping List” that contains a list of values separated by returns—

```
apples
oranges
papayas
```

—and you want to extract the first item in the list, you can substitute the pipe character “|” for the return character “¶”:

```
External("Menu-ExtractParameter","1|" & Substitute (Shopping List, "¶", "|"))
```

Menu-IsFrontTableOccurrence

External ("Menu-IsFrontTableOccurrence", parameter)

Menu_IsFrontTableOccurrence

Menu_IsFrontTableOccurrence (parameter)

This is function quickly tells you if a specific FileMaker table occurrence is the active layout in the front file window.

Parameter Format:

External("Menu-IsFrontTableOccurrence", Table Occurrence Name)

Returns 1 if the Table Occurrence Name parameter is identical to the table occurrence name of the active layout in the front window; 0 otherwise.

Menu-ContextChange

External ("Menu-ContextChange", parameter)

Menu_ContextChange

Menu_ContextChange (parameter)

Menu-ContextChange allows you to trigger a script whenever the context changes to match the file, table occurrence, layout, and/or window name context you've defined.

Script Parameter Format:

<Context Prefix>:
<Mode,Script Name>

Context-Specific Prefix Syntax

You can also specify ContextChange configurations to be activated by any or all of the following:

File Name
Table Occurrence Name
Layout Name
Window Name

The format used in the string's prefix is:

<file/table/layout/window>:

Context-specific prefixes add significant power and flexibility to your control of FileMaker's menus!

(For a complete explanation of how to work with Context-Specific Prefixes, see the section on String Prefixes and Different Types of Menu Definitions.)

Examples

If you want to trigger a script named "Go to Menu" every time a file named "Widgets" comes to the front in Browse mode, your string would be:

```
<Widgets///>:
<B,Go to Menu>
```

If you want to trigger a script named Contact Record Check whenever the Window name is “Widget Contacts,” your string would be:

```
<///Widget Contacts>:
<B,Contact Record Check>
```

Menu-FMPWindow

External ("Menu-FMPWindow", parameter)

Menu_FMPWindow

Menu_FMPWindow (parameter)

This function gives you the ability to move and size the FileMaker application window.

This is a Windows-only function since there is no separate application window on the Macintosh platform.

FMPWindow Parameter Format:

left|top|width|height

Setting Position and Dimensions

To position the top left corner of the application window at 20, 20 with dimensions of 800 x 600, the string would be:

20|20|800|600

You can use FileMaker Pro’s built-in functions to get the current screen height and screen width and position the application window specific to each display. You can, for example, calculate how to precisely center the application window.

Maximizing the Application Window

To maximize the application window, use the following string:

-1|-1||

SecureFM's Password Feature

Password Registration

SecureFM's Password is an advanced feature that gives you an added layer of security, allowing you to ensure that only the solution creator can control FileMaker's menus and toolbars.

Using a password is entirely OPTIONAL. This makes SecureFM backward compatible with plug-in commands designed to run on earlier versions of SecureFM. It also means you can simply choose not to add a password to your solution if you feel that this advanced level of security isn't necessary for your solution.

Password for Added Security

Registering SecureFM with a password prevents users from re-configuring the menus unless they know the password. When SecureFM is registered with a password, that password must also be used to change menu configurations and toolbar access.

If you don't use a password, a FileMaker-skilled user who has script access to any FileMaker file and understands the syntax of SecureFM could enable all menu items. Even a novice user who has access to the SecureFM Menu Editor file (which can be downloaded by anyone from New Millennium's web site) can simply hit the "Re-enable All" button. Registering SecureFM with a password prevents these possibilities.

For details on how to use passwords with SecureFM, see the documentation on Menu-Register and any other SecureFM functions you wish to use in your solution.

FileMaker's Custom Menus and SecureFM

Comparing Parallel Features

There are several crucial differences between FileMaker's custom menus and SecureFM's capabilities:

	SecureFM	FMP Custom Menus
Modifiable Menus	Allows you to modify ALL menus.	Does NOT allow you to modify the Format, Scripts, or Windows menus
Menu Sets for Multiple Files	Allows you to define menus centrally. You can define sets of menus to apply to all files open, or to specific files, table occurrences, layouts, and/or windows	Requires you to create menu sets file-by-file.
Dynamic Use of Menu Sets	Using regular expressions, SecureFM allows you to set custom menus to groups of files, table occurrences, layouts, or windows that share some common element in their names.	Nothing equivalent
Adding New Layouts, etc.	Menus can be dynamically tied to context elements (file, table occurrence, layout, window). Much easier to implement and more reliable, since menus can apply to new layouts automatically.	Menus can be changed only by script or by linking a menu to each layout. If using the layout method, this needs to be done manually for each new layout.
Modes	Allows you to define menus in ALL modes, including Layout mode	Does NOT control menus in Layout mode.
Window Widget Buttons	Full control over window widget buttons. Allows you to modify both the application and window closeboxes, as well as the minimize and restore widgets	Only controls the basic window closebox
Quit / Exit	Allows you to disable, rename and script the Quit/Exit menu item	No control over Quit/Exit
Compatibility with FMP 7	SecureFM will work with FMP 7 through 11	Custom menus only work in FMP 8 and later

SecureFM gives you a powerful set of additional functions that include:

	SecureFM	FMP Custom Menus
Hot Keys	Remap any key — with or without a modifier like Cmd or Ctrl — to call your own custom scripts!	Keyboard shortcuts can be created for custom menu items, but they are much more limited. They are not dynamic by context and they require that each keyboard shortcut be linked to a menu item. That also means that custom menus must be changed every time you want to change the active keyboard shortcuts. Also, FMP's keyboard shortcuts can only be assigned with modifiers (Cmd or Ctrl).
Trigger Scripts	Trigger a script any time a user changes the file, layout, window, or exits a field In fact, SecureFM can be used to leverage the functionality of FMP's custom menus, by enabling a user to change the custom menus when the context changes. Works with FileMaker 7 through 11.	Script triggers only exist in FileMaker 10 & 11
Remove the Entire Menubar	With a single command, remove the entire FileMaker menu bar, disable all command keys, and disable window buttons like close, maximize, and minimize.	Nothing equivalent

Using SecureFM with Custom Menus

If you choose to, you can use both custom menus and SecureFM at the same time. You can, for example, use FileMaker’s built-in capabilities to create custom menus, but still use SecureFM to disable menu items under certain circumstance, set up hot keys, etc.

Using SecureFM with Runtime Solutions

SecureFM can be used to add an increased layer of security for your stand-alone FileMaker Runtime solutions.

Creating a Plug-Ins Folder for Runtime Solutions

In a normal (non-runtime) FileMaker solution, plug-ins must be placed in the FileMaker “Extensions”. The same is true for runtime solutions, but you may need to manually create these folders.

To create a plug-ins folder for your runtime solution:

- Within the folder containing your runtime solution, create a new folder named “Extensions”.
- Then place your copy of SecureFM into the new folder.
- When you relaunch your runtime solution, check the plug-in preferences in the Edit menu. Make sure SecureFM is listed among the plug-ins, and that it is marked as enabled.

Menu Differences in Runtime Solutions

Remember that SecureFM disables FileMaker's menu items based on their numeric location, not by name. Since some menus in a stand-alone runtime solution are different from standard FileMaker menus, a configuration designed to run on a standard FileMaker database often will not modify the same items in a runtime solution.

Just as with different versions of FileMaker Pro and FileMaker Developer, runtime files can require a menu definition string unique to its menus. You can calculate the correct string using the SecureFM Menu Editor file or by manually adjusting strings created for other versions of FileMaker menus.

Menus Affected by the Runtime Binding Process

Four menus get modified when turning your FileMaker solution into a stand-alone runtime solution: File, View, Scripts, and Help. If your SecureFM string does not modify any of these menus, then it should not require any modification to work in your runtime solution.

- File Menu
  Many items get removed from the File menu in runtime solutions, leaving users with many fewer functions.

- View Menu
  The View menu remains identical, except that the Layout Mode menu item is completely removed. This changes the item numbers for all subsequent items in the menu, reducing their item number by one.

- Scripts Menu
  In runtime solutions, the ScriptMaker menu item (and the subsequent divider) get removed, so you only see the displayed scripts.

- Help Menu
  The FileMaker-specific Help menu items get removed. If, during the binding process, you specified a script to act as a Help script, the script's name will be added to the bottom of the Help menu items.

If your SecureFM configuration affects one of these four menus, the menu items may need to be adjusted to work properly on the runtime solution's menus. Remember that the menu items are counted from top to bottom. (And don't forget that the horizontal menu dividers count as menu items.)

Troubleshooting

If you are encountering unexpected results when using SecureFM with your FileMaker solution, try the following troubleshooting steps to help pinpoint the source of the problem.

- Make sure there is only one version of SecureFM installed

Having two different versions of SecureFM installed at the same time may cause problems. Delete any earlier versions of SecureFM. Simply dragging the new plug-in into the correct folder will not assure that the old version will be replaced because their names may be different.

- Make sure SecureFM is installed correctly

If SecureFM appears to not be doing anything, make sure the plug-in is installed correctly.

Place the SecureFM plug-in into the FileMaker “Extensions” folder inside your FileMaker Pro folder.

Restart FileMaker after installing the plug-in.

Make sure SecureFM is selected in the Plug-Ins section of the Application Preferences (found in the Edit Menu on Windows and in the FileMaker menu on Mac OS X).

- Restart FileMaker

If you are using an unregistered version of SecureFM, it ceases to function 60 minutes after FileMaker Pro is launched. To restore SecureFM functions before you are ready to purchase a user license, you must restart FileMaker Pro.

- Check the Registration Status

Make sure that SecureFM is actually installed and registered. You can do this by attempting to re-register SecureFM:

```
Set Field ["Registration Response Field"; "External ("Menu-Register";
    "LicenseeName|MacRegistrationCode|WinRegistrationCode")"]
```

If the response in the Registration Response Field is blank, SecureFM is not installed; in which case, none of the SecureFM functions will work properly. You should first make sure that the SecureFM plug-in is placed into the Extensions folder found within the FileMaker folder. After you have done that, go to the Application Preferences. Select the Plug-Ins tab and make sure that SecureFM is in the list and checked.

If the Registration Response Field is showing an "Invalid Registration", SecureFM is running in Demo mode. Double check your registration codes if you have already purchased a user license.

MenuMagic Registration Note: When the plug-in is registered with a basic SecureFM license key, the core security functions (Menu-Disable, Menu-Toolbar, etc.) will work fine, but the MenuMagic functions (Menu-Rename, Menu-Script, Menu-Event, etc.) will run in demo mode for only 1 hour from launch. If some of the plug-in functions appear to be working properly but the new MenuMagic functions are not, make sure that you are registering the plug-in with the correct license key.

- Check the SecureFM response.

When an error occurs, SecureFM will normally return a response in your Set Field step. You can pause your script and visually check SecureFM's response before proceeding.

For example:

```
Set Field ["Response Field", "External ("Menu-Disable", "<*>:<1,B,<1,2,3>>
 <7,B,<1>>"]
Pause/Resume Script
```

Or, build an error check directly into your script:

```
Set Field ["Response Field", "External ("Menu-Disable", "<*>:<1,B,<1,2,3>>
 <7,B,<1>>"]
If [not IsEmpty (Response Field)]
  Beep
```

```
        Show Message ["An error has occurred."]
        Halt Script
    End If
```

- When attempting to configure FileMaker's menus, SecureFM gives the response "Incorrect Password".

This indicates that there is an inconsistency between the password used when registering SecureFM and the password used in your current SecureFM command. This can happen in the following situations:

You attempted to re-register SecureFM with a new password after already registering SecureFM with a different password.

You must first clear the password field, quit FileMaker and then relaunch the SecureFM Menu Editor file before you can enter a new password.

(The only time you can register with a new password without having to first restart FileMaker is if you initially registered without a password and later want to register with a password.)

You attempted to configure FileMaker's menus or toolbars, but the password in the Menu-Disable, Menu-Rename, Menu-Script, or Menu-Toolbar step does not match the password used in the Menu-Register step.

Make sure the password is at the end of the configuration string and that it matches the password used in the Menu-Register step.

(If 'Incorrect Password' appears twice in SecureFM's response, it is probably because you are attempting use multiple functions in the same step and both have the incorrect password.)

For more information about SecureFM's password capabilities, see the Password section of the SecureFM documentation.

If you have further questions about SecureFM, please visit our SecureFM FAQs web page at <http://www.newmillennium.com>

New Millennium
COMMUNICATIONS

New Millennium Communications is a software development company located in Boulder, Colorado. We specialize in making tools and advanced templates for FileMaker Pro developers.

New Millennium Communications
1332 Pearl Street
Boulder, CO 80302 USA
303-444-1476

www.newmillennium.com plug-ins@nmci.com