MenuControl 4

FileMaker Plug-In Manual

Dacons

This tutorial assumes that you have elementary knowledge of FileMaker Pro and you know how to use plug-ins and external functions of this database platform.

Please send feedback to info@dacons.net

Website: http://www.dacons.net

This version of the manual was updated on June 14, 2010.

CONTENTS

FUNCTION OVERVIEW

MenuControl 4 Features 6
    Function Overview 6

INTRODUCTION

Getting Started 7
    About this manual 7
    The MenuControl concept 7
    Default FileMaker items and custom menus items 8
    Software requirements for Windows 8
    Software requirements for Macintosh 9
    Installing the FileMaker MenuControl plug-in 9

CHAPTER 1

Creating a Custom Menu 10
    Planning the menu structure 10
    Creating scripts for custom menu items 11
    Creating two global text fields 12
    Creating one script to set the custom menu 13
    Additional parameters for custom menus 14
    The menu code 15
    The MenuComposer 17
    Setting custom menus in the start script 17
    Keyboard shortcuts 18
    Menu item properties 18
    Changing menu items 19
    Adding and removing menu items 21

CHAPTER 2

Customizing the FileMaker Menu 23
    Using default FileMaker menu items 23
    Default FileMaker menu items and localizations 25
    Behavior of default FileMaker menu items 25
    Mixed menus 26

CHAPTER 3
Contextual Menus 27
- Standard Contextual Menus 28
- Contextual menus for fields or hot zones 29
- Removing and restoring contextual menus 31

CHAPTER 4
Pop-up Menus 33

CHAPTER 5
Alternative Menus 35
- Alternative Main Menus 36
- Alternative Context Menus 36
- Restoring Alternative Menus 36

CHAPTER 6
Priority Order of Menus 38
- Main menus priority order 38
- Context menus priority order 39

CHAPTER 7
Additional Security Features 41
- Disable window control buttons 41
- Remove toolbars 41
- Make ThinClient 42

CHAPTER 8
Important Developer Guidelines 43
- End-user access 43
- Securing a MenuControl solution 44
- Disable smart quotes 44
- FileMaker Preferences Access 45
- Unique menu names 45
- Help Menu under Mac OS 45
- Application Preferences menu under Mac OS X 45
- Runtime solutions and MenuControl 46
- MenuControl in a network environment 46

CHAPTER 9
Additional Developer Guidelines 47
- Menu structure basics 47
- MenuControl in multi-file solutions 47
- Saving a changed menu code 48
- Menu code debugging 49
- Using Auto Update with FileMaker Server 49

Legal FileMaker Acknowledgements 49

CHAPTER 10

Using MenuControl Functions 50
- Accessing external functions 50
- Function parameters 51
- Mandatory and optional parameters 52
- Result codes 52

CHAPTER 11

External Functions Reference 54
- Cont_Version 54
- Cont_BuildMenu 55
- Cont_BuildContextMenu 57
- Cont_RestoreMenu 59
- Cont_RestoreContextMenu 60
- Cont_RemoveFMContextMenus 61
- Cont_BuildAltMenu 62
- Cont_BuildAltContextMenu 64
- Cont_RestoreAltMenu 66
- Cont_RestoreAltContextMenu 67
- Cont_ChangeItem 68
- Cont_AddItem 69
- Cont_RemoveItem 70
- Cont_BuildPopupMenu 71
- Cont_ControlToolsBars 72
- Cont_SecureWindow 73
- Cont_MakeThinClient 74
- Cont_SetErrScript 75
- Cont_GetItemString 76
- Cont_GetLastScript 77
- Cont_GetMenuCode 78
- Cont_FreezeMenu 79
- Cont_RefreshMenu 80
- Cont_RegisterSession 81

FUNCTION OVERVIEW

MenuControl 4 Features

MenuControl lets you create your very own **main menus, context and pop-up menus** including custom keyboard shortcuts.

The powerful alternative menu function enables you to change the default FileMaker menu, which is used for *all* files on a particular workstation. Develop user friendly and secure FileMaker solutions with unsurpassed ease and flexibility.

Function Overview

- Create your own custom menu including submenus or
- Modify the default FileMaker menus to your needs
- Create mixed menus by combining the default FileMaker menu items with your own custom menus items
- Install different menus for different files, layouts and modes only once and always have the correct menu shown automatically
- Create your own keyboard shortcuts for your menu items
- Create pop-up menus
- Create custom contextual (right-click) menus or disable them completely
- Change menu items on the fly
- Disable/enable menu items according to the situation
- Toggle toolbars on/off (not available for Mac OS X under FileMaker 6/5.x)
- Disable the file window control buttons (Close, Maximize/Restore and Minimize)
- ThinClient function lets you remove the complete menu
- Alternative menu function lets you customize the default FileMaker menu for all files on a particular computer
- Create the menus with the help of the MenuComposer as easy as point-and-click
- Full cross-platform compatibility
- Full control over user navigation and access
- Added security to your solution

INTRODUCTION

Getting Started

This chapter describes how to use this manual and explains the concept of MenuControl for FileMaker. In addition, it provides all information you need to install the plug-in.

About this manual

This manual is written in an easy-to-follow style. If you are new to MenuControl you should read this introduction and **Chapter 1** (→ Creating a Custom Menu, p.10) before proceeding with any other section of the manual. Once you understood the basics of MenuControl you can skip to any chapter and focus on the information you currently require.

Professional FileMaker developers should also read the chapters → **Important Developer Guidelines** (p.43) and → **Additional Developer Guidelines** (p.47). The chapter → **External Functions Reference** (p.54) provides an in-depth reference for all plug-in functions. However, if you are new to FileMaker plug-ins you should read the chapter → **Using MenuControl Functions** (p.50) before.

You will find examples in most sections of this manual. Such examples are demonstrated in the *Quick Start* file that comes with the MenuControl download package. Whenever an example can be found in the *Quick Start* file you will see the icon on the right.

Quick Start

> Important hints and rules of thumb which you find in this manual are highlighted like this box.

The MenuControl concept

Implementing your own menus is very simple. In general you can create your own menus in four simple steps using:

1 the MenuComposer to create your custom menus via simple point-and-click.

2 one global text field in your solution for the result of the plug called **Result Field**. The result field indicates if the function succeeded or failed.

3 one global text field in your solution that holds the menu code: **Menu Code Field**.
4 one script to set the custom menu upon start-up of your solution.

The MenuComposer is a simple point-and-click style application that lets you create and preview your custom menus. The MenuComposer writes the menu code for you. The menu code is needed by the plug-in to set a custom main menu, contextual menu or pop-up menu. It contains vital information about menu items, their status, keyboard shortcut and scripts assigned.

Only **two** global fields, the menu code and a single plug-in function are needed to set your own menu.

Default FileMaker items and custom menus items

In general you can use two different types of menu items with MenuControl:

1. **Default FileMaker menu items:**
   Default FileMaker menu items are all menu items available by default from FileMaker.

2. **Custom menu items:**
   Custom menu items trigger your own FileMaker scripts. It is important to understand that for custom menu items scripts have to be created first.

You can use a mix of those two types of menus items in your solution within one menu.

With MenuControl you can use *any* script as menu item. And you only need to include those FileMaker menu items into your solution that are relevant to your specific application.

More details about how to create a custom menu consisting of scripts and FileMaker menu items can be found in the chapters → **Creating a Custom Menu** (p.10) and
→ **Customizing the FileMaker Menu** (p.23).

Software requirements for Windows

FileMaker Pro 5.0 or higher, including FileMaker Pro 6.x, FileMaker 7 (Pro, Developer or Runtime), FileMaker 8-11 (Pro, Advanced or Runtime) on the following Windows platforms: Windows XP, Windows 2000 and Windows 98.

When running MenuControl 4 under Windows 2000 and FileMaker **7-11** you need Service Pack 4 for Windows 2000.

Software requirements for Macintosh

FileMaker Pro 5.0 or higher, including FileMaker Pro 6.x, FileMaker 7 (Pro, Developer or Runtime), FileMaker 8-11 (Pro, Advanced or Runtime) on the following Mac platforms:
Mac OS 9.2 or higher including Mac OS 10.2 and higher.

Mac Classic (OS 9.2) needs the “CarbonLib” to run MenuControl 4. CarbonLib 1.6 is available at: http://docs.info.apple.com/article.html?artnum=120047

Installing the FileMaker MenuControl plug-in

Before installing the MenuControl plug-in ensure your select the correct version from the download package for your operating system and FileMaker version.

In addition, ensure that you delete *all* previous versions of MenuControl installed in your system.

MenuControl comes in six versions:

- Windows for FileMaker 5-6
- Windows for FileMaker 7-11 (Unicode)
- Windows for FileMaker 7-11 (Classic)
- Macintosh for FileMaker 5-6
- Macintosh for FileMaker 7-11 (Unicode)
- Macintosh for FileMaker 7-11 (Classic)

Place the MenuControl plug-in in the following folder:

Macintosh:
FileMaker 5-6 in the *FileMaker Extensions* folder
FileMaker 7-11 in the *Extensions* folder

Windows:
FileMaker 5- 6 in the FileMaker *System* folder
FileMaker 7-9 in the FileMaker *Extensions* folder

The folder you have to look for is located in the FileMaker application folder.

CHAPTER 1
Creating a Custom Menu

In the following a custom main menu is created using only our own scripts as menu items from scratch.

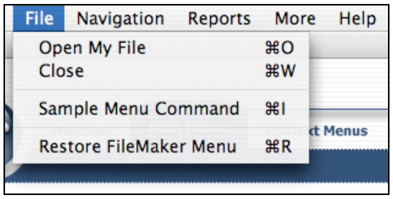
Planning the menu structure

Since you can have different menus for different files and/or layouts, it is a good practice to write down the menu structure of your project including the keyboard shortcuts ahead of time. This will speed up the development process and will eliminate errors.

Here is a simplified outline of a main menu that can be found in the *Quick Start* file. Click on the *Main Menus* tab and choose the button *Custom Menu* to set the main menu that is based on the following structure.

File	Navigation	Reports	More	Help
Open My File	Continue	Report 2004	Custom Menus	MenuControl on the Web
Close	Go Back	Report 2003	Include Keyboard Shortcuts	
-	-	Report 2002	And Checkmarks	
Sample Menu Command	Welcome	Older Reports	And Sub Menus	
-	Main Menus			
Restore FileMaker Menu	Contextual Menus			
	Pop-up Menus			
	Change Items			

The following screenshot shows how the final menu looks like in FileMaker.

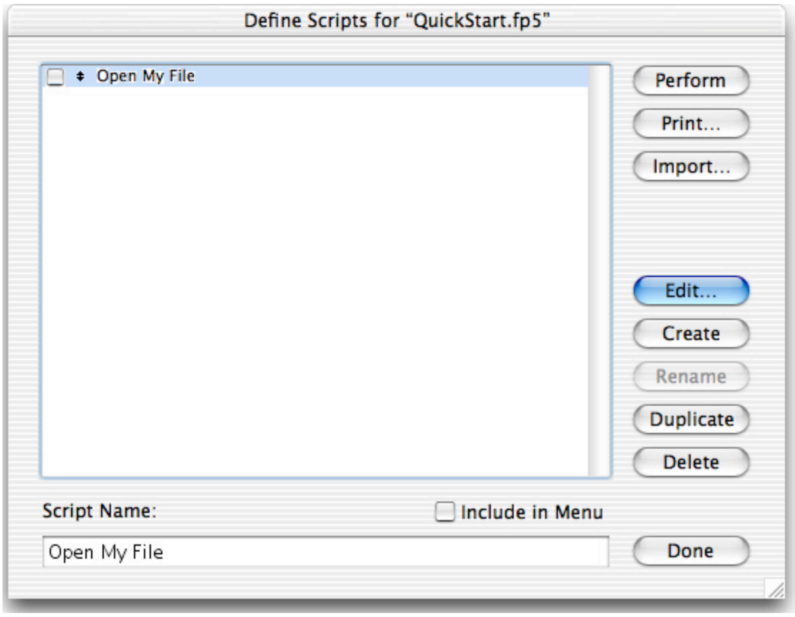


Custom main menu in FileMaker

Creating scripts for custom menu items

Once the menu structure is planned it is time to create the appropriate scripts for the various custom menu items. Remember, custom menu items trigger your scripts.

Example:
The custom menu item *Open My File* should trigger a certain script that needs to be created. In the ScriptMaker a script called *Open My File* is created. Please note that menu item can have a different name than your script.



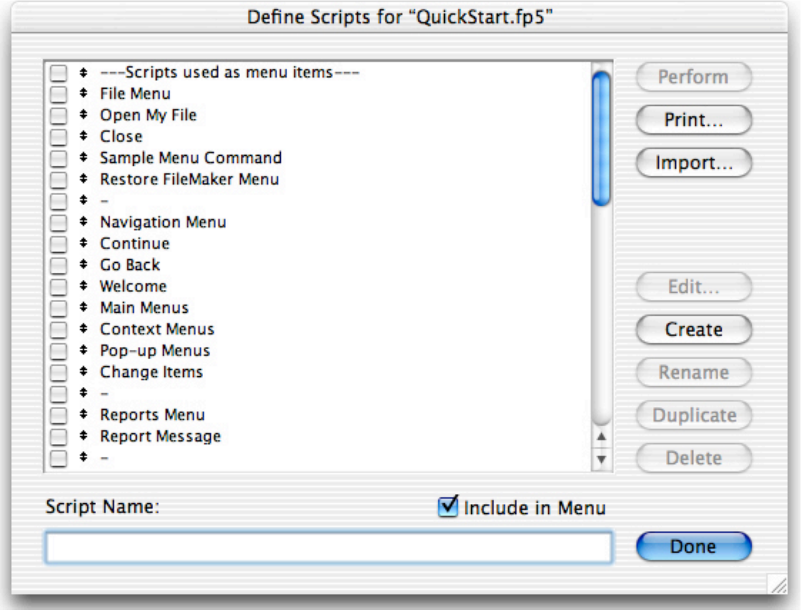
Script Name: Open My File

☐ Include in Menu

ScriptMaker showing the script 'Open My File'

In this case the script *Open My File* will simply show a message for demonstration purposes.

In the same manner scripts for all custom menu items are created.



ScriptMaker showing all scripts needed for the menu

Note: The first step is to create the scripts that your menu item will trigger.

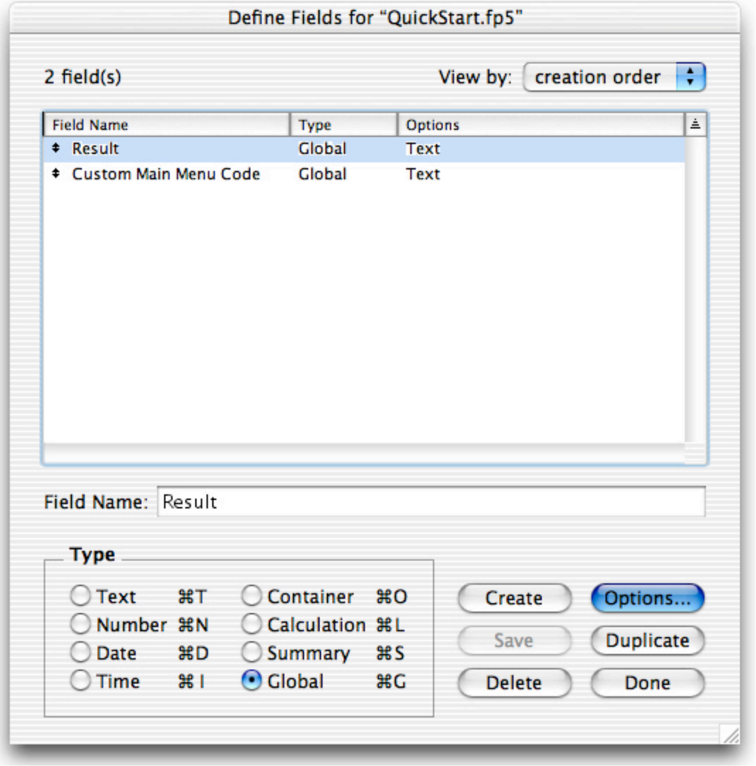
Creating two global text fields

The next step is to create two new fields, for this tutorial they are called:

1. **Result**: Defined as global text field.
   This field will contain the result code that is returned by the MenuControl plug-in. It gives an indication whether the operation succeeded or failed.

2. **Custom Main Menu Code**: Defined as global text field.
   In order for the plug-in to set your menu structure it has to read the menu code, which will be placed in this field. The menu code contains the structure of the menu code such as:
   - Main menus and cascading sub menus
   - Menu items
   - Assigned scripts to menu items
   - Assigned status of the menu items (enabled/disabled)
   - Assigned keyboard shortcuts for the menu items etc.

The MenuComposer will create the menu code for you according to the menu structure you create.



Define Fields for “QuickStart.fp5”

2 field(s)

View by: creation order

Field Name	Type	Options
Result	Global	Text
Custom Main Menu Code	Global	Text

Field Name: Result

Type
○ Text ⌘T
○ Number ⌘N
○ Date ⌘D
○ Time ⌘I
○ Container ⌘O
○ Calculation ⌘L
○ Summary ⌘S
● Global ⌘G

Two global text fields defined in FileMaker 6

Note: Details on how to create global text fields can be found in the FileMaker Online Help.

It is suggested placing the two global text fields in a new layout (named *Admin* or *Developer*). The end-user should not have access to this layout because it will contain the data that is used by the plug-in to set the menu for your solution. Please refer to the section → **Additional Security Features** (p.41) for more details on how to secure your FileMaker solution with MenuControl.

Creating one script to set the custom menu

In order for MenuControl to set a custom main menu, the external function *Cont_BuildMenu* is used.

1 Open the ScriptMaker and create a new script – in this example it is called: *Set Complete Custom Main Menu*

2 Use the script command *Set Field*. As field select *Result*.

3 Click the *Specify* button and chose the *Cont_BuildMenu* external function, select the *Custom Main Menu Code* field and as parameters give the menu a name. In this case it is called *CustomMainMenu*.

If you are unfamiliar on how to work with FileMaker's *Set Field* script step please consult the chapter → **Using MenuControl Functions** (p.54).

The final function as part of a *Set Field* script step is explained in the following table. It will set a custom menu for the entire file.

Function Call	Explanation
`Cont_BuildMenu(`	Plug-in function to set a custom main menu
`Custom Main Menu Code;`	Field containing the menu code followed by a parameter separator
`"CustomMainMenu"`	Name of the custom main menu; must be unique
`)`	End of the plug-in function call

Please note that in FileMaker 7 and higher the name of the table occurrence is part of a field name in calculations. Thus, in FileMaker 7 and higher *Custom Main Menu Code* would be *Quick Start::Custom Main Menu Code*.

Additional parameters for custom menus

Cont_BuildMenu has additional parameters that can be used if your solution requires different sets of menus. Utilizing those parameters will allow you to set the menus only once, within the start-script of your solution and MenuControl will always display the correct menu for different layouts, modes or files.

The following shows the function with additional possible parameters.

```
Cont_BuildMenu ( Menu Code ; Menu Name ; Layout Name ; Mode ; File
Name )
```

Parameter	Explanation
`Menu Code`	**Mandatory:** Menu code to describes the contents of a custom menu.
`Menu Name`	**Mandatory:** Unique menu name of the custom menu; used to change the menu later.
`Layout Name`	**Optional:** Specify a layout name using this parameter. If the value is empty the menu will be attached to all layouts of the file.
`Mode`	**Optional:** Specify a FileMaker user mode ("Browse", "Find" or "Preview") to attach the menu only to a specific mode. Leave this parameter empty to attach the menu to all modes.
`File Name`	**Optional:** Specify the name of a file using this parameter if you would like to attach the menu to a file other than the current one. Leave this parameter empty to attach the menu to the current file.

Note: If multiple main menus are installed for one solution via start-script it is important to know that the order of priority is higher the more parameters have been defined for a menu. For more details refer to the chapter → **Priority** Order of Menus (p.39).

Parameters must be separated by the semicolon characters (";"), optional parameters at the end can be omitted. The first two parameters of the *Cont_BuildMenu* plug-in function are mandatory. All other parameters are optional.

To set the menu for a certain layout, you have to include the name of the layout in addition to the *Menu Name* parameter:

```
Cont_BuildMenu ( MainMenuCodeField ; "MyMenuName" ; "LayoutA")
```

A start script where different menus for different layouts are being set could look like this within *Set Field* script steps:

- `Cont_BuildMenu ( MenuCodeFieldA ; "MyMenuNameA" ; "LayoutA")`
- `Cont_BuildMenu ( MenuCodeFieldB ; "MyMenuNameB" ; "LayoutB")`
- `Cont_BuildMenu ( MenuCodeFieldC ; "MyMenuNameC" ; "LayoutC")`

If your menus will not change when the solution is used by end-users there are not more MenuControl functions you need to utilize. Small menu change at runtime such as adding or removing certain commands can be achieved using function like *Cont_AddItem*, *Cont_RemoverItem* or *Cont_ChangeItem*. More details on this can be found in the sections → **Changing menu items** (p.19) and → **Adding and removing menu items** (p.21).

It is recommended that you create a script using *Cont_RestoreMenu* before you set any custom menu for your solution. It is suggested that you make a button in your *Developer* layout ensuring that you can reset the default FileMaker menu at any time to work on your solution.

```
Cont_RestoreMenu
```

For more details, please refer to the *Restore FileMaker Menu* script of the *Quick Start* file.

> ***Note:*** The *Cont_BuildMenu* function has additional parameters that allow you to set different main menus for different layouts, modes and files only once. This will allow automatic updates of different menus when switching layouts, files or modes.

The menu code

MenuControl's *Cont_BuildMenu* function needs a menu code that contains all necessary information about your custom menu like:

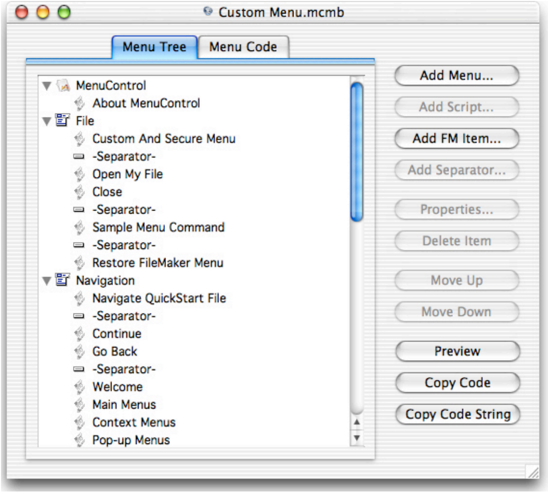
- Main menu items (such as *File* and *Edit*)
- Menu items that scripts or represent default FileMaker items
- Custom menu items that trigger scripts

- Default FileMaker menu items
- Submenu items
- Keyboard shortcuts
- Separators
- Status of a particular menu item (items can be enabled or disabled)
- etc.

You do not have to understand the menu code in detail. The MenuComposer will generate the menu code automatically according to the menu you create using a simple point-and-click interface.

Quick Start

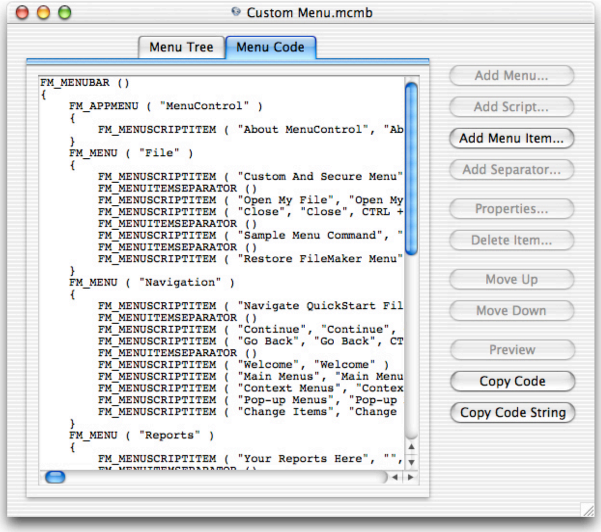
The following screenshot shows how our menu structure for the *Quick Start* file looks like in the MenuComposer.



Menu structure in the MenuComposer

The MenuComposer generates the menu code according to the menu structure you define by simple point-and-click. It should be place in the *Custom Main Menu Code* field and the plug-in's external function *Cont_BuildMenu* will read this code and will set the menu accordingly.

This is a view at the MenuComposer with the menu code visible (switch to the *Menu Code* tab to see it).



```
FM_MENUBAR ()
{
    FM_APPMENU ( "MenuControl" )
    {
        FM_MENUSCRIPTITEM ( "About MenuControl", "Ab
    }
    FM_MENU ( "File" )
    {
        FM_MENUSCRIPTITEM ( "Custom And Secure Menu"
        FM_MENUITEMSEPARATOR ()
        FM_MENUSCRIPTITEM ( "Open My File", "Open My
        FM_MENUSCRIPTITEM ( "Close", "Close", CTRL +
        FM_MENUITEMSEPARATOR ()
        FM_MENUSCRIPTITEM ( "Sample Menu Command", "
        FM_MENUITEMSEPARATOR ()
        FM_MENUSCRIPTITEM ( "Restore FileMaker Menu"
    }
    FM_MENU ( "Navigation" )
    {
        FM_MENUSCRIPTITEM ( "Navigate QuickStart Fil
        FM_MENUITEMSEPARATOR ()
        FM_MENUSCRIPTITEM ( "Continue", "Continue",
        FM_MENUSCRIPTITEM ( "Go Back", "Go Back", CT
        FM_MENUITEMSEPARATOR ()
        FM_MENUSCRIPTITEM ( "Welcome", "Welcome" )
        FM_MENUSCRIPTITEM ( "Main Menus", "Main Menu
        FM_MENUSCRIPTITEM ( "Context Menus", "Contex
        FM_MENUSCRIPTITEM ( "Pop-up Menus", "Pop-up
        FM_MENUSCRIPTITEM ( "Change Items", "Change
    }
    FM_MENU ( "Reports" )
    {
        FM_MENUSCRIPTITEM ( "Your Reports Here", "",
```

Menu code generated by the MenuComposer

> ***Note:*** The menu code is needed by the MenuControl plug-in to set your custom menu. It will be generated by the MenuComposer automatically according to the menu structure you create using simple point-and-click.

The MenuComposer

The MenuComposer tool which is also part of the download package helps you to create the menu and generates the menu code as shown earlier.

The MenuComposer will automatically create the menu code according to your menu structure. Please refer to the MenuComposer manual for further details.

Setting custom menus in the start script

When creating a custom menu for your FileMaker Pro solution or runtime application chances are that you would never like the complete default FileMaker Pro menu to appear for the end user.

It is recommended setting a custom menu with the *Cont_BuildMenu* function upon startup of your solution. You can set a start-script as follows.

FileMaker 6/5:
FileMaker Preferences > Document > General Tab
Look for the line: When opening MySolution > Perform script

FileMaker 7 and higher:
File > File Options > Open/Close tab > When opening this file > Perform script

In addition you do not want users to have access to your solution without the MenuControl plug-in installed. It is recommended having a check in your start-script to make sure MenuControl is installed before your solution really opens.

To find out more about this and other important scripting hints refer to the chapter
→ **Important Developer Guidelines** (p.43).´

Keyboard shortcuts

MenuControl allows you to define your own keyboard shortcuts for your custom menu items. In general the keyboard shortcut combinations *Ctrl* and *Alt* for Windows and *Cmd* and *Option* for Macintosh as well as F-keys (Function keys) are available.

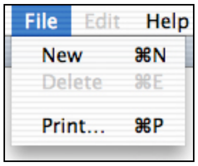
A list of available keyboard shortcuts can be found in the MenuComposer manual.

Menu item properties

MenuControl allows you to display different conditions for each menu item.

Disabled menu items

Menu items can be shown as disabled. A disabled menu item is grayed-out and not functional. This is an additional security feature, giving you the possibility of disabling certain menu items in certain situations in your solution. This is possible for custom menu items *as well as* default FileMaker menu items.

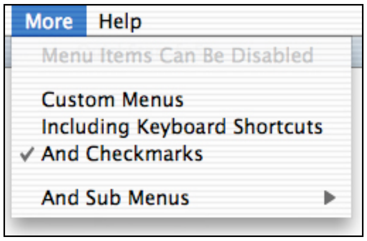


Disabled Delete command

The plug-in function *Cont_ChangeItem* can be used to modify (i.e. disable) menu items. For details please refer to the section → **Changing menu items** (p.19).

Menu item checkmark

Custom menu items can have checkmarks on both Windows and Mac OS.
This is possible for custom menu items only, not for default FileMaker menu items.



Custom menu item with checkmark

Checkmarks can be used in menus to indicate that a certain option is turned on or active.

Changing menu items

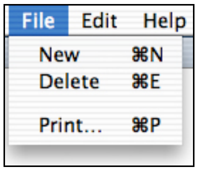
Your FileMaker solution should have one main menu structure that makes it easy for the end user to find the way around your application.

However, a user-friendly menu should be customized automatically according to the situation. Customization can increase security, if you deny the user access to certain menu items in certain situations.

Instead of creating a whole new menu code for each situation MenuControl allows you to change, add or remove single menu items via external functions.

In this example the *Delete* menu item in the *File* menu as seen in the *Quick Start* file under the *Change Items* tab is being disabled.

Quick Start



Menu item Delete is enabled

The *Delete* menu item was disabled with a simple script. The same script with a different parameter can enable the menu item again.

The function used to change the condition of the *Delete* menu item is called *Cont_ChangeItem.*

Note: You can change the condition of menu items anytime with *Cont_ChangeItem*.

Determine the location of a menu item

To make use of the function *Cont_ChangeItem* the location of the menu item that will be changed has to be determined. MenuControl has a simple internal index system for the location of menu items.

The menu that contains the *Delete* menu item has been called *ChangeItemMainMenu* when it was set using the function *Cont_BuildMenu*. Thus, the location of the *Delete* menu item is **ChangeItemMainMenu:1>2** (*File* is the first main menu item) and *Delete* is the second menu item under *File*.

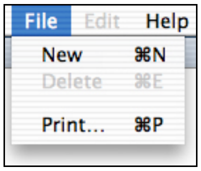
> ***Note:*** Before changing a menu item, its location within a menu has to be identified first.

The following function is used to change a menu item as part of a *Set Field* script step.

Function Call	Explanation
`Cont_ChangeItem (`	Plug-in function to change menu items
`"ChangeItemMainMenu:1>2";`	Location of the menu item in the menu followed by a parameter separator
`"FM_MENUSCRIPTITEM`	Type of menu item
`(""Delete"",`	Name of menu item. Double quotation needed if the item code is used directly in the calculation editor.
`""Sample Delete Command"",`	Name of the script which the menu item triggers
`CTRL + ""E"",`	Keyboard shortcut
`Disabled )"`	Status of the menu item (in this case disabled)
`)`	End of plug-in function call

Please note that the double quotation marks shown above are needed if the menu code of the menu item to be changed is inserted directly into the calculation editor. In the example above a custom script item is set. Thus, the tag *FM_MENUSCRIPTITEM* is used. To set a default FileMaker menu item use the tag *FM_MENUITEM* instead.

Once the script *Disable Item Delete* is called, the *Delete* menu item will be disabled.



Menu item Delete is disabled

In the same way the menu item can be enabled again by using *Enabled* instead of *Disabled* when calling the function above.

> ***Note:*** Separators count as menu items in the menu index system.

The following menu structure will be used to explain the internal index system to determine the location of a menu item in more detail. The name of this custom menu which was set when it was installed using the plug-in function *Cont_BuildMenu* is *ChangeItemMainMenu*.

Menu Item	Item Index
File	`ChangeItemMainMenu:1`
New	`ChangeItemMainMenu:1>1`
Delete	`ChangeItemMainMenu:1>2`
Separator	`ChangeItemMainMenu:1>3`
Print	`ChangeItemMainMenu:1>4`

Menu Item	Item Index
Edit	`ChangeItemMainMenu:2`
Undo	`ChangeItemMainMenu:2>1`
Separator	`ChangeItemMainMenu:2>2`
Cut	`ChangeItemMainMenu:2>3`
Copy	`ChangeItemMainMenu:2>4`
Paste	`ChangeItemMainMenu:2>5`
Clear	`ChangeItemMainMenu:2>6`

According to this index system the *Copy* menu item under *Edit* has the following location: ChangeItemMainMenu:2>4

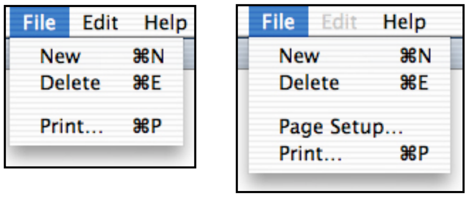
Please note that the Application Menu item under OS X has the following index in this case: **ChangeItemMainMenu:0**

Adding and removing menu items

MenuControl allows you to add or remove menu items on the fly according to the situation in your solution. In certain situations it might make sense to add or remove menu items.

Adding a menu item

In this example a *Page Setup* menu item is added in the *File* menu as seen in the *Quick Start* file on the *Change Items* tab.



File menu before (left) and after (right) the 'Page Setup' item was added

To add a menu item on the fly the following plug-in function is used in a *Set Field* script step: *Cont_AddItem*

Determine the location where the new menu item should appear:
The *Page Setup* menu item in this example should be placed in the *File* menu where the current *Print* menu item is located.

According to the index system this would be *ChangeItemMainMenu:1>4*

(*File* is the first main menu item and *Print* the fourth menu item down. Remember, separators count as menu items.)

Function Call	Explanation
`Cont_AddItem (`	Plug-in function to add a menu item
`"ChangeItemMainMenu:1>4";`	Location at which the new item is to be inserted

`"FM_MENUITEM ( ""Page Setup…"",`
`FM_PRINT_SETUP )"`
	Menu code of the new item including item type (FM_MENUITEM), caption and item tag. Double quotation needed if the item code is used directly in the calculation editor.

`)`	End of the plug-in function call

After the *Page Setup* item is added, the *Print* item will move down one spot. *FM_PRINT_SETUP* is just one example of a default FileMaker menu item. Please review the *Menu Code Reference* paper for details about all supported FileMaker items.

Removing a menu item

To remove the *Page Setup* menu item from the example menu mentioned above the plug-in function *Cont_RemoveItem* is used in a *Set Field* script step as follows:

```
Cont_RemoveItem ( "ChangeItemMainMenu:1>4" )
```

CHAPTER 2

Customizing the FileMaker Menu

MenuControl gives you the freedom to customize the default FileMaker menu. Default FileMaker menu items will not trigger any scripts. They will be handled internally as default menu items. Scripts only have to be created for custom menu items.

Using default FileMaker menu items

If you decide to customize the default FileMaker menu, the MenuComposer preloads the set of default FileMaker menu items according to the FileMaker version you choose. You simply have to point-and-click to select the default FileMaker menu items you would like to delete from your menu structure.

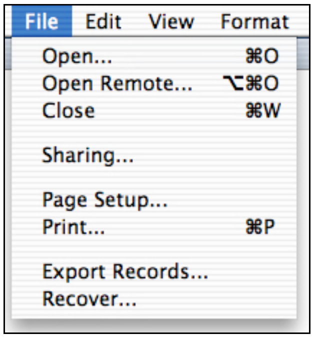
You can rename the default FileMaker menu items to your liking in the MenuComposer.

You can delete certain default FileMaker menu items and functions that could compromise the security of your solution and in addition will make your application more user-friendly.

Example:
The menu item *Define Fields* in the FileMaker 6 *File* menu or the menu item *Accounts and Privileges* in the FileMaker 7 *File* > *Define* menu are irrelevant for the end-user of your solution.

With MenuControl you can not only disable those menu items but remove them completely.

In the following example the default FileMaker *File* menu was scaled down using MenuControl.

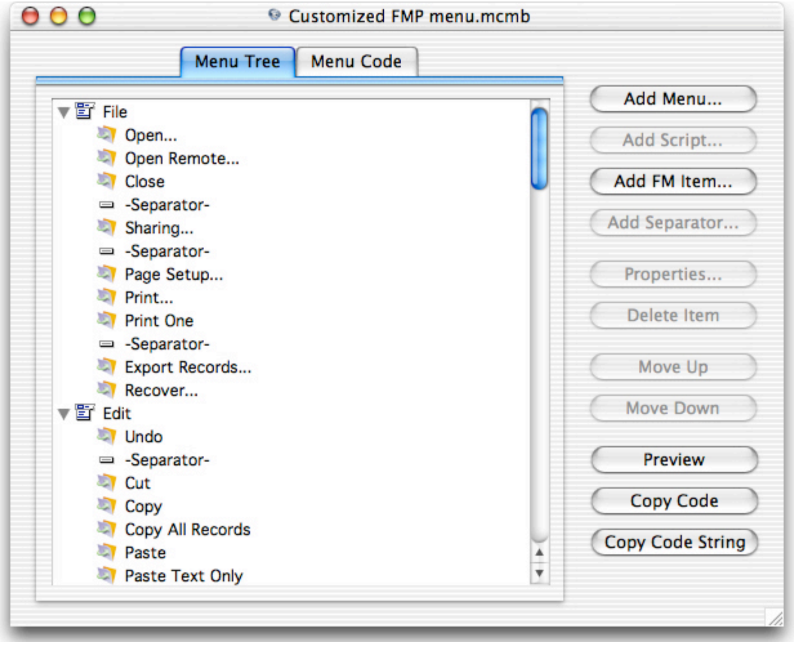


File menu that was scaled down using MenuControl

This screenshot was taken from the *Custom FileMaker Menu* which can be found on the *Main Menu* tab of the *Quick Start* file.

Some of the menu items that only apply to developers have been eliminated.
Of course, the choice is yours. This is only an example.

The menu code created by the MenuComposer for a customized default FileMaker menu is in principal the same as the menu code for a complete custom menu and can be used the same way in conjunction with the plug-in function *Cont_BuildMenu*.



Customized FileMaker File menu in the MenuComposer

In the example shown in the screenshot above menu items not needed simply have been deleted from the menu tree in the MenuComposer.

A step-by step instruction on how to create a custom default FileMaker Pro menu can be found in the MenuComposer manual.

To experience an example of a customized default FileMaker menu, open the *Quick Start* file and go to the *Main Menu* tab. Here, click the *Set Custom FileMaker Menu* button.

Once you created a customized FileMaker menu in the Composer, you simply need to place the menu code in your menu code field of your solution, which is used by the *Cont_BuildMenu* plug-in function.

> MenuControl lets you customize the default FileMaker menu to your needs.

Default FileMaker menu items and localizations

FileMaker 6/5 is available in different languages. FileMaker 7 and higher let's users switch between different user interface languages.

MenuControl shows the captions or FileMaker items always according to the current FileMaker language. However, MenuComposer enables you to define a fixed caption for a FileMaker item. If a fixed caption has been defined by disabling the option *Use local caption* for a FileMaker item the fixed caption will be shown for the item no matter which FileMaker language is active.

For more details please review the *MenuComposer 4 Manual*.

Behavior of default FileMaker menu items

When working with default FileMaker menu items the following points are important.

1. You can use the plug-in functions *Cont_AddItem*, *Cont_RemoveItem* or *Cont_ChangeItem* to add/remove or enable/disable default FileMaker menu items.

2. Some default FileMaker menu items such as *New Record* or *New Request* are only available in certain FileMaker user modes (Browse, Find or Preview). MenuControl will handle the default FileMaker menu items as they would be in the default FileMaker menu environment. Thus, the *New Request* menu item will only be available in Find mode; it will be disabled in the Browse mode.

3. Some default menu items are disabled in certain situations. Example: The *Format* menu is disabled when no text field is selected. This will not be different if the *Format* menu is included in a custom menu using MenuControl.

> Default FileMaker menu items maintain the same behavior as if they would be used in the default FileMaker menu environment.

Mixed menus

The flexibility of MenuControl enables you to create mixed menus.

The term *mixed menu* means that you can include default FileMaker menu items and custom menu items that trigger your scripts in one menu structure. Default FileMaker menu items can be used as is and no scripts have to be created for them. Or you can customize the default FileMaker menu by including your own menu items in the MenuComposer.

It is suggested looking at the *Quick Start* file. Under the section *Main Menu* a mixed menu structure has been created. There, the *Edit*, *Insert* and the *Format* main menu items are default FileMaker menu items and all other menu items were customized for this file. To create this mixed menu the MenuComposer preloads the set of default FileMaker menu items according to the FileMaker version you choose. You simply have to point-and-click to select the default FileMaker menu items and your custom menu items you would like to include into your menu structure.

The MenuComposer manual gives a step-by-step instruction on how to create mixed menus.

Once you have created a mixed menu in the Composer, you simply need to place the menu code in your menu code field of your solution, which is used by the plug-in function *Cont_BuildMenu*.

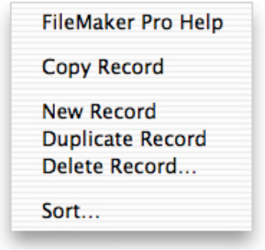
> MenuControl enables you to create mixed menu containing only the default FileMaker menu items that you want and your own menu items.

CHAPTER 3

Contextual Menus

A contextual (or context) menu is shown when the user right-clicks (Windows) or control-clicks (Mac OS). Just like with main menu items, contextual menus can consist of custom menu items and default FileMaker menu items.

The following shows a typical FileMaker default contextual menu as it appears in the Browse mode when right-clicking in a layout.



FileMaker context menu

With MenuControl you have the option to set different types of contextual menus.

A custom context menu can be set as the following:

- For an entire file
- For a specific layout
- For a specific area within a layout normally used for fields or other hot zones
- For a certain mode
- Or any combination of the above
- MenuControl can also remove *all* context menus from a file including the default FileMaker context menu

Multiple context menus for different layouts, fields or files can be active at the same time.

MenuControl manages them for you and always shows the right context menu when the user right-clicks (Windows) or control-clicks (Mac).

Just like with the main menu, different contextual menus can be set upon start-up of your solution.

> ***Note:*** If multiple menus are installed for one solution via start-script it is important to know that the order of priority is higher the more optional parameters have been defined when the menu was set. For more details refer to the chapter → **Priority Order of Menus** (p.39).

Standard Contextual Menus

In this section a context menu for an entire layout has been set. It will be shown everywhere in the layout when the user right-clicks (Windows) or control-clicks (Mac).

You can have a look at the example in the *Quick Start* file under the *Context Menu* tab.

Quick Start

The following fields are needed to set a contextual menu for a layout:

1. **Result**: Defined as global text field.
   This field will contain a result code returned by the plug-in.
   It gives an indication whether the operation succeeded or failed.

2. **Layout Context Menu Code**: Defined as global text field.
   This field will contain the menu code created by the MenuComposer that contains the structure of the context menu:
   - Menu items
   - Cascading sub menus of the context menu
   - Assigned scripts to menu items
   - Assigned status of the menu items (enabled/disabled)

Have a look at the script *Set Layout Context Menu* in the ScriptMaker of the *Quick Start* file. It contains the following plug-in function call within a *Set Field* script step.

Function Call	Explanation
`Cont_BuildContextMenu (`	Plug-in function to install a context menu
`Layout Context Menu Code;`	Name of the field containing the menu code followed my a parameter separator
`"LayoutContextMenu";`	Name of the context menu followed my a parameter separator
`"Context Menus"`	Name of the layout to which the context menu will be attached
`)`	End of plug-in function call

The function *Cont_BuildContextMenu* sets the contextual menu.

The menu code in the field *Layout Context Menu Code* was created using MenuComposer. In addition to the menu code a name of the context menu is passed to the plug-in. Menu names help to manage a specific menu when several have been set.

The name of the layout is set which enables the plug-in to show this context menu only for this layout.

To set a context menu for all layouts of a file, just omit the layout name from the function call.

Contextual Menus can be *attached* to layouts, hot zones in layouts (such as fields), FileMaker modes (Find, Browse, Layout) and entire FileMaker files using the following parameters:

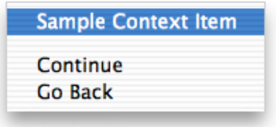
```
Cont_BuildContextMenu ( Menu Code ; Menu Name ; Layout Name ; Hot
Zone ; Mode ; File Name )
```

Please refer to the chapter → **External Functions Reference** (p.54) for more details about this external function and its parameters.

Making use of those parameters will ensure that always the correct contextual menu is shown – if you in fact use different contextual menus.

Set different menus upon start-up of your solution only once and always have the correct contextual menus for the current layout, mode and file appear.

Using the parameters *Layout Name*, *Mode*, *Hot Zone* and *File Name* is optional.



Custom context menu used in the Quick Start file

Contextual menus for fields or hot zones

MenuControl 4 lets you specify a context menu, which will show up only when the user right-clicks (Windows) or control-clicks (Mac) in a certain area of a specific layout.

This behavior is demonstrated with the second contextual menu example in the *Quick Start* file under the *Contextual Menu* tab. Have a look at the script *Set Field Context Menu* in the ScriptMaker of the *Quick Start* file.

The menu code of the second context menu is stored in the field called
Field Context Menu Code.

In this example a custom context menu should be shown when the user right-clicks the field *Context Menu Field*. Its size will define the boundaries of the hot or target zone where the custom contextual menu will be displayed.

The boundaries of this field and its position and are passed to the plug-in by adding the following parameter to the *Cont_BuildContextMenu* function call:

FileMaker 6/5:

```
FieldBounds(Status(CurrentFileName); Status(CurrentLayoutName);
"Context Menu Field")
```

FileMaker 7 and higher:

```
FieldBounds(Get(FileName); Get(LayoutName); "Context Menu Field")
```

The *Context Menu Field* may or may not be made visible to the end user in a layout. In your solution the *Context Menu Field* might be a text field that is accessible to the end user. In this case the field itself will be the target zone of the context menu. It might also be an invisible field that is placed on a hot zone of a layout (i.e. the image of a map) to pass the borders of this hot zone to the MenuControl plug-in.

This is demonstrated in the *Set Field Context Menu* script. The following function call is used in the *Set Field* script step.

FileMaker 7 and higher:

Function Call	Explanation
`"Cont_BuildContextMenu";`	Plug-in function to install a context menu
`Layout Context Menu Code & "|" &`	Name of the field containing the menu code followed my a parameter separator
`"FieldContextMenu" & "|" &`	Name of the context menu followed my a parameter separator
`Get(LayoutName)     & "|" &`	FileMaker function that passes the name of the current layout to the plug-in followed by a parameter separator
`FieldBounds(Get(FileName); Get(LayoutName); "Context Menu Field")`	FileMaker 7 function that passes the boundaries of the target field to the plug-in followed by a parameter separator
`)`	End of plug-in function call

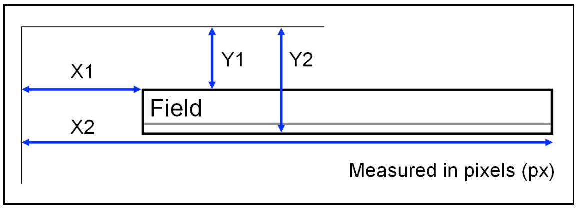
FileMaker 6/5:

Function Call	Explanation
`External(`	Beginning of the plug-in function call
`"Cont-BuildContextMenu";`	Plug-in function to install a context menu
`Layout Context Menu Code & "|" &`	Name of the field containing the menu code followed my a parameter separator
`"FieldContextMenu" & "|" &`	Name of the context menu followed my a parameter separator
`Status(CurrentLayoutName) & "|" &`	FileMaker 6/5 function to that passes the name of the current layout to the plug-in followed by a parameter separator

`FieldBounds(Get(FileName);`
`Get(LayoutName);`
`"Context Menu Field"`
	FileMaker 6/5 function that passes the boundaries of the target field to the plug-in followed by a parameter separator

`)`	End of plug-in function call

In the example above the FileMaker *FieldBounds* function is used to pass the location and size of the target field to the plug-in. MenuControl needs four values (X1, X2, Y1 and Y2) as shown in the following image. All of them are measured in pixels. *FieldBounds* provides these four values but they can also be set manually.



Hot zone for a context menu

Please note that due to a limitation in FileMaker contextual menus created for list or table views will only work for the first record. It is suggested to create one contextual menu for the whole layout of a list or table view layouts.

Removing and restoring contextual menus

The plug-in function *Cont_RemoveFMContextMenus* will remove all default FileMaker contextual menus. No right-click (Windows) or control-click (Mac OS) will appear if no custom context menus have been set for that file using MenuControl.

The default FileMaker contextual menu can be restored using the plug-in function *Cont_RestoreContextMenu*.

More details about this and other MenuControl functions can be found in the chapter → **External Functions Reference** (p.54).

The plug-in functions *Cont_ChangeItem*, *Cont_AddItem* and *Cont_RemoveItem* can be used with contextual menus to add, remove or change menu items just as this has been shown for main menus.

Just like main menu items the items of custom context menus can have keyboard shortcuts.

Custom contextual menus can be set for all layouts of a file or for certain hot zones such as fields. In addition, all default FileMaker contextual menus can be completely eliminated from a solution.

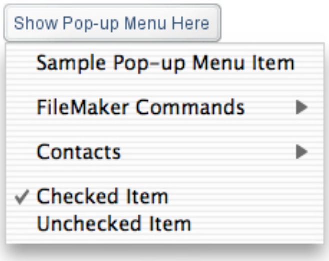
CHAPTER 4
Pop-up Menus

MenuControl enables you to show pop-up menus on the fly from a FileMaker script. Pop-up menus will be shown at specific locations immediately after the plug-in function *Cont_BuildPopupMenu* is called.

Just as main menus and contextual menus, pop-up menus are based on a menu code. Thus, a pop-up menu can contain default FileMaker menu items and custom menu items or a mix of both.

Unlike FileMaker's default pop-up menus and pop-up lists which are based on Value Lists, MenuControl pop-up menus created can trigger scripts. According to what the user chooses different actions can be taken.

The following example is shown in the *Quick Start* file under the *Pop-up Menu* tab.



Custom pop-up menu created with MenuControl

The script *Show Pop-up Menu* demonstrates how to show a pop-up menu using MenuControl.

The following fields are used:

1 **Result**: Defined as global text field.
This field will contain a result code returned by the plug-in indicating if the function succeeded or an error occurred.

2 **Popup Menu Code**: Defined as global text field.

The menu code created by the MenuComposer that contains all details about the pop-up menu such as:

- Menu items
- Cascading sub menus of the pop-up menu
- Assigned scripts to menu items
- Assigned status of the menu items (enabled/disabled)

In the script the following plug-in function is called within a *Set Field* script step to show the pop-up menu immediately.

Function Call	Explanation
`Cont_BuildPopupMenu (`	Plug-in function to install a context menu
`Popup Menu Code;`	Name of the field containing the menu code followed my a parameter separator
`"368";`	Location of the pop-up menu in the layout from top (measured in pixels) followed by a parameter separator
`"141"`	Location of the pop-up menu in the layout from left measured in pixels)
`)`	End of plug-in function call

In addition to the field which contains the menu code of the pop-up menu to be shown two optional numbers are passed to the plug-in. They indicate the location (X and Y) of the point in the layout where the pop-up menu will be shown. The numbers for a location can be retrieved using the *Object Size* legend dialog window. It can be made visible in the Layout mode under the *View* menu. Click on the unit to change to pixels (px).

An alternate way to determine the position for the pop-up menu is the FileMaker *FieldBounds* function. Refer to the chapter → **External Functions Reference** (p.54) for more details about this function.

The pop-up menu will be shown at the current mouse cursor position if the X and Y coordinates are not set.

Note: FileMaker scripts that are triggered by a pop-up menu item will not execute until the script which showed the pop-up menu has ended. Thus, it is recommended to keep scripts short that *show* pop-up menus. Ideally, they only contain one command that shows the pop-up menu.

CHAPTER 5

Alternative Menus

MenuControl enables FileMaker developers to set customized main and context menus for their database solutions. However, the concept of custom main and context menus that was discussed in this manual so far refers to menus that are specifically designed for a certain database solution.

There might also be situations when the access to FileMaker functions the end-user has should be restricted in general for all files that may ever be opened with a certain FileMaker installation. For example, consider the FileMaker copy installed on a PC of a data entry clerk. Here, it might make sense to remove the *New Database* command from the FileMaker menu so that no new database can be created by the clerk – no matter which solution is in use.

MenuControl lets you change the default FileMaker main and context menu. The feature to achieve this is called *alternative menus* because such menus are only shown in Browse, Find or Preview mode whenever no other custom MenuControl menu has been specified for the current file. They replace the default FileMaker menus, which would be shown normally if no custom MenuControl menu has been installed.

A typical use would be to remove all commands from the FileMaker main menu except for the *File* > *Open* item and a *Help* menu so that users can do nothing with FileMaker than opening a database solution which then sets custom menus that provide commands for the specific solution.

You can install one alternative main and context menu to be shown in all FileMaker user modes (Browse, Find and Preview). Alternatively, you can also install a separate main and context menu for every mode. In any case alternative main and context menus should only contain FileMaker menu items and no custom script items because an alternative main menu could be shown for any FileMaker file or even if no file is open in FileMaker.

Alternative Main Menus

The following MenuControl function is used to set alternative main menus:

```
Cont_BuildAltMenu (
Menu Code;
Menu Name;
Persistency;
Mode        )
```

The *Persistency* and the *Mode* parameters are optional. The persistency parameter specifies whether the alternative menu, which has been set, should be saved for future FileMaker sessions. The values can be “Session” or “Save”. If the parameter is not specified the plug-in will not save the alternative main menu for future sessions. Use the *Mode* parameter to install an alternative main menu for a specific FileMaker mode. The values can be “Browse”, “Find” or “Preview”. Thus, there can be up to three alternative main menus installed at the same time. By default an alternative menu is installed for all three FileMaker user modes. In Layout mode, the plug-in will always show the default FileMaker main menu.

Alternative Context Menus

In addition to alternative main menus, alternative context menus can be installed using MenuControl. However, since an alternative context menu does not know in which kind of files it might be used (it is installed for all files which do not set a custom MenuControl context menu) specifications such as target layouts or hot zones can not be made for alternative context menus.

```
Cont_BuildAltContextMenu (
Menu Code;
Menu Name;
Persistency;
Mode        )
```

Just as alternative main menus, alternative context menus can specify their persistency. In addition, alternative context menus can be installed for all FileMaker user modes (Browse, Find and Preview). Thus, there can be up to three alternative context menus installed at the same time.

Review the chapter → **External Functions Reference** (p.54) for further details about the plug-in functions *Cont_BuilAltMenu* and *Cont_BuildAltContextMenu*.

Restoring Alternative Menus

Just as custom menus alternative menus can be uninstalled. Therefore, two special functions are used, *Cont_RestoreAltMenu* and *Cont_RestoreAltContextMenu*.

Note that the name of the context menu which has been set when the menu was installed must be passed to the plug-in for security purposes when an alternative menu is removed. Besides the *Menu Name* parameter the *Restore* functions for alternative menus have a *Persistency* parameter.

More details on this functionality can be found in the chapter → **External Functions Reference** (p.54).

Alternative main and context menus can replace the default FileMaker main and context menu for *all* FileMaker files on a certain computer.

CHAPTER 6

Priority Order of Menus

As shown in previous chapters MenuControl 4 enables you to set custom main menus and contextual menus for FileMaker files.

All main and contextual menus which have been installed using either *Cont_BuildMenu* or *Cont_BuildContextMenu* or any of the functions that install alternative menus are managed by MenuControl automatically. Normally, a start script is used to *install* several main and context menus for different files and layouts. *Several* menus can be *installed* at the same time but there can only be *one* menu that is *active* at a time.

This chapter explains the conditions according to which MenuControl decides which main or context menu is shown. You can control the priority of a menu using the parameters of the plug-in functions that install the menu. The following sections explain the impact that parameters have on the priority of custom main and context menus.

Main menus priority order

When installing a main menu using *Cont_BuildMenu* this main menu is always attached to a specific FileMaker file. This can be the current file or another one but it is always a single specific file. In addition, the main menu can be attached to a specific layout and/or FileMaker user mode (Browse, Find or Preview).

If more than one main menu has been installed for the current file using a start script, MenuControl determines which menu is shown in a certain situation based on the following priority order.

Priority	Main Menu Type	Assigned to Layout	Assigned to Mode
1	Custom main menu	✓	✓
2	Custom main menu	✓	
3	Custom main menu		✓
4	Custom main menu		
5	Alternative main menu		✓
6	Alternative main menu		
7	Default FileMaker main menu		

Example:
The main menu *Menu1* is attached to the layout *Customers* (main menu priority order 2). At the same time the main menu *Menu2* is attached to the same layout but it is also attached to the *Find* mode (main menu priority order 1). This means that whenever the layout *Customers* is shown in Find mode *Menu2* will be active because it has the highest priority. In Browse and Preview mode, *Menu1* will be shown.

If two (or more) main menus of the same priority order have been set and both would actually be active in the same situation MenuControl shows the menu that has been installed last.

MenuControl will always show the default FileMaker main menu in Layout mode. In Layout mode all custom main and context menus will be replaced by the default FileMaker menus for the Layout mode.

Context menus priority order

In addition to the layout and mode settings available for the installation of a custom *main* menu, a custom *context* menu can be assigned to a target zone (hot zone) in a layout.

As soon as a custom context menu has been installed for the current file, all default FileMaker context menus in Browse, Find and Preview mode are removed for that file.

If more than one context menu has been installed for the current file, MenuControl determines which menu is shown based on the following priority order.

Priority	Context Menu Type	Assigned to Layout	Assigned to Mode	Assigned to Hot Zone
1	Custom context menu	✓	✓	✓
2	Custom context menu	✓		✓
3	Custom context menu	✓	✓	
4	Custom context menu	✓		
5	Custom context menu		✓	
6	Custom context menu			
7	No context menu is shown if all context menus have been removed for the current file: *Cont_RemoveFMContextMenus*			
8	Alternative context menu		✓	
9	Alternative context menu			
10	Default FileMaker context menu			

Example:
The custom context menu *Context1* is assigned to the file *Contacts* (context menu priority order 6). In addition, the context menu *Context2* is assigned to the layout *Details* of the *Contacts* file (context menu priority order 4). Moreover, *Context3* is assigned to a hot zone in the same layout *Details* (context menu priority order 2).

When the user right-clicks (Windows) or control-clicks (Mac) in any layout of the file *Contacts* which is not the *Details* layout, *Context1* will be shown. *Context2* is shown when the user right-clicks or control-clicks anywhere in layout *Details* but not in the hot zone defined for *Context3*. The menu *Context3* is shown when the user right-clicks or control-clicks in the hot zone in layout *Details.*

If two (or more) context menus of the same priority order have been set and both would actually be active in the same situation MenuControl shows the menu that has been installed last.

MenuControl will always show the default FileMaker context menu in Layout mode.

> When installing several custom main or contextual menus the *Priority Order of Menus* defines which of the menus will be active in a certain situation.

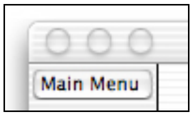
There is no priority order for pop-up menus. Such menus are removed as soon as they are closed so there is no need for MenuControl to manage pop-up menus.

CHAPTER 7
Additional Security Features

In addition to custom menus, MenuControl has additional features that can help to increase the security of your FileMaker solution or runtime application.

Disable window control buttons

This function restricts access to the window control buttons of a FileMaker file. Under Windows the Close, Maximize/Restore and Minimize buttons are removed. Under Mac OS X the Close, Maximize/Restore and Minimize buttons are disabled. Under Mac Classic the Close button is removed.



Disabled window control buttons under Mac OS X

The plug-in function *Cont_SecureWindow* is used to achieve this.
More details can be found in the chapter → **External Functions Reference** (p.54).

Remove toolbars

The FileMaker toolbars can be selectively disabled. You can disable the *Standard* and *Text Formatting* toolbars or both. This feature is not available under Windows and Mac Classic for all supported versions of FileMaker. Under Mac OS X this feature is supported for FileMaker 7 and higher only since earlier versions of FileMaker for Mac OS X do not support toolbars.

To hide and show toolbars the plug-in function *Cont_ControlToolbars* is called.
More details can be found in the chapter → **External Functions Reference** (p.54).

Make ThinClient

MenuControl provides a feature that lets you remove all menus items completely.
Only the application window frame under Windows remains. Under Mac, the *Help* menu will only be shown together with the application menu (Mac OS X).

The plug-in function *Cont_MakeThinClient* is called to achieve this.

This security feature can be reset without restarting FileMaker. If the solution file is closed, FileMaker is reset to normal mode without triggering functions in a close script. However, you can also use the *Cont_RestoreMenu* plug-in function to reset the menu.

More details can be found in the chapter → **External Functions Reference** (p.54).

Note: Additional security features provided by MenuControl let you disable the window control buttons of a file, control the FileMaker toolbars, and eliminate all nonessential menus.

CHAPTER 8
Important Developer Guidelines

It is highly recommend considering the points mentioned in this chapter when creating a FileMaker solution or runtime application with MenuControl, as they will increase the security of your FileMaker Pro solution or runtime significantly.

End-user access

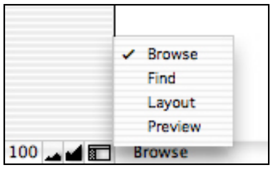
You as a FileMaker solution developer should create at least one layout containing all global fields that hold the menu codes of your MenuControl menus. There should be one global text field for each menu.

End-users should be restricted from accessing the layout mode using FileMaker *Access Privileges* (FileMaker 6/5) or *Accounts & Privileges* (FileMaker 7 and higher).

If you have only one user or the same group of users (all of them have the same Access Privileges) you might not want your users to enter a password when you solution starts. In order to avoid that you can define the end-user password as the default password of every file of your solution.

Further details on how to restrict user access can be found in the FileMaker Help and/or the FileMaker Manual.

When logged in as developer or administrator you will have full access to the file. This means you can always switch to layout mode using the mode pop-up menu located at the left bottom of a FileMaker window – this works no matter which custom menu is set.



Modes pop-up menu in FileMaker

When you switch to layout mode FileMaker will display the default FileMaker menus for this mode no matter which custom menus have been installed for other modes. This includes both types, main and contextual menus.

If you would like to restore FileMaker menus in user modes (Browse, Find and Preview) for solution debugging purposes, it is suggested to create a global menu restore script using the plug-in functions *Cont_RestoreMenu* and *Cont_RestoreContextMenu*. Assign this script to a button in your administration layout so you always have access to it.

Securing a MenuControl solution

You cannot prevent users from removing the plug-in from the FileMaker Pro System folder. However, you can use the plug-in function *Cont_Version* to have your solutions check upon startup if the plug-in is installed (via start-script).

If this function returns an empty result (because it cannot find the plug-in) the scripting in your solutions should be such that it simply does not start and tells the user why it does so (error dialog). That means the MenuControl plug-in *must* be installed for your solution to open and function. This was implemented in the *Quick Start* file as part of the start script.

The following is an example of such a start script.

```
If [Cont_Version = ""]
      Go to Layout ["Startup Error Message"]
Else
Go to ["Solution Welcome Layout"]
```

The *Startup Error Message* should have a close button only.

Disable smart quotes

It is necessary to disable the *Smart Quotes* option in the FileMaker document options when working with MenuControl.

FileMaker 6/5:
Preferences > Document > uncheck *Use smart quotes*

FileMaker 7 and higher:
File Options > Text > uncheck *Use Smart quotes*

MenuControl cannot handle menu codes from text fields with *smart quotes* switched on since FileMaker will pass different character codes for quote symbols to the plug-in when this option is enabled.

FileMaker Preferences Access

It is highly recommend not giving end-users access to the FileMaker preferences by simply not including it in the menu.

If certain preference settings are needed for your solution, it is recommended setting them using an installer application that writes them directly to the Windows Registry. Creating files that update the Windows Registry automatically can also accomplish this. For Mac OS a preferences file can be provided.

Unique menu names

MenuControl enables developer to set and modify menus of files other than the current one. To avoid security issues, it is recommended to choose password-like names for your menus. Think of a prefix that you use for all of the menus of your solution. Other developers will not be able to access your menus unless they know their exact names. In order to avoid conflicts with other solutions that use MenuControl and might be open at the same time it is also recommended to include the name of your solution into the menu names.

Help Menu under Mac OS

If the default FileMaker Help menu is not used for a solution, the operating system automatically inserts a *Help* menu under Mac OS (Mac Classic and Mac OS X). It is up to the developer to fill the *Help* menu with menu item(s). If no menu item is assigned, the system will display a disabled item called *Help not available* in the Help menu. It makes sense having at least one menu item linking to the FileMaker Online Help or a custom help layout of your solution.

Application Preferences menu under Mac OS X

The Preferences menu in the application menu (normally called “FileMaker”) under Mac OS X is a system menu which can not be removed. However, MenuControl enables you to place your own menu item in this menu that triggers a preferences script of your solution. Therefore, the MenuComposer provides an option called “Mac OS X Preferences Item” for scripts that are added to the application menu. Alternatively, the following menu code disables the Prefaces menu:

```
FM_MENUBAR ()
{
    FM_APPMENU ( "MySolution" )
    {
        FM_MACPREFERENCES ( "Preferences", "", Disabled )
    }
}
```

Runtime solutions and MenuControl

In order for a FileMaker plug-in to work with runtime solutions they need to be placed in the following folder, which has to be created manually or by an installer if it is not created by the FileMaker runtime binder.

FileMaker 5/6:
Windows: *System* folder
Macintosh: *FileMaker Extensions* folder

FileMaker 7 and higher:
Windows and Macintosh: *Extensions* folder

MenuControl in a network environment

MenuControl has to be installed on every client computer where a custom menu is needed. You can set different menus for different users/clients. Please note that MenuControl does *not* support FileMaker Web sharing networks that use browsers as clients. However, MenuControl can be used for solutions that are shared over the Internet using FileMaker clients.

CHAPTER 9

Additional Developer Guidelines

This chapter provides additional helpful guidelines that you should consider when developing a FileMaker solution with custom MenuControl menus.

Menu structure basics

When creating your menus ensure an easy-to-understand menu structure that only changes when appropriate. There should not be too many sub menu levels within a main menu. Of course, this depends on the complexity of your solution.

It helps to group items within one menu level using separators. This makes it easier for end-users to grasp the overall structure of a menu much faster. It is recommended using an *Edit* menu for each layout that lets the end user modify text fields. This menu should contain at least the items *Undo*, *Cut*, *Copy*, *Paste* and *Spell-Check*.

It is advised to associate the menu commands with common keyboard shortcuts. In addition, ensure that users can switch to any main function or main files of your database system at any time.

You should provide menu items that link to a help layout and provide copyright information about your solution and about the FileMaker platform. On Windows operating systems it is suggested locating these menu items in a Help menu. On Mac platforms these items can be placed in the Apple/Application menu.

The article *Guidelines for Designing Menus* by J. R. Brown and S. Cunningham is a great source: http://web.engr.oregonstate.edu/~pancake/cs252/guide_menus.html

MenuControl in multi-file solutions

Most of the times, complex FileMaker 6/5 solutions but also FileMaker 7-11 solutions consist of more than one FileMaker file.

When designing FileMaker 7-11 solutions you should always have *one* user interface file only. This file represents all tables of the solution. Tables can be part of the interface file or they can be occurrences of tables from other files. In a start script of this file, all custom main and context menus are installed.

In this regard, FileMaker 6/5 solutions require a little bit more planning. Most FileMaker 6/5 solutions consist of more than one user interface file because on this FileMaker platform a file cannot contain or represent several tables. Thus, custom main and context menus have to be set for several files in a start-up routine of a solution. There are two different options to set menus for multi-file solutions on FileMaker 6/5.

Each file can have its own start script that sets all custom main and context menus for that single file. Alternatively, you can have one master menu installation script for all files of the solution since MenuControl lets you install custom menus for files other than the current one.

Each method has certain advantages. It depends on your solution which way fits best. If you want to debug files with custom menus independently from other related files, it might be better to have each file set its own menus in a start script.

However, when setting custom main menus for different files from different start scripts you might notice that the main menu is updated several times during the start-up procedure of the solution. This behavior is also called *menu flickering*. To avoid menu flickering, set all main menus of a solution from a single start script or use the functions *Cont_FreezeMenu* and *Cont_RefreshMenu*. For more details about these functions review the chapter → **External Functions Reference** (p.54).

Saving a changed menu code

MenuControl 4 provides functions that let you change the items of existing menus. You can add, remove or modify specific menu items without updating the whole menu.

In your solution you might want to provide options for end-users that require dynamic menus. For instance, you might add features like *Recent Contacts* to a menu.

When a menu has been changed during a session based on user interaction you can save the menu code of that menu when the solution is closed. Next time the user starts the solution the menu will look exactly like they did when he/she used it the time before.

To retrieve the current menu code of a menu the plug-in function *Cont_GetMenu* code is triggered in a close script of the file. The menu code should be saved to a text field which is used by the start script of the file to install the menu next time the solution is opened.

For multi-user solutions it might be necessary to manage menu codes of customized user menus in a separate database table.

Menu code debugging

The MenuComposer application can check the syntax of the menu code for you. The MenuControl plug-in also checks the syntax of a menu code before setting a menu. If a menu code contains errors you are told which line you have to check by the result that is returned by the plug-in.

However, before setting a menu it is not checked if the script names that are used for custom menu commands are actually available in the target file. So if a custom menu item is triggered and MenuControl cannot find the associated script in the current file a plug-in error message is shown.

In order to avoid this error message the name of a custom error script can be passed to the plug-in using the function *Cont_SetErrScript*. If a custom error script has been set for the current file the plug-in will trigger this script in case that a script of a menu item is not found. The plug-in error message will only be shown if the custom error script cannot be found, either.

This functionality can be used to create custom debug messages such as “A script could not be found. Contact your solution developer”, messages in a language other than English or other routines.

Using Auto Update with FileMaker Server

When using MenuControl in network solutions you can update the plug-ins on all workstations using the Auto Update plug-in and FileMaker Server. In order to compare the local and the remote versions of the plug-in you will need to convert the result returned by the plug-in function *Cont_Version* as follows:

FileMaker 6/5:

```
TextToNum(External("Cont_Version";"Version"))
```

FileMaker 7 and higher:

```
GetAsNumber(Cont_Version ("Version"))
```

This function returns an integer that can be used for comparisons. Please review the documentation of the Auto Update plug-in which is part of the FileMaker Server manual for further details.

Legal FileMaker Acknowledgements

When creating a FileMaker Pro menu with MenuControl you must include a menu item that shows the complete Copyright Information of FileMaker as displayed in the default FileMaker dialog *About FileMaker*.

CHAPTER 10
Using MenuControl Functions

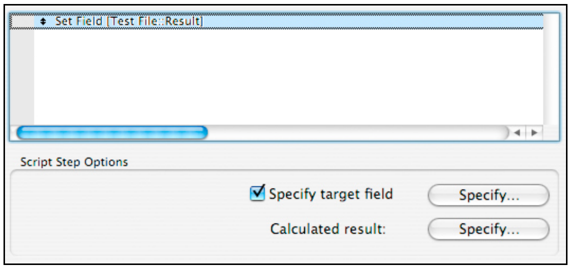
This chapter gives you a general introduction on how to use MenuControl plug-in functions with FileMaker. It explains where you find the plug-in functions in FileMaker and how to trigger them. The next chapter provides in-depth information about all MenuControl functions and their parameters.

Accessing external functions

Since MenuControl is a FileMaker plug-in, so-called *external functions* are used to trigger the plug-in from a FileMaker database solution. They are called external functions because these functions are provided by a plug-in and thus, they are not part of the actual FileMaker application. In order to use the external functions provided by a plug-in it must always be installed and enabled in the FileMaker application preferences.

To access external functions provided by installed plug-ins the FileMaker calculation editor is used. FileMaker shows this dialog whenever a calculation has to be defined (i.e. for calculated fields, validation calculations etc). In most cases you will trigger MenuControl functions from a script. The easiest way to perform an external function call from a script is to add the script step *Set Field* to your script.

Next, you specify the target field which will contain the result of the operation. The result of MenuControl operations is a simple result code which tells you if a function could be executed successfully or if errors occurred. Therefore, you should setup a global text field in your solution and call it *Result*. Specify the field *Result* as target field of the calculation.



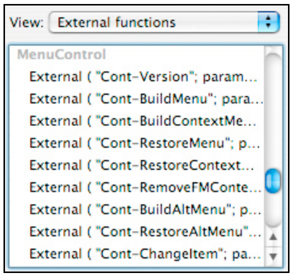
```
Set Field [Test File::Result]
```

Script Step Options

☑ Specify target field

Set Field command in ScriptMaker

Click the *Specify* button (the second *Specify* button in FileMaker 7 and higher) to open the calculation editor. In the top right corner of the dialog you find a drop-down menu called *View*. Change to the section *External functions*. All MenuControl functions are listed in the section *MenuControl*. If you have additional plug-ins installed you might have to scroll down to the *MenuControl* section of the list. If the MenuControl functions do not appear in the list, the plug-in is not installed correctly.



View: External functions

MenuControl
External ("Cont-Version"; param…
External ("Cont-BuildMenu"; para…
External ("Cont-BuildContextMe…
External ("Cont-RestoreMenu"; p…
External ("Cont-RestoreContext…
External ("Cont-RemoveFMConte…
External ("Cont-BuildAltMenu"; p…
External ("Cont-RestoreAltMenu"…
External ("Cont-ChangeItem"; pa…

MenuControl functions in the Calculation Editor

Double-click on the MenuControl function in the list which you would like to use for this script step operation. For an example click on *Cont_BuildMenu (Menu Code ; Menu Name {; Layout Name ; Mode ; File Name })* if you want to set a main menu. As a result the function is copied to the large text box of the calculation editor.

Function parameters

In order to tell the MenuControl plug-in exactly what it should do when the function is triggered, its *parameters* need to be specified, separated by semicolon character.

The function *Cont_BuildMenu* has six parameters. In this chapter they are listed as follows:

```
Cont_BuildMenu ( Menu Code | Menu Name | Layout Name | Mode | File
Name )
```

In the calculation editor it looks like this:

```
Cont_BuildMenu (
Menu Code;
Menu Name;
Layout Name;
Mode;
File Name   )
```

Note that each parameter like *Menu Code* and *Menu Name* represents a text value. Thus, a parameter value has to be the name of the text field or a text value itself. If it is a text value, the parameter must be enclosed in quotation marks (*"Value")*. The example above assumes that the parameters *Menu Code*, *Menu Name* etc. have been defined as text fields or global text fields in the file.

It is also recommended that you format the function call as shown above. One line is used for each parameter. This gives a much better overview than a single line of text. However, for the plug-in it does not make any difference how you format the function call.

Mandatory and optional parameters

Most functions have some parameters that are mandatory and some that are optional. Mandatory parameters are listed in the beginning and optional parameters are listed at the end of a function call. This allows you to omit optional parameters which you would not like to use in your function call.

The function *Cont_BuildMenu* has only two mandatory parameters: *Menu Code* and *Menu Name*. All other parameters (*Layout Name*, Mode and *File Name*) are optional. If you do not want to specify any of the optional parameters the function call could simply look like this:

```
Cont_BuildMenu(
Menu Code;
Menu Name   )
```

In other cases you my want to skip one optional parameter but use another optional parameter which is listed after the first one. To achieve this, an empty text value has to be passed to the plug-in for the parameter that is skipped. In the following example the parameter *Layout Name* is skipped. The parameter *File Name* is not used either but it does not have to be skipped since it is the last one in the parameter list and can simply be omitted.

```
Cont_BuildMenu (
Menu Code;
Menu Name;
"";
Mode         )
```

Note that you do not need to ad a semicolon character (";") after the last parameter (in this case *Mode*) even if the last optional one (*File Name)* has been omitted.

Result codes

Most plug-in functions that are called return a result code and a short description. The result code indicates if the plug-in function succeeded or failed. If it failed the code also provides details about the reason. You can evaluate result codes with a FileMaker script and perform certain activities (i.e. show an error message to the user).

The following table lists all result codes returned by the plug-in.

Result Code	Explanation
000	“OK” (plug-in function succeeded)
-001	"Invalid menu item" (returned Cont_ChangeItem, Cont_AddItem and Cont_RemoveItem - menu item location is incorrect)
-002	"Incorrect parameter" (one of the parameters is specified incorrectly)
-003	"Bad menu name" (returned by menu building functions – unique and valid menu name is not specified)
-004	"Menu is not installed"
-005	"Generic error" (internal plug-in error – please contact the Dacons Support is you should ever get this error)
-006	"Invalid registration data"
-007	"Bad mode" (mode name is specified incorrectly)
-008	"Hot zone is specified, but layout name is empty" (returned by Cont_BuildContextMenu only)
-009	"Invalid secure string" (returned by Cont_SecureWindow only - secure string is incorrect)

Some MenuControl functions such as *Cont_GetMenuCode* return text content (in this case the menu code of a specified menu) as result instead of “000 (OK)” if they succeed.

Please find a list of all MenuControl functions in the next chapter → **External Functions Reference** (p.54).

CHAPTER 11
External Functions Reference

In this chapter you find detailed specifications about all MenuControl 4 functions and their parameters. A general introduction on how to use these functions is provided in the chapter → **Using MenuControl Functions** (p.50).

Cont_Version

Description:	This function returns the version of the MenuControl plug-in. Use this function to check if the plug-in is installed when you database solution starts (start script). The version information can also be used for the AutoUpdate plug-in which pushes an updated version of a plug-in to all FileMaker network clients automatically.
Syntax:	Cont_Version ({parameter})
Parameters:	If no parameter is used (empty parameter “”) all items of information mentioned below are returned in the following order: Version, Product Platform and Copyright, separated by space character. To retrieve only a specific item from the list below use the appropriate value for the *Parameter*. In addition to the empty value (“”) it can have the following values: Version – Returns the exact version of the plug-in installed. In most cases you will pass this value to the plug-in to retrieve the version number. Product – Returns the name of the product which is “MenuControl”. Platform – Return the platform the plug-in is running on. This can be *Windows*, *Mac OS X* or *Mac Classic*. Copyright – Returns copyright information of the plug-in.
Result:	The plug-in returns the result in text format. Depending on the parameter the plug-in version or other pieces of information are returned. If no MenuControl plug-in is installed, an empty value is returned as result.

Cont_BuildMenu

Description:	This function creates a main menu based on the menu code which is passed to the plug-in. A custom main menu can contain script items and default FileMaker menu items. Main menus are automatically attached to a certain FileMaker file. The plug-in switches main menus automatically when another file comes to front. A main menu can also be attached to a certain layout of a file or a FileMaker mode.	
Syntax:	Cont_BuildMenu (Menu Code ; Menu Name {; Layout Name ; Mode ; File Name })	
Parameters:	Menu Code	This parameter is **mandatory**. Use this parameter to pass menu code information to the plug-in that describes the structure and items of the menu to be set. The menu code can be generated using the MenuComposer, a simple point-and-click tool that comes with MenuControl. Use the *Menu Code* parameter to pass the name of a field to the plug-in which contains the menu code (recommended) or paste the menu code directly into the FileMaker Calculation Editor. Therefore, the function “Copy Code String” has to be used in the MenuComposer. Menu codes for main menus start with the tag *FM_MENUBAR*.
	Menu Name	This parameter is **mandatory**. Every main or contextual menu which is set using MenuControl is set with a unique name. This name can be used later to update or reset a specific menu. Please note that the menu name must be unique. It is recommended to use your solution name as part of the menu name. Example: “MySolutionMainMenu1”
	Layout Name	This parameter is **optional**. If you do not specify a value for this parameter the main menu will be installed for all layouts of the file. However, if you do specify the name of a layout using this parameter, the menu will only be shown if this layout is visible. If the user switches to other layouts, other custom main menus will be shown or the default FileMaker menu will be shown if not other menus have been installed for the file.
	Mode	This parameter is **optional**. MenuControl menus support the FileMaker modes *Browse*, *Find* and *Preview*. By default a menu is installed for all three modes if no value is specified for this parameter. If you want to install a menu for a certain mode (i.e. Find mode with certain commands that should only be available in this mode) specify the appropriate mode using this parameter. Values can be “Browse”, “Find” or “Preview”.
	File Name	This parameter is **optional**.

	A main menu is always installed for a certain file. MenuControl restores the appropriate menu automatically when another file comes to front. When the value of this parameter is empty the plug-in will install the main menu for the current file. If you would like to install a main menu for another file of your solution you can use this parameter to specify the file name. Note: This parameter only supports file names (including extensions) but not file paths.
Result:	The plug-in returns “OK (000)” as result when the main menu could be installed. Otherwise an error code is returned. If the menu code contains syntax errors MenuControl returns the line of the menu code to be checked.

Cont_BuildContextMenu

Description:	This function creates a context menu. Context menus are shown when the user right-clicks (Windows) or control-clicks (Mac) a certain area of a layout. MenuControl can create context menus that contain script items or default FileMaker menu items. You can create context menus for a whole file, a certain layout, a certain field in a layout or other hot zones. MenuControl switches context menus automatically when the user goes to another layout or file.	
Syntax:	Cont_BuildContextMenu (Menu Code ; Menu Name {; Layout Name ; Hot Zone ; Mode ; File Name })	
Parameters:	Menu Code	This parameter is **mandatory**. Use this parameter to pass menu code information to the plug-in that describes the structure and items of the context menu to be set. The menu code can be generated using the MenuComposer, a simple point-and-click tool that comes with MenuControl. Use the *Menu Code* parameter to pass the name of a field to the plug-in which contains the menu code (recommended) or paste the menu code directly into the FileMaker Calculation Editor. Therefore, the function “Copy Code String” has to be used in the MenuComposer. Menu codes for context menus start with the tag *FM_CONTEXTMENU*.
	Menu Name	This parameter is **mandatory**. Every main or contextual menu which is set using MenuControl is set with a unique name. This name can be used later to update or reset a specific menu. Please note that the menu name must be unique. It is recommended to use your solution name as part of the menu name. Example: “MySolutionContextMenu1”
	Layout Name	This parameter is **optional**. If you do not specify a value for this parameter the context menu will be installed for all layouts of the file. However, if you do specify the name of a layout using this parameter, the context menu will only be shown if this layout is visible and the user right-clicks (Windows) or control-clicks (Mac). If the user switches to other layouts, other custom context menus will be shown or the default FileMaker context menu will be shown if not other context menus have been installed for the file.
	Hot Zone	This parameter is **optional**. In many cases context menus are installed for certain parts of a layout only such as a certain field. When the user right-clicks (Windows) or control-clicks (Mac) in this area a special context menu will be shown. To use this parameter the *Layout Name* parameter has to be specified first since a Hot Zone can only be installed for a certain layout. This parameter accepts the result returned by the FileMaker function *FieldBounds*. Use the result of this function to pass the boundaries of a certain field

	in a layout to MenuControl. Alternatively, you can format this parameter manually to create a Hot Zone as follows: “X1 Y1 X2 Y2” X1: Represents the offset of the Hot Zone from the left layout border. Y1: Represents the offset of the Hot Zone from the top layout border. X2: The offset of the Hot Zone from the left layout border plus the width of the Hot Zone. Y2: The offset of the Hot Zone from the top layout border plus the height of the Hot Zone. All values are indicated in pixels. Due to a FileMaker limitation context menus that are set for a Hot Zone can only be shown in the first record line in List View layouts.
Mode	This parameter is **optional**. MenuControl menus support the FileMaker modes *Browse*, *Find* and *Preview*. By default a context menu is installed for all three modes if no value is specified for this parameter. If you want to install a menu for a certain mode (i.e. Find mode with certain commands that should only be available in this mode) specify the appropriate mode using this parameter. Values can be “Browse”, “Find” or “Preview”.
File Name	This parameter is **optional**. A context menu is always installed for a certain file. MenuControl restores the appropriate menu automatically when another file comes to front. When the value of this parameter is empty the plug-in will install the context menu for the current file. If you would like to install a context menu for another file of your solution you can use this parameter to specify the file name. Note: This parameter only supports file names (including extensions) but not file paths.

Result: The plug-in returns “OK (000)” as result when the context menu could be installed. Otherwise an error code is returned. If the menu code contains syntax errors MenuControl returns the line of the menu code to be checked.

Cont_RestoreMenu

Description:	Use this function to remove a custom main menu which has been installed using MenuControl. This function can also remove all custom main menus from the current file. If a specific menu has been restored, another menu is shown according to the *Priority Order of Menus*. If no other custom main menu is installed the default FileMaker main menu will be used.	
Syntax:	Cont_RestoreMenu ({ Menu Name })	
Parameters:	Menu Name	This parameter is **optional**. Pass the name of a specific main menu which has been assigned to it when it was installed using the function Cont_BuildMenu to reset this specific menu. If an empty parameter is passed to the plug-in all custom main menus for the current file (for all layouts and modes) are restored and the default FileMaker menu will be shown if no alternative main menu has been installed. If a menu for a different file than the current one should be restored, the Menu Name parameter is mandatory.
Result:	The plug-in returns “OK (000)” as result when the specified menus could be restored. Otherwise an error code is returned.	

Cont_RestoreContextMenu

Description:	Use this function to remove a custom context menu which has been installed using MenuControl. This function can also be used to restore all custom context menus of the current file. If a specific context menu has been restored, another context menu is shown according to the *Priority Order of Menus*. If no other custom context menu is installed the default FileMaker context menu will be used. This function also restores all default FileMaker context menus for the current file in case they have been removed using Cont_RemoveFMContextMenus.	
Syntax:	Cont_RestoreContextMenu ({ Menu Name })	
Parameters:	Menu Name	This parameter is **optional**. Pass the name of a specific context menu which has been assigned to it when it was installed using the function Cont_BuildContextMenu to remove this specific menu. If an empty parameter is passed to this function all custom context menus are removed from the current file. If a menu for a different file than the current one should be restored, the Menu Name parameter is mandatory.
Result:	The plug-in returns “OK (000)” as result when the specified menus could be restored. Otherwise an error code is returned.	

Cont_RemoveFMContextMenus

Description:	This function removes all default FileMaker context menus from a file. Use this function if you would not like to show any default FileMaker context menus.	
Syntax:	Cont_RemoveFMContextMenus ({ File Name })	
Parameters:	File Name	This parameter is **optional**. Use this parameter to specify the name of a FileMaker file from which you would like to remove all default context menus. Leave this parameter empty to refer to the current file. This parameter supports only the name of a file (including its extension) but not the path.
Note:	To restore default FileMaker context menus for the current file the function Cont_RestoreContextMenus has to be called with empty parameter.	
Result:	The plug-in returns “OK (000)” as result when all default FileMaker context menus have been restored from the specified file. Otherwise an error code is returned.	

Cont_BuildAltMenu

Description:	This function lets you set an alternative main menu which is shown for all files and layout for which no custom MenuControl menu has been set. An alternative menu should be installed to restrict access to FileMaker functions no matter which file the user is currently working with. Alternative menus should contain FileMaker menu items only and no custom script items.	
Syntax:	Cont_BuildAltMenu (Menu Code ; Menu Name {; Persistency ; Mode })	
Parameters:	Menu Code	This parameter is **mandatory**. Use this parameter to pass menu code information to the plug-in that describes the structure and items of the alternative main menu to be set. The menu code can be generated using the MenuComposer, a simple point-and-click tool that comes with MenuControl. Use the *Menu Code* parameter to pass the name of a field to the plug-in which contains the menu code (recommended) or paste the menu code directly into the FileMaker Calculation Editor. Therefore, the function “Copy Code String” has to be used in the MenuComposer. Menu codes for main menus start with the tag *FM_MENUBAR*.
	Menu Name	This parameter is **mandatory**. Every main or contextual menu which is set using MenuControl is set with a unique name. This is also the case for alternative main menus. The menu name can be used later to update or reset a specific menu. Please note that the menu name must be unique. It is recommended to use your solution name as part of the menu name. Example: “MySolutionAltMainMenu1”
	Persistency	This parameter is **optional**. Alternative menus can be installed and saved so that the same menu is used in later FileMaker sessions. By default alternative menus are not stored. Leave this parameter empty or use the value “Session” if the alternative menu should not be stored. This will have the effect that in future FileMaker sessions the default FileMaker main menu will be shown for files that have no other custom MenuControl menu installed. Set this parameter to “Save” in order to keep the alternative menu. MenuControl will install it every time FileMaker starts (the MenuControl plug-in has to be installed).
	Mode	This parameter is **optional**. MenuControl menus support the FileMaker modes *Browse*, *Find* and *Preview*. By default an alternative menu is installed for all three modes if no value is specified for this parameter. If you want to install an alternative menu for a certain mode specify the appropriate mode using this parameter. Values can be “Browse”, “Find” or “Preview”. Note that if you install an alternative menu only for a specific mode the default FileMaker menu will be shown in the other modes if no additional alternative menus have

been installed. You can install several alternative main menus for different modes (up to three). They all have to have different Menu Names.

Result:	The plug-in returns “OK (000)” as result when the alternative main menu could be installed. Otherwise an error code is returned. If the menu code contains syntax errors MenuControl returns the line of the menu code to be checked.

Cont_BuildAltContextMenu

Description: This function lets you set an alternative context menu which is shown for all files and layout for which no custom MenuControl menu has been set. An alternative context menu should be installed to restrict access to FileMaker functions no matter which file the user is currently working with. Alternative menus should contain FileMaker menu items only and no custom script items.

Syntax: Cont_BuildAltContextMenu (Menu Code ; Menu Name {; Persistency ; Mode })

Parameters:

Menu Code

This parameter is **mandatory**.
Use this parameter to pass menu code information to the plug-in that describes the structure and items of the alternative context menu to be set. The menu code can be generated using the MenuComposer, a simple point-and-click tool that comes with MenuControl. Use the *Menu Code* parameter to pass the name of a field to the plug-in which contains the menu code (recommended) or paste the menu code directly into the FileMaker Calculation Editor. Therefore, the function “Copy Code String” has to be used in the MenuComposer.
Menu codes for context menus start with the tag *FM_CONTEXTMENU*.

Menu Name

This parameter is **mandatory**.
Every main or contextual menu which is set using MenuControl is set with a unique name. This is also the case for alternative context menus. The menu name can be used later to update or reset a specific menu. Please note that the menu name must be unique. It is recommended to use your solution name as part of the menu name. Example: “MySolutionAltContextMenu1”

Persistency

This parameter is **optional**.
Alternative menus can be installed and saved so that the same menu is used in later FileMaker sessions. By default alternative menus are not stored. Leave this parameter empty or use the value “Session” if the alternative menu should not be stored. This will have the effect that in future FileMaker sessions the default FileMaker menu will be shown for files that have no other custom MenuControl context menu installed. Set this parameter to “Save” in order to keep the alternative menu. MenuControl will install it every time FileMaker starts (the MenuControl plug-in has to be installed).

Mode

This parameter is **optional**.
MenuControl menus support the FileMaker modes *Browse*, *Find* and *Preview*. By default an alternative menu is installed for all three modes if no value is specified for this parameter. If you want to install an alternative menu for a certain mode specify the appropriate mode using this parameter. Values can be “Browse”, “Find” or “Preview”. Note that if you install an alternative menu only for a specific mode the default FileMaker context

menu will be shown in the other modes if no additional alternative menus have been installed. You can install several alternative context menus for different modes (up to three). They all have to have different Menu Names.

Cont_RestoreAltMenu

Description:	This function removes an alternative FileMaker main menu which has been set before using the function Cont_BuildAltMenu.	
Syntax:	Cont_RestoreAltMenu (Menu Name {; Persistency })	
Parameters:	Menu Name	This parameter is **mandatory**. Specifies the name of the alternative main menu which is to be removed. This name has been set when the alternative main menu was set using Cont_BuildAltMenu
	Persistency	This parameter is **optional**. Alternative menu settings can be changed temporarily for the current session or they can be saved for all future FileMaker session. Set this parameter to “Session” or leave it empty if you would not like to store the changed settings for future FileMaker sessions. Set this parameter to “Save” if you would like to store the changes for future sessions.
Result:	The plug-in returns “OK (000)” as result when the alternative main menu could be installed. Otherwise an error code is returned. If the menu code contains syntax errors MenuControl returns the line of the menu code to be checked.	

Result:	The plug-in returns “OK (000)” as result when the alternative context menu could be installed. Otherwise an error code is returned. If the menu code contains syntax errors MenuControl returns the line of the menu code to be checked.

Cont_RestoreAltContextMenu

Description:	This function is triggered to remove an alternative context menu which has been installed for a specific FileMaker mode or for all FileMaker modes to replace the default FileMaker context menu in those files for which no custom MenuControl context menu has been installed.	
Syntax:	Cont_RestoreAltContextMenu (Menu Name {; Persistency })	
Parameters:	Menu Name	This parameter is **mandatory**. The name of the alternative context menu which is to be removed has to be passed to this function.
	Persistency	This parameter is **optional**. It specifies whether the settings should have effect for the current FileMaker session only or if they should be saved for all future FileMaker sessions. Leave this parameter empty or set it to “Session” if you would not like to save changes for future sessions. Otherwise set the parameter to “Save”.
Result:	The plug-in returns “OK (000)” as result when the default FileMaker context menu could be restored. Otherwise an error code is returned.	

Cont_ChangeItem

Description:	This MenuControl function is used to change a menu item of an existing custom main or context menu which has been installed before.	
Syntax:	Cont_ChangeItem (Item Location ; New Item Code)	
Parameters:	Item Location	This parameter is **mandatory**. This parameter tells MenuControl where the menu item which is to be updated is located. The location of a menu item is specified as follows: *“MenuName:1>3”* This would direct MenuControl to the third item located in the first menu section of the menu called “MenuName”. Please note that separators count as menu items, too. To access the application menu under Mac OS X or the Apple Menu under Mac Classic refer to it using the number *0*: *“MenuName:0>1”* refers to the first item in the application menu of the custom menu with the name “MenuName”. For menus which use several levels of cascading sub menus the path of a menu item could be longer such as: *“MenuName:1>2>5”*
	New Item Code	This parameter is **mandatory**. This parameter passes the updated menu code for the specified menu item to the plug-in. Note that the menu code only consists of the code for one menu item such as *FM_MENUSCRIPTITEM (“Script”, Script Name)* If you embed the item code into a FileMaker function call (calculation editor) all quotation marks have to be doubled. The example above would look like this: *FM_MENUSCRIPTITEM (““Script””, Script Name)*
Result:	The plug-in returns “OK (000)” as result when the specified menu item could be updated successfully. Otherwise an error code is returned.	

Cont_AddItem

Description:	This MenuControl function is used to add a menu item to an existing custom main or context menu which has been installed before.	
Syntax:	Cont_AddItem (Item Location ; Item Code)	
Parameters:	Item Location	This parameter is **mandatory**. This parameter tells MenuControl where to insert the new menu item. The location of a menu item is specified as follows: *“MenuName:1>3”* This would direct MenuControl to the third item located in the first menu section of the menu called “MenuName”. Please note that separators count as menu items, too. To access the application menu under Mac OS X or the Apple Menu under Mac Classic refer to it using the number *0*: *“MenuName:0>1”* refers to the first item in the application menu of the custom menu with the name “MenuName”. For menus which use several levels of cascading sub menus the path of a menu item could be longer such as: *“MenuName:1>2>5”*. The new item will be inserted at the specified location in the menu. The item that is currently positioned at this location (and all items below it) will be moved down.
	Item Code	This parameter is **mandatory**. This parameter passes the menu code of the new to the plug-in. Note that the menu code only consists of the code for one menu item such as *FM_MENUSCRIPTITEM (“Script”, Script Name)* If you embed the item code into a FileMaker function call (calculation editor) all quotation marks have to be doubled. The example above would look like this: *FM_MENUSCRIPTITEM (““Script””, Script Name)*
Result:	The plug-in returns “OK (000)” as result when the specified menu item could be added to the menu successfully. Otherwise an error code is returned.	

Cont_RemoveItem

Description:	The function removes a specific menu item from a custom menu which has been set before using MenuControl. This can be a custom main or context menu.	
Syntax:	Cont_RemoveItem (Item Location)	
Parameters:	Item Location	This parameter is **mandatory**. This parameter tells MenuControl where the menu item which is to be removed is located. The location of a menu item is specified as follows: *“MenuName:1>3”* This would direct MenuControl to the third item located in the first menu section of the menu called “MenuName”. Please note that separators count as menu items, too. To access the application menu under Mac OS X or the Apple Menu under Mac Classic refer to it using the number *0*: *“MenuName:0>1”* refers to the first item in the application menu of the custom menu with the name “MenuName”. For menus which use several levels of cascading sub menus the path of a menu item could be longer such as: *“MenuName:1>2>5”*
Result:	The plug-in returns “OK (000)” as result when the specified menu item could be removed successfully. Otherwise an error code is returned.	

Cont_BuildPopupMenu

Description:	Pop-up menus are generated on the fly. They will appear in the layout in the moment when this function is called. In most cases you will show pop-up menus from scripts when the user clicks a button in a layout.	
Syntax:	Cont_BuildPopupMenu (Menu Code {; Location X ; Location Y })	
Parameters:	Menu Code	This parameter is **mandatory**. Use this parameter to pass menu code information to the plug-in that describes the structure and items of the pop-up menu which is to be shown. The menu code can be generated using the MenuComposer, a simple point-and-click tool that comes with MenuControl. Use the *Menu Code* parameter to pass the name of a field to the plug-in which contains the menu code (recommended) or paste the menu code directly into the FileMaker Calculation Editor. Therefore, the function “Copy Code String” has to be used in the MenuComposer. Menu codes for pop-up menus start with the tag *FM_POPUPMENU*.
	Location X	This parameter is **optional**. Specifies the offset from the left layout border in pixels at which the pop-up menu will be shown.
	Location Y	This parameter is **optional**. Specifies the offset from the top layout border in pixels at which the pop-up menu will be shown.
Note:	The parameters *Location X* and *Location Y* can also be separated by a space character instead of a pipe character (“|”). Thus, the result returned by the FileMaker *FieldBounds* function can be use to specify the location of a pop-up menu in a layout. Leave the parameters *Location X* and *Location Y* empty to show the pop-up menu at the current cursor position.	
Result:	The plug-in returns “OK (000)” as result when the pop-up menu could be shown. If the menu code contains syntax errors the result returned by this function contains the menu code line number to be checked.	

Cont_ControlToolsBars

Description:	This function lets you control the availability of the FileMaker Standard toolbar and the FileMaker Text toolbar.	
Syntax:	Cont_ControlToolBars (Toolbar Availability)	
Parameters:	Toolbar Availability	This parameter is **mandatory**. It can have one of the following values: *Standard, Text, On* *Standard, Text, Off* *Standard, On* *Standard, Off* *Text, On* *Text, Off*
Note:	Toolbar settings are not stored or switched automatically. When you solution hides toolbars users can get them back by switching to another file which shows the default FileMaker menu and re-enable toolbars. Use FileMaker access privileges to prevent users from using FileMaker commands that they should not use for your solution (such as deleting records).	
Result:	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.	

Cont_SecureWindow

Description:	This function restricts access to the window control buttons of a FileMaker file. Under Windows the Close, Maximize/Restore and Minimize buttons are removed. Under Mac OS X the Close, Maximize/Restore and Minimize buttons are disabled. Under Mac Classic the Close button is removed.	
Syntax:	Cont_SecureWindow (Secure Status {; File Name ; Security String })	
Parameters:	Secure Status	This parameter is **mandatory**. This parameter can have the values “ON” or “OFF” specifying if the secure window feature should be switched on or off for the specified file.
	File Name	This parameter is **optional**. This parameter specifies the name of the file (including extension) for which the secure window feature should be switched on or off. By default the function refers to the current file (if the value of this parameter is empty).
	Security String	This parameter is **optional**. To make your setting secure you can pass a string to the plug-in using this function. MenuControl will then allow changes of the secure window feature settings for the specific file only if the same Security String is again passed to the plug-in. The Security String is saved for the entire FileMaker session.
Note:	Under FileMaker 7 and higher a single file can have several windows. The secure window feature refers to all windows of a file.	
Result:	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.	

Cont_MakeThinClient

Description:	This function installs an empty main menu. All menus except for the Help menu (Mac OS X and Mac Classic) and the Application Menu (Mac OS X) are removed. In addition, the FileMaker Standard and the Text toolbars are hidden which equals the function Cont_ControlToolBars with the parameter “Standard, Text, Off”
Syntax:	Cont_MakeThinClient
Parameters:	This function does not require a parameter; just use “”. The main menu is removed for the current file.
Note:	This function does not remove context menus from the current file. To remove all default FileMaker context menus the function Cont_RemoveAllFMContextMenus has to be triggered.
Result:	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.

Cont_SetErrScript

Description:	By default MenuControl shows an error message to the user whenever a menu item should trigger a script that can not be found in the current file. This function lets you set the name of a script which is triggered instead.	
Syntax:	Cont_SetErrScript (Script Name)	
Parameters:	Script Name	This parameter is **mandatory**. It sets the name of the script which is to be triggered whenever MenuControl can not find a script.
Note:	The script which is set using this function will always be searched by the plug-in in the file which is currently active in the moment a menu script can not be found. When setting an error script this function does not check if the error script really exists in the current file. It will trigger the error script if it can be found. Otherwise the normal error message will be shown to the user.	
Result:	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.	

Cont_GetItemString

Description:	MenuControl enables you to store invisible information with menu items in their “String” tag such as: *FM_SCRIPTITEM (“Script” , Script Name, String: “My String”)* This string can contain any information. In most cases you will want to retrieve this information from MenuControl when an item has been clicked. This enables you to use the same script for several menu items and take different actions within that script depending on the menu item that was clicked by the user. You could, for instance, show the name of a file to be opened in the menu and store its path in the String tag of the menu item. The function Cont_GetItemString always returns the String value of the last menu item that was clicked.
Syntax:	Cont_GetItemString
Parameters:	This function does not require any parameter value; just use “”.
Result:	This function returns the string that has been stored in the String tag of the last menu item that was clicked. If errors occur this function returns an error code.

Cont_GetLastScript

Description:	This function returns the name of the last script that has been triggered by MenuControl.
Syntax:	Cont_GetLastScript
Parameters:	No parameter is needed for this function; just use “”.
Result:	This function returns the name of the last script which has been triggered by MenuControl. If errors occur and error code is returned instead.

Cont_GetMenuCode

Description:	This function returns the menu code of a specified menu. This can be a main or a context menu. This function is very handy if you would like to store the menu code of a menu which has been changed during the session (i.e. by using function like Cont_ChangeItem, Cont_AddItem or Cont_RemoveItem) so you can re-install it the next time your database solution is used.	
Syntax:	Cont_GetMenuCode (Menu Name)	
Parameters:	Menu Name	This parameter is **mandatory**. The name of the menu of which you would like to retrieve the menu code. This can be a main menu, a context menu or an alternative (main or context) menu.
Result:	This function returns the current menu code of the specified menu. Otherwise an error code is returned.	

Cont_FreezeMenu

Description:	Use this function to freeze the main menu bar. This can be useful during the start-up process of your solution if it consists of many different user interface files which all set a main menu. In such cases your solution should freeze the main menu bar by triggering this function then install all main menus for all files and update the menu bar as a last step by calling Cont_RefreshMenu.
Syntax:	Cont_FreezeMenu
Parameters	This function does not need a parameter; just use “”.
Result	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.

Cont_RefreshMenu

Description:	This function updates the main menu after it has been frozen by triggering Cont_FreezeMenu.
Syntax:	Cont_RefreshMenu
Parameters:	This function does not need a parameter; just use “”.
Result:	This function returns “OK (000)” as result when it could be executed successfully. Otherwise an error code is returned.

Cont_RegisterSession

Description:	To remove all trial limitation from the plug-in it has to be registered using the registration data you retrieve from Dacons after you purchased a MenuControl license. The plug-in can be registered manually using the preferences dialog (FileMaker Application Preferences > Plug-Ins > MenuControl). To avoid manual plug-in registration when you ship your database solution to a client or distribute a FileMaker Runtime application with MenuControl you can also register the plug-in from the start script of your solution by calling this function with your registration data *before any other plug-in function is triggered.* Note that registration from script will not be stored. This function has to be performed every time your solution starts if you decide to register the plug-in via external function.	
Syntax:	Cont_RegisterSession (User Name ; User Code)	
Parameters:	User Name	This parameter is **mandatory**. Your user name provided by Dacons after registration.
	User Code	Your user code provided by Dacons after registration.
Result:	This function returns “OK (000)” as result when the current session could be registered. Otherwise an error code is returned.	