

FileMaker® Pro 10 Advanced

Development Guide



© 2007-2009 FileMaker, Inc. All rights reserved.

FileMaker, Inc.
5201 Patrick Henry Drive
Santa Clara, California 95054

FileMaker, the file folder logo, Bento and the Bento logo are either trademarks or registered trademarks of FileMaker, Inc. in the U.S. and other countries. Mac and the Mac logo are the property of Apple Inc. registered in the US and other countries. All other trademarks are the property of their respective owners.

FileMaker documentation is copyrighted. You are not authorized to make additional copies or distribute this documentation without written permission from FileMaker. You may use this documentation solely with a valid licensed copy of FileMaker software.

All persons, companies, email addresses, and URLs listed in the examples are purely fictitious and any resemblance to existing persons, companies, email addresses or URLs is purely coincidental. Credits are listed in the Acknowledgements documents provided with this software. Mention of third-party products and URLs are for informational purposes only and constitutes neither an endorsement nor a recommendation. FileMaker, Inc. assumes no responsibility with regard to the performance of these products.

For more information, visit our website at www.filemaker.com.

Edition: 01

Contents

Chapter 1

Introducing FileMaker Pro Advanced

About this guide	5
Using the FileMaker Pro Advanced documentation	5
Where to find PDF documentation	6
Abiding by the license agreement for runtime solutions	6
Your responsibilities as a developer	7

Chapter 2

Creating database solutions

Using the Developer Utilities	9
About creating runtime solutions	10
Converting and upgrading previous solutions	11
Binding the solution	12
Starting runtime database solutions	13
Distributing runtime solutions	13
Organizing solution components	13
Choosing a distribution method	14
Testing before and after creating your solution	15
Distributing updates to runtime database solutions	16
Creating Kiosk solutions	17

Chapter 3

Customizing database solutions

Copying or importing field and table schemas	19
Creating custom functions	20
About custom menus	20
Custom menu example	21
Creating custom menus	22
Creating custom menu items	22
Creating custom menu sets	24
Creating custom layout themes	24
Requirements for theme files	27

Chapter 4

Debugging and analyzing files

Debugging scripts	29
Disabling script steps	30
Using the Data Viewer	31
Using the Database Design Report	33

Chapter 5

Developing third-party FileMaker plug-ins

About external functions	35
About the example plug-in	35
Contents of the FMExample folder	36
Contents of the Example folder	36
Contents of the Support folder	36
Installing, enabling, and configuring the example plug-in	37
Description of the FMExample plug-in's external functions	38
Using the example plug-in	39
Customizing the plug-in example	40
Customizing the example resources	40
Customizing FMPluginExample.cpp	40
Customizing FMPluginPrefs.cpp	41
Customizing FMPluginFunctions.cpp	41
Requirements for writing external function plug-ins	41
API code files	41
Option string syntax	42
Naming conventions for external functions	42
FileMaker messages sent to the plug-in	42
Initialization message	43
Shutdown message	43
Idle message	44
Preferences message	44
External Function message	44
GetString message	45
Avoiding potential Mac OS X resource conflicts	45
Providing documentation for your plug-in	45

Appendix A

Feature comparison of the runtime application with FileMaker Pro

Application and document preferences	48
Menu command comparison	49
Ignored script steps	54
Stored registry settings or preferences	55

Index

57

Chapter 1

Introducing FileMaker Pro Advanced

Welcome to FileMaker® Pro Advanced. This product includes advanced development and customization tools designed especially for database developers. You can use either FileMaker Pro or FileMaker Pro Advanced to create and test your database solutions.

In addition to all of the features that are available with FileMaker Pro, FileMaker Pro Advanced includes:

- Developer Utilities, for creating, customizing, and deploying runtime database solutions
- Database Design Report feature, for publishing comprehensive documentation on structures or schemas of databases
- Script Debugger, for systematic testing and debugging of FileMaker scripts
- Data Viewer, for monitoring fields, variables, and calculations
- Copy feature, for copying fields or tables. You can also import table schema for use within the same file or across different files.
- Custom Menus feature, for creating customized menus for the solution
- Custom Functions feature, for creating custom functions for use anywhere within the solution

About this guide

This *Development Guide* includes information about features that are available with FileMaker Pro Advanced. In addition, this guide gives an overview of how to create custom layout themes and external function plug-ins.

See FileMaker Pro Help for detailed information on product features.

To send your feedback on this guide, see www.filemaker.com/company/documentation_feedback.html.

Using the FileMaker Pro Advanced documentation

This *Development Guide* is one component in a comprehensive documentation suite provided with FileMaker Pro Advanced. Some of the documents are provided in print and in portable document format (PDF), while others are available in PDF only. FileMaker Pro Advanced also includes an online Help system to provide details on FileMaker Pro features.

This guide assumes that you are familiar with FileMaker Pro or FileMaker Pro Advanced, and that you have created a database solution that you want to work on using the FileMaker Pro Advanced features. If you are new to the FileMaker family, start with the *FileMaker Pro User's Guide*.

The following manuals are included:

- *FileMaker Pro Advanced Development Guide* (this manual): describes how to use the features available in FileMaker Pro Advanced
- *Installation and New Features Guide for FileMaker Pro and FileMaker Pro Advanced*: contains installation instructions and a list of the new features in the current version
- *FileMaker Pro User's Guide*: contains key concepts and basic procedures

- *FileMaker Pro Tutorial*: contains step-by-step lessons that teach you how to create and use FileMaker Pro databases
- *FileMaker Pro Advanced Database Design Report XML Output Grammar* manual: describes the FileMaker Pro Advanced Database Design Report (DDR) XML output grammar for users who want to create tools that analyze or process the structure of databases
- *FileMaker Instant Web Publishing Guide*: describes how to make FileMaker Pro and FileMaker Pro Advanced databases accessible to web browser users over an intranet or the internet
- *FileMaker ODBC and JDBC Guide*: describes how to share FileMaker data with other applications using ODBC and JDBC.

Where to find PDF documentation

To access PDFs of FileMaker documentation:

- in FileMaker Pro Advanced, choose **Help** menu > **Product Documentation**
- click the **Learn More** button in the Quick Start Screen
- see www.filemaker.com/documentation for additional documentation

Most PDF manuals are located in the folder where you installed FileMaker Pro Advanced. If you installed FileMaker Pro Advanced in the default folder location, the PDF manuals are located here:

- **Windows:** C:\Program Files\FileMaker\FileMaker Pro Advanced\English Extras\Electronic Documentation
- **Mac OS:** Macintosh HD/Applications/FileMaker Pro Advanced/English Extras/Electronic Documentation

To view the PDF files, you need a PDF reader. In Mac OS X, you can use either the built-in Preview application or Adobe Reader. Windows users need Adobe Reader. If you do not have Adobe Reader, you can download it from the Adobe website at www.adobe.com.

All of the PDF files use the tagged Adobe Portable Document format (PDF). Tagged PDF files work with assistive technology such as the screen readers JAWS and Window-Eyes for Windows. For more information about tagged PDF files, see the Adobe website at www.adobe.com.

Abiding by the license agreement for runtime solutions

The FileMaker Pro Advanced license agreement allows you royalty-free distribution of an unlimited number of FileMaker Pro runtime database solutions. However, there are several terms and conditions in the license agreement you must abide by, including the following:

- You must provide all of the end-user technical support.
- You must provide an “About” layout that includes your name, address, and the telephone number for your technical support. For more information about creating an About layout, see the next section.

Note You must read and agree to the terms and conditions of the FileMaker Pro Advanced license agreement, available through the FileMaker Pro Advanced installer, before using the FileMaker Pro Advanced software.

Your responsibilities as a developer

FileMaker has established procedures for repairing files. If a customer complies with these procedures, then FileMaker may supply a repaired file to the customer.

If you distribute database files with passwords or you have removed full access privileges and do not want FileMaker to repair a file for a customer who requests this service, you must:

1. Notify your customers in writing and keep a record of such notice that your database solution contains passwords or data that can only be provided by you.
2. Every file in your runtime database solution must contain an About layout accessible from any layout in the database.
3. The layout name must begin with the word “About.”
4. The About layout must contain these items:
 - solution name
 - your company name and contact information
 - your support policy (for example, how and when you are available for technical support)
5. The About layout must contain this exact warning:

“USER WARNING: This database solution contains password(s) that can only be provided by the Developer identified above.”

For more information about creating an About layout, see Help.
6. If full access privileges have been permanently removed from your database solution by selecting the Remove admin access from files permanently option in the Developer Utilities, then the About layout must contain this exact warning:

“USER WARNING: This file is not customizable. Contact the above named Developer for information on customizing this database solution.”

The accounts and privileges protection in a FileMaker file should not be viewed as an absolute barrier that will prevent a customer from accessing files. FileMaker cannot guarantee that a customer will not be able to identify or bypass the password through third-party solutions or tools. Therefore, FileMaker recommends that you take appropriate steps to protect your consulting and development efforts without relying solely upon the password. For more information about accounts and privileges, see Help.

If you have a dispute with your customer, you must resolve this dispute directly with the customer. FileMaker is unable to, and will not, attempt to resolve such disputes.

Chapter 2

Creating database solutions

FileMaker Pro Advanced provides Developer Utilities that let you:

- rename a set of database files and automatically update the internal links to related files and scripts
- bind your database files into a stand-alone runtime database solution that does not require FileMaker Pro or FileMaker Pro Advanced in order to be used on a computer
- remove administrative access from all accounts and prevent users from modifying most design or structural elements of your databases
- display your database files in Kiosk mode
- add the FileMaker Pro filename extension to your files

Note See Help for detailed, comprehensive information and step-by-step procedures about using FileMaker Pro Advanced.

Using the Developer Utilities

To customize your database files or bind the files to a runtime solution:

1. Close all of your database files that you are going to customize.
2. Choose Tools menu > Developer Utilities.
3. If you have used the Developer Utilities on the same database before and saved your settings, click Load Settings.
A dialog box opens so that you can browse to find your settings file.
4. Click Add to locate the files that you want to customize.
5. If you are binding multiple files into a runtime solution, double-click a file in the list to specify the primary file.
6. To rename a file, select the file in the list, type the new name in the Rename file box, and click Change.
7. To remove a file, select the file in the list and click Remove.
8. Under Project Folder, click Specify to choose the location in which the copy of the database solution will be saved.
9. If you do not want the new files to overwrite earlier versions, clear Overwrite matching files within the Project Folder.

Important If Overwrite matching files within the Project Folder is selected, the Developer Utilities will overwrite files with the same names as those in the list of files.

10. Do one of the following:

- If you want to create a copy of your database files with new names, click **Create**.

Note FileMaker Pro Advanced automatically updates internal links to related files and scripts.

- If you want to further customize your database files or bind the files, under **Solution Options**, click **Specify**.

11. In the Specify Solution Options dialog box, select one or more options:

To	Do this
Bind databases to runtime applications	Select Create Runtime solution application(s) . Note This option can be combined with all others, except Databases must have a FileMaker file extension . See “About creating runtime solutions” on page 10.
Permanently prohibit any administrative access to your solution	Select Remove admin access from files permanently . Important Once removed, administrative access cannot be restored to the custom solution. For more information about removing Admin access to databases, see Help.
Force accounts without full access privileges to open your solution in Kiosk mode	Select Enable Kiosk mode for non-admin accounts . See “Creating Kiosk solutions” on page 17.
Add the FileMaker extension to the filenames of database files	Select Databases must have a FileMaker file extension . Note This option is not available if you select Create Runtime solution application(s) . You can use this feature to add extensions to files that do not have extensions.
Create a log file to record any errors encountered during processing	Select Create Error log for any processing errors . Specify a location and a filename for the error log. Notes <ul style="list-style-type: none"> ■ If you don't specify a filename and location for the error log, it will be saved to the project folder with the filename Logfile.txt. ■ If an error occurs during the processing of the options, the error is logged in the error log. An error message may also indicate that an error has been encountered.

12. Click **OK**.**13.** To be able to quickly repeat the process, click **Save Settings**, and choose a folder and location for your settings file. For more information about saving solution settings, see Help.**14.** Click **Create**.

About creating runtime solutions

Use the Developer Utilities to produce a stand-alone runtime database solution that users can access without running FileMaker Pro or FileMaker Pro Advanced. The Developer Utilities create a copy of your files, and bind the database file or files to a runtime application with a name that you specify.

Runtime applications do not have all the functionality and features of FileMaker Pro. For a complete list of the differences between the runtime application and FileMaker Pro, see appendix A, “Feature comparison of the runtime application with FileMaker Pro.”

You may need to bind your database files several times before you prepare them for delivery to your users. When you have completed development and the final version is bound and ready to distribute, you should thoroughly test your runtime solution to ensure that it behaves as expected.

Note FileMaker Pro and FileMaker Pro Advanced allow you to include as many database tables as you need in a database file. This capability eliminates one of the main reasons for using multiple files. However, other elements, like scripts and access privileges, are stored at the file level and so some complex solutions will still benefit from using multiple files.

Before you begin to build your database solution, you need to decide how users will interact with it. Your database solution might have any of the following components:

- a primary database file that connects all of the auxiliary files
- scripts and buttons to open and close auxiliary files, return to the primary file, display a splash screen layout at startup, or quit a runtime application
- common elements and a consistent appearance for cross-platform solutions
- tooltips and custom menus
- a custom layout theme used for every file in the solution
- an About layout to introduce your solution (required)
- a custom Help system that provides usage tips for your solution
- multiple privilege sets that can specify levels of access to layouts, menus, specific tables, record, fields, and so on
- password-protected accounts assigned to privilege sets that determine the level of access of account users

For information about what users need in order to use your runtime database solution, see “Distributing runtime solutions” on page 13.

Converting and upgrading previous solutions

If you have developed a FileMaker Pro runtime database solution using the Solutions Development Kit (SDK) for FileMaker Pro 3.0 or earlier, the Binder utility in the FileMaker Pro 4.0 Developer Edition, or the Developer Tool in FileMaker Developer 5.x and 6.0, you can upgrade your solution and provide your users with the converted files. Files bound to a runtime application using the earlier tools must be rebound using the Developer Utilities.

You must convert FileMaker Pro files created in version 6.0 or earlier to the new file format. You can convert a single file or convert multiple files at once. For more information about converting files, see Help.

Once you have converted the files, you can upgrade them to take advantage of newer FileMaker Pro and FileMaker Pro Advanced features. If necessary, create scripts to import users’ existing data from the old runtime database solution into the new, upgraded solution. See Help for more information on importing data into upgraded runtime solutions. Use the Developer Utilities to bind the solution files into a new, upgraded runtime database solution.

Distribute the new upgraded runtime database solution and provide instructions for how users can upgrade their files by converting the old files in the new runtime application and importing their data.

Binding the solution

To bind database files into a runtime database solution:

1. Follow the procedures in “Using the Developer Utilities” on page 9.
2. In the Specify Solution Options dialog box, select **Create Runtime solution application(s)**.
3. To name your runtime application:
 - For **Runtime Name**, type a name. The name is used for the runtime application filename and for the name of the folder that contains the runtime database solution files.
 - For **Extension**, type a three-character filename extension. The extension is used to associate the solution files with the runtime applications.

For more information about naming runtime solutions, see [Help](#).

4. For **Bindkey**, type a key between 1 and 24 characters long.

The binding key links the runtime application to the database files and ensures that the bound files will only open in the appropriate runtime application. The binding key is case-sensitive. For more information on setting the binding key, see [Help](#).

Important Binding installs system files pertaining to each platform. If your solution will be used in Windows, bind it using the Developer Utilities for Windows. If your solution will be used on Mac OS X, bind it using the Developer Utilities for Mac OS X. If you’re creating a solution to be used on both Windows and the Mac OS X, create two separate runtime solutions by binding the original solution files twice: first using FileMaker Developer Utilities for Windows, and then using FileMaker Developer Utilities for Mac OS X. Use the same binding key on both platforms.

5. To add a company logo or other custom image to the closing splash screen, click **Specify**, select the closing image, and click **Select**.

The image should be at least 32 x 175 pixels (72 dpi) or higher, otherwise it will be distorted when displayed. The supported image formats are JPEG and GIF.

6. For **Delay**, set the number of seconds that you want the splash screen to display.

You can preview the effect that your custom splash screen will have by clicking the **Preview** button.

7. Once you have specified options, click **OK**.

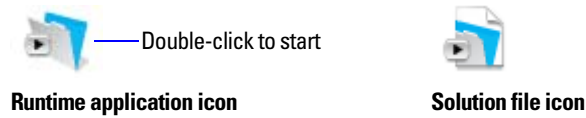
8. To be able to quickly repeat the process, click **Save Settings**, and choose a folder and location for your settings file.

For more information about saving and reusing Developer Utilities settings, see [Help](#).

9. Click **Create**.

The Developer Utilities copy all of the runtime files to a new folder created inside the Project Folder and named after the runtime solution.

Starting runtime database solutions



Important Your users should start your solution by double-clicking the runtime application icon, not the solution file icon. Double-clicking the icons for the solution or auxiliary files might result in errors, depending on whether there are other copies of the runtime application on their hard disk. If your users have more than one solution on their computers associated with the same three-character extension and they double-click the icon for the solution file, the first solution installed will attempt to open the file, and this might not be the correct application for the specific file.

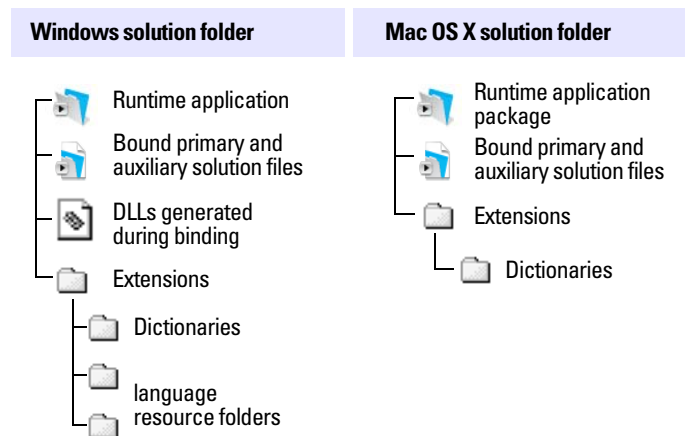
Distributing runtime solutions

The final steps in developing your runtime database solution are to bundle all of the necessary files together, choose how you will distribute your solution—for example, on a CD-ROM or over a network—and provide your users with documentation for installing your solution. In addition, your documentation should include instructions for starting the runtime application and what to do if a file is damaged.

Organizing solution components

When you bind your database files into a runtime database solution, the Developer Utilities create a new solution folder and place the runtime application, the bound primary and auxiliary database files, and an Extensions folder inside it. For Windows runtime solutions there are also required Dynamic Link Library (DLL) files.

Note When you move multiple files into one folder to create a runtime solution, be aware that your internal links are affected. For this reason, every data source must include a path that is just the filename of the file being referenced. Although the runtime application will check other data sources, it will then be able to find the file in the same folder in which it resides. You can still keep any absolute or relative paths in the same data source reference in case the files are also used in FileMaker Pro or FileMaker Pro Advanced.



Example of Windows and Mac OS X solution contents for distribution

Important These files and folders must not be renamed.

For details on the contents of the Mac OS X runtime application package and the Windows Extensions folder and DLLs, see Help.

If your runtime database solution requires custom files, you should provide the files with the runtime files. Plug-ins should be stored in the Extensions folder. If a developer uses a font not found on a user's system, the runtime application will make a font substitution. If a font is included with the runtime, provision should be made for its installation through the installer program. See "Using a custom installation program" below.

In addition to the runtime files, you will need to provide installation instructions for your users. For more information on documenting developer solutions, see Help.

Choosing a distribution method

After you have organized the files that comprise your solution, you need to decide how your users will install them. You can distribute your bundled solution on a CD-ROM, over a network, or via the internet. In order to run your runtime database solution, your users will need the same minimum equipment and software required by the FileMaker Pro Advanced application. See the *Installation and New Features Guide for FileMaker Pro and FileMaker Pro Advanced*.

Using a custom installation program

You should use a custom installation program to package your runtime solution for installation by users. Configuring a custom installation application to install runtime database solution files may require more engineering than using a compression utility, but will help to ensure that your users do not have difficulties installing your runtime solution.

Here are some custom installation applications you might want to use:

- MindVision Installer VISE
- InstallShield MultiPlatform
- MacInstallerBuilder

Using a compression utility program

If your runtime database solution is not complex and you have confidence in the technical experience of your end users, you might consider a compression utility program rather than a custom installation program.

Sharing solutions over a network

Users cannot share your runtime database solution over a network unless they access the files using FileMaker Pro or FileMaker Pro Advanced installed on their machines. You must have a master password to enable or change network access to the file. For optimal performance, you can host the solution files using FileMaker Server.

For information about the FileMaker Server and FileMaker Pro products, and information about volume license sales, see the FileMaker website at www.filemaker.com.

Recovering damaged files

Power failures, hardware problems, or other factors can damage a FileMaker database file. If your database solution becomes damaged, your users will need to recover the damaged file. When the runtime application discovers a damaged file, a dialog box appears, telling the user to contact the developer. Even if the dialog box does not appear, files can become corrupted and exhibit erratic behavior.

For information about recovering runtime files, see Help.

Creating an About layout

For runtime database solutions, the FileMaker Pro Advanced license specifies that you must create an About layout that provides information for your users on how to contact you for technical support. FileMaker uses the About layout to distinguish databases created by developers using FileMaker Pro Advanced rather than users of FileMaker Pro.

For more information about what is required to appear in the About layout for runtime database solutions, see “Your responsibilities as a developer” on page 7.

Creating a custom Help layout

The FileMaker Pro Advanced Help system is not available in runtime applications.

Create a Help layout that provides instructions for how to use your custom solution and add data to it. Then create a script in the primary file of your solution to display the Help system. Use the custom menus feature to make the script available as a command in the Help menu.

To create a web page to document your solution, put a web viewer in your Help layout that opens the web page.

To display your custom Help menu on Mac OS, you must start with an empty menu. For more information about creating and editing custom menus, see “About custom menus” on page 20.

Testing before and after creating your solution

You should verify the functionality of your database solution by testing it thoroughly before and after you customize it with the Developer Utilities.

To ensure the quality of your custom database solution:

- Verify every function and option in your solution. If you’re developing a solution for both platforms, test it on both Windows and Mac OS X platforms.

- Make sure your runtime database solution does not use a standard FileMaker Pro feature that is hidden or disabled in the runtime application. See appendix A, “Feature comparison of the runtime application with FileMaker Pro.”
- Verify that all scripts and buttons work as expected. This is especially important if you’re displaying your solution in Kiosk mode. See “Creating Kiosk solutions” on page 17.
- Verify your installation procedures and test other instructions in the documentation.
- Verify that your database layouts display well on monitors with different color capabilities and resolutions and on the smallest size monitor your users may be using.
- Test your runtime database solution with actual data. This is especially important if users are upgrading from earlier versions of the runtime application and need to import data into new solution files.
- Make sure all the auxiliary files and DLLs (Windows) are present.
- Show your database solution to intended users to uncover any usability issues.
- Install your bundled database files on a completely different computer to verify that all the files associated with the primary file can be found.
- If you’re assigning passwords or permanently removing full access privileges, test all access levels.
- Make sure your database solution contains an About layout that notifies users of the level of access you’re providing.

Important You should keep an unbound version of any runtime database solution files, especially if you’ve permanently removed full access privileges.

Distributing updates to runtime database solutions

If you make feature enhancements or modifications to the primary bound file of your runtime database solution, you can distribute the updated file to your users without rebinding it. If you change the filename of the primary file, however, you’ll need to rebind the file and distribute a new version of the runtime application along with the updated file.

To distribute new or updated auxiliary files for your runtime database solution, bind them first using the original binding key. If you are distributing a new auxiliary file that requires new data sources in the main file or that requires other files to interact with it, you must update all files that have been modified.

If you forget the original binding key for your runtime database solution and want to update or add a file, you’ll need to rebind all of the database files with a new binding key and redistribute the entire solution.

To distribute an updated primary file:

1. Open the original primary file from your copy of the runtime solution in FileMaker Pro Advanced.
2. Make the changes to the primary file.
3. If necessary, create an Import script so users can import their existing data into the new primary file.
For more information about importing data into upgraded runtime solutions, see Help.
4. Send your users a copy of the new primary file with instructions to replace the old primary file in the runtime database solution folder.

To distribute a new or updated auxiliary file:

1. In FileMaker Pro Advanced, create the new auxiliary file or open the original auxiliary file (before it was bound) and make changes as required.
2. If necessary, create an Import script so users can import their existing data into the new file.
For more information about importing data into upgraded runtime solutions, see Help.
3. Use the Developer Utilities to rebind all of the files in the runtime database solution and include the new or updated auxiliary file.
Use the same binding key that you used for the primary file.
4. Send your users a copy of the new or updated auxiliary file along with instructions to place it in the runtime database solution folder, replacing the old file if appropriate.
As long as the binding key has not changed, you don't need to redistribute the runtime application or other solution files.

Creating Kiosk solutions

Kiosk mode is a way of displaying your database solution or your runtime database solution on a full screen, without any toolbars or menus. As the name suggests, Kiosk mode can be used to present your database to users as an information kiosk. You can design your database to run through a touch screen.

Kiosk mode is ignored if the solution is opened by accounts with the Full Access privilege set, a privilege set that allows management of extended privileges, or a privilege set that allows modification of layouts, value lists, and scripts.

For your solution to display in Kiosk mode, you must:

- create an account with a limited privilege set or create a specific Kiosk account.
- enable Kiosk mode. At the same time that you enable Kiosk mode, you can bind the database as a runtime solution.
- clear the default option of logging into the file with the Admin account.

To create a Kiosk account:

1. Ensure you have a limited access account.
2. With the database solution open, choose File menu > Manage > Accounts & Privileges.
3. In the Manage Accounts & Privileges dialog box, click **New**.
4. In the Edit Account dialog box, type an account name, click **Active** for the Account Status, and select **New Privilege Set** from the Privilege Set list.
5. In the Edit Privilege Set dialog box, give the privilege set a name and description.
6. For Layouts, Value Lists, and Scripts, select either **All view only** or **All no access**.
7. Clear the **Manage extended privileges** checkbox.
8. Select other options as required and click **OK**.

To enable Kiosk mode:

1. Follow the procedures in “Using the Developer Utilities” on page 9.
2. In the Specify Solution Options dialog box, select Enable Kiosk mode for non-admin accounts.
3. Select other options as required click OK.
4. To be able to quickly repeat the process, click **Save Settings**, and choose a folder and location for your settings file.

For information on saving solution settings, see [Help](#).

5. Click **Create**.

If you did not bind the files to a runtime application, the Developer Utilities copy the selected database files to the Project Folder. If you did bind the files to a runtime application, the Developer Utilities copy all of the runtime files to a new folder created inside the Project Folder and named after the runtime solution.

To change the default option of logging into the file with the Admin account:

1. With the database solution open, choose **File menu > File Options**.
2. On the Open/Close tab, clear **Log in using**.
3. Click **OK**.

When you create a solution to run in Kiosk mode, you need to provide navigation for your solution and the ability for users to quit your solution.

Note If you have a previous Kiosk solution that displayed the status area, you will need to update your solution. You cannot display the status toolbar or layout bar in a Kiosk solution. You will need to add record navigation, script paused status, and script Cancel and Continue buttons to your layouts.

For more information on using scripts and buttons to control Kiosk solutions, see [Help](#).

Chapter 3

Customizing database solutions

You can use FileMaker Pro Advanced to customize your solutions beyond what is possible with FileMaker Pro. You can:

- copy and paste fields for use within the same file or other database files
- copy or import existing tables into your database file
- create custom functions for use anywhere within a file
- create custom menus
- create custom layout themes

Important You must have full access privileges to customize database solutions.

Note See Help for detailed, comprehensive information and step-by-step procedures about using FileMaker Pro Advanced.

Copying or importing field and table schemas

You can copy or import field and table schemas within a file or to other database files.

With FileMaker Pro Advanced, you can consolidate tables from a multi-file solution into one file. There are two methods for consolidating solutions:

- Copy table schemas — Open source files to select and copy the tables you want. Then, paste the table schemas into the destination file.
- Import table schemas — Import table schemas directly into the destination file. You can import just the schemas or import data with a single schema. (To import the data with a single schema, choose File menu > Import Records > File.)

To	Do this
Copy a field schema	Choose File menu > Manage > Database > Fields tab. Select the field from the list, then click Copy. Data is not copied.
Copy a table schema	Open the file that contains the table you want to copy. Choose File menu > Manage > Database > Tables tab. Select the table from the list, then click Copy.
Import a table schema	Open the file into which you want to import a table. Choose File menu > Manage > Database > Tables tab. Click Import. Select the source file and table, and click OK.

For more information about copying or importing fields and tables, see Help.

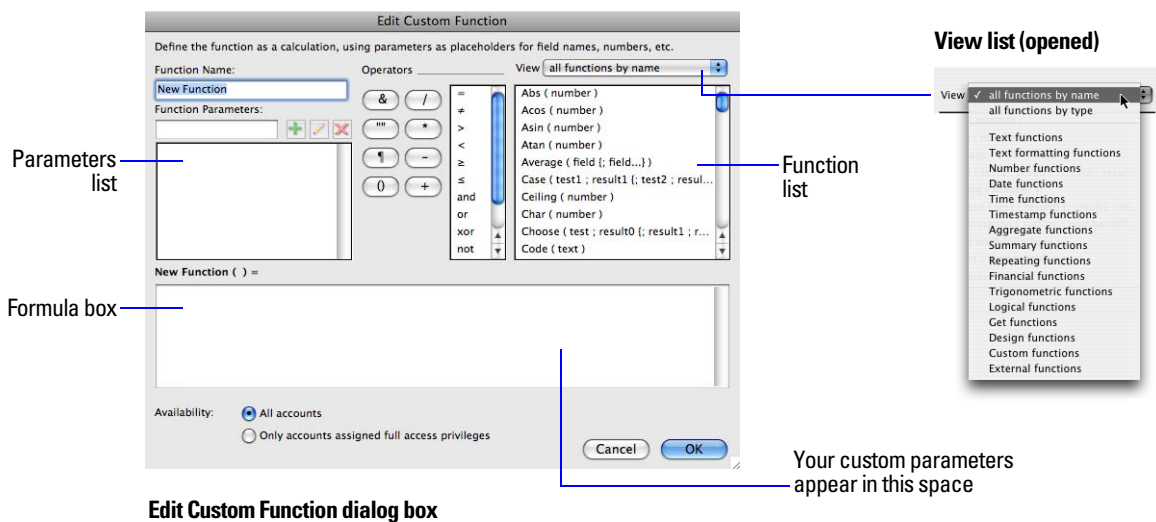
Creating custom functions

Use the Custom Functions feature to create custom functions that can be reused anywhere in the database file in which they are created. Once formulas are written for the function, they don't have to be rewritten to be applied to other fields or used in other scripts.

You can maintain and edit custom functions and the formulas they contain in one central location. Any change made to the custom function will be copied to all instances where that custom function has been used.

To create a custom function:

1. Choose File menu > Manage > Custom Functions.
2. In the Manage Custom Functions dialog box, click New.



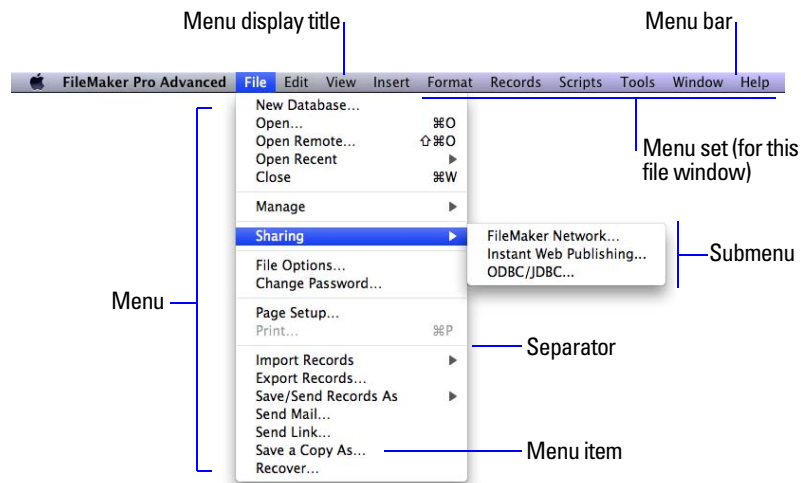
3. In the Edit Custom Function dialog box, type a name for the function and build a formula.
4. Click OK.

For more information about creating custom functions, see Help.

About custom menus

With FileMaker Pro Advanced, you can create custom menus, menu items, and menu sets for your database solutions. You can:

- create a menu or edit an existing menu
- duplicate or delete a menu
- add, duplicate, or delete menu items
- specify menu item properties, such as display title, shortcut, and action.



Custom menu terminology

You can customize menus by:

- editing a copy of a standard FileMaker menu. Use this method to make minor changes to existing menus, for example, to modify the properties of a few menu items.
- starting with an empty menu. Use this method to make significant changes to menus, for example, to add menus and change menu item properties.

Custom menu example

The following example shows how to customize the New Record menu item that appears in the Records menu. You can rename the New Record menu item to New Invoice, then attach a script that runs when the user chooses the New Invoice menu item. Finally, you can change the default menu set so your new custom menu set displays when a user opens the database.

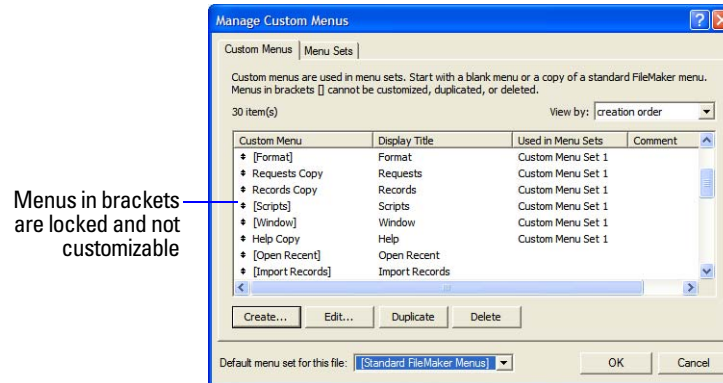
This example assumes the database contains a script called My New Invoice. My New Invoice automates several tasks, like switching to the Invoices layout and creating an empty record.

1. Open the database and choose File menu > Manage > Custom Menus. Double-click the Records Copy menu to edit a copy of the standard Records menu.
2. Select the New Record menu item to modify the properties of the menu item. Change the title of the New Record menu item to New Invoice.
3. Change the action of the menu item to run the My New Invoice script.
4. Set the default menu set for the file to Custom Menu Set #1.
5. Click OK.

Creating custom menus

To create a custom menu:

1. Choose File menu > Manage > Custom Menus > Custom Menus tab.



2. Click Create.
3. In the Create Custom Menu dialog box, do one of the following, then click OK:
 - Click Start with an empty menu.
 - Click Start with a standard FileMaker menu, then select a menu from the list.

The Edit Custom Menu dialog box appears.

4. In the Edit Custom Menu dialog box, specify a menu name, the title that you want to display in the menu bar, the menu platform, and the modes in which to display the menu. See Help for more information on these options.
5. Create custom menu items, as described below.

Creating custom menu items

After you create a menu, you can create menu items. You can also create or edit menu items that are copies of the Standard FileMaker menus. Menu items can be commands, submenus, or separators. You can create a menu item that is based on a standard FileMaker command or you can create a menu item that initially does not have an assigned command.



When you base a menu item on a FileMaker command, that menu item inherits all the properties of that command. You can override properties (title, shortcut, or action) to customize the menu item.

When you create a menu item that does not have an assigned command, an untitled menu item appears in the Menu Items list of the Edit Custom Menu dialog box. You can then customize the menu item properties.

To create or edit a custom menu item:

1. Choose File menu > Manage > Custom Menus > Custom Menus tab.
2. In the Manage Custom Menus dialog box, select the menu from the list, then click Edit.

3. In the Edit Custom Menu dialog box, specify which menu items are included in the menu:

To	Do this
Add a command	<p>Click Command.</p> <p>In the Specify FileMaker Command dialog box, do one of the following, then click OK.</p> <ul style="list-style-type: none"> Click No command assigned. FileMaker Pro Advanced adds an Untitled menu item to the Menu Item list. You must specify properties for this command (see Stepstep 4). Click Use a FileMaker command, then select a command from the list. <p>The command determines the action or behavior of the menu item.</p> <p>Tip From the Edit Custom Menu dialog box you can press Shift and click Command to create an Untitled command menu item.</p>
Add a submenu	<p>Click Submenu to add another menu's commands to the current menu.</p> <p>In the Specify Submenu dialog box, select the menu you want to include, then click OK.</p> <p>Note You can add up to 100 menus to the menu bar. If you add a menu that includes itself as a submenu, you may quickly reach the limit.</p>
Add a separator line	<p>Select the menu item from the list under which you want the separator line to appear, then click Separator.</p>
Duplicate a menu item	<p>Select a menu item from the list, then click .</p>
Delete a menu item	<p>Select a menu item from the list, then click .</p>

4. Select each menu item from the Menu Items list and specify its properties.

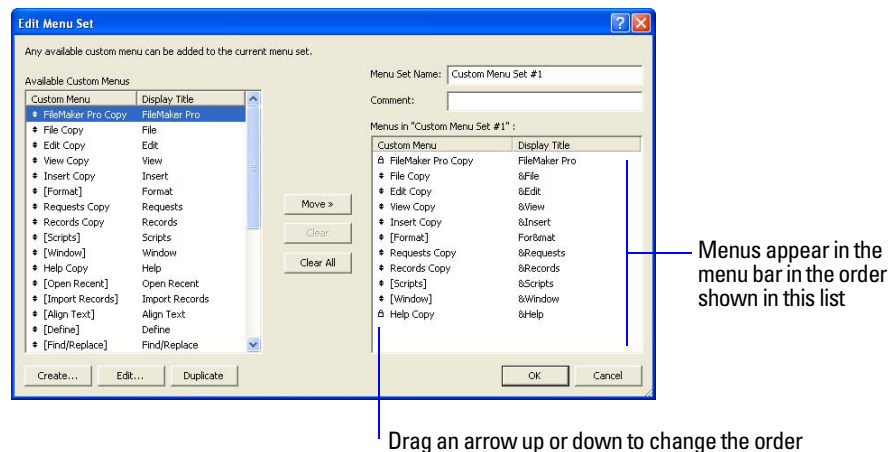
To	Do this
Change a command for a menu item	<p>Click Specify. In the Specify FileMaker Command dialog box, select a command, then click OK.</p>
Choose a platform for a menu item	<p>Select Windows, Macintosh, or both. Your menu item will appear on the platforms you select.</p> <p>Note Some commands are valid on only one platform.</p>
Change the title for a menu item	<p>Select Title and enter the text. To specify an access key (Windows), type an ampersand (&) before the character you want to use as the access key. For example, type &Open to display the Open menu item with the letter "O" as the access key.</p> <p>To base the menu title on the result of a calculation, click Specify, then build a formula in the Specify Calculation dialog box.</p> <p>Note Untitled menu items do not have a checkbox.</p>
Define a shortcut for a menu item	<p>Select Shortcut. In the Specify Shortcut dialog box, type the key combination, then click OK.</p> <p>For more information about keyboard shortcuts, see Help.</p>
Perform a script or script step when a user selects a menu item	<p>Select Action, do one of the following, then click OK.</p> <ul style="list-style-type: none"> Select Script. In the Specify Script Options dialog box, select a script and enter the optional script parameters as necessary. Select Script Step. In the Specify Script Step dialog box, select a step and specify options as necessary, then click OK. <p>Note Untitled menu items do not have an Action checkbox.</p> <p>Tip To affect the behavior of a currently running script (for example, to halt, exit, resume, or pause the script), choose Script Step and then use the Perform Script script step.</p> <p>For more information about scripts and script steps, see Help.</p>

Creating custom menu sets

You can create custom menu sets to include the menus you require.

To create or edit menu sets:

1. Choose File menu > Manage > Custom Menus > Menu Sets tab.
2. Create a menu set, or edit or duplicate an existing menu set. You can also delete menu sets that your users will not need.
3. Specify which menus to include in the menu set, and click OK.



Important Duplicating a custom menu set creates a copy of the menu set and references the same custom menus as the original menu set. It does not duplicate the custom menus. If you change a menu item, you change the menu for all menu sets in which the menu is included.

Once you create menu sets, you must specify options for installing them. You can:

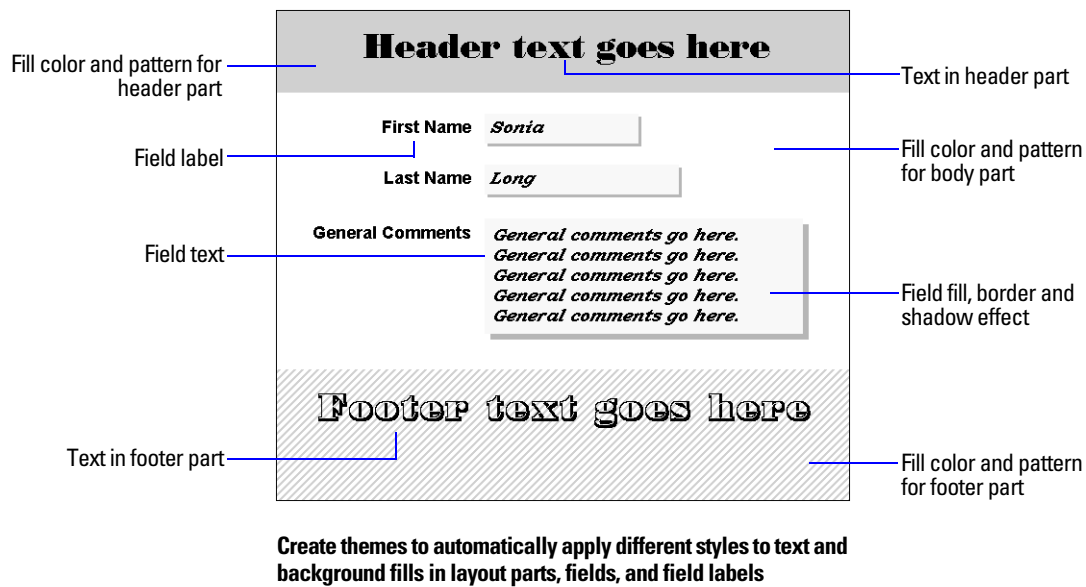
- specify a default menu set for a file
 - specify menu sets for individual layouts
 - create scripts that change menu sets
 - specify menus within the menu set to display according to mode
- switch menu sets using the FileMaker Pro Advanced Tools menu

For more information about creating, installing, and testing custom menu sets, see Help.

Creating custom layout themes

FileMaker Pro and FileMaker Pro Advanced use a variety of layout themes to describe the colors, patterns, fonts, and borders of text, fields, and parts in a new layout.

A theme is an Extensible Markup Language (XML) document that can be read and edited in a text editor (such as Notepad for Windows or BBEdit for Mac OS X) or XML editor (such as XMLSpy or XMetaL). You can customize an existing theme or create your own, and then use the New Layout/Report assistant to apply the custom theme when you create layouts for your databases. You can modify attributes defined by the theme in Layout mode after the layout is created. However, you can't apply a theme to an existing layout.



Note A FileMaker theme is not a stylesheet and does not contain positioning information for objects on a layout.

Important The XML for a layout theme must be well-formed and comply with the required syntax. Omitting a required element or attribute, or mismatching start and end tags will result in an unusable document and FileMaker Pro Advanced will be unable to parse the XML or display the theme in the New Layout/Report assistant.

To create or modify a theme:

1. Make a copy of one of the theme files in the Themes folder.

Windows: FileMaker Pro Advanced\Extensions\English\Themes\

or

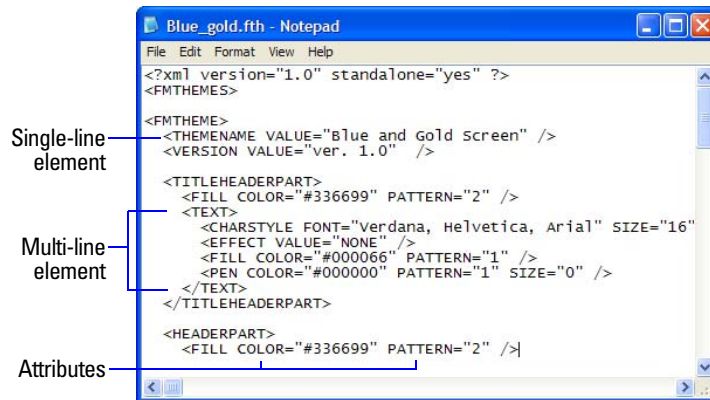
Mac OS X: FileMaker Pro Advanced/FileMaker Pro Advanced.app/Contents/Resources/English.lproj/Themes/

Important The total number of theme files is limited to 50.

2. Rename the copy and include the .fth extension with the new filename.

Keep the new file in the Themes folder. In order for the New Layout/Report assistant to display a theme option, the theme file must reside in the Themes folder and it must have the .fth filename extension.

3. Open the theme file in a text editor.



4. Change the name of a theme by replacing the value of the THEMENAME element with a new name.

```
<THEMENAME VALUE="Purple and White Screen" />
```

Important If your THEMENAME value contains any upper-ASCII characters, use the HINT attribute to ensure that the theme name will appear on both the Windows and Mac OS X platforms.

5. Change the values of other elements and attributes.

For example, to change the background fill color of the body part in a layout to a light purple, change the color hexadecimal (hex) value to #9933CC:

```
<BODYPART>
  <FILL COLOR = "#9933CC" PATTERN = "2" />
```

6. Remove any elements that you don't want to specify.

Be sure to remove the entire single-line or multi-line element including its start and end tags.

7. Scroll down to the next FMTHEME element and repeat these steps to change the THEMENAME value and other elements.

8. Save the file in text format with the filename extension .fth in the Themes folder.

Each new THEMENAME value will appear in the New Layout/Report assistant as a Layout Theme option.

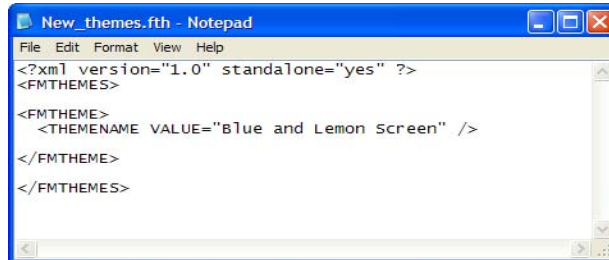
9. In FileMaker Pro Advanced, choose Layouts menu > New Layout/Report to use your theme.

Follow the instruction in the New Layout/Report assistant. Names of custom themes appear as options. The third panel presents you with a list of themes to select from.

If your new themes don't appear in the New Layout/Report assistant, you might have made a syntax error.

Requirements for theme files

Every theme file must begin with an XML-document processing instruction that declares it as an XML document using the XML 1.0 specification. In addition, an XML document for a layout theme must contain the `<FMTHEMES>` and `</FMTHEMES>` start and end tags for the file. This `FMTHEMES` root element can contain one or more `FMTHEME` elements.



Minimum elements required for a theme file

For more information on theme elements and attributes, see Help.

Chapter 4

Debugging and analyzing files

The FileMaker Pro Advanced features explained in this chapter are:

- the Script Debugger for systematic testing and debugging of FileMaker scripts
- the Disable script step feature for testing portions of a script
- the Database Design Report feature for publishing comprehensive documentation on database schema and options
- the Data Viewer for monitoring fields, variables, and calculations

Note See Help for detailed, comprehensive information and step-by-step procedures for using FileMaker Pro Advanced.

Debugging scripts

With FileMaker Pro Advanced, you can use the Script Debugger to:

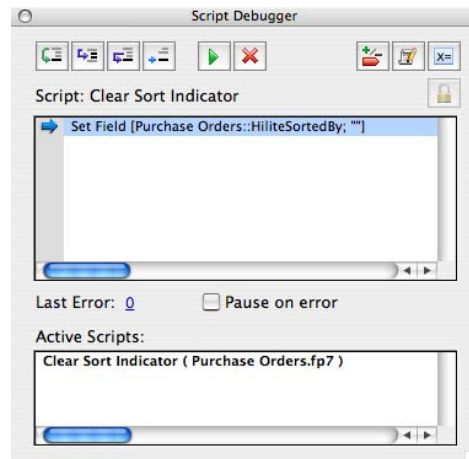
- debug startup scripts (the Script Debugger menu is enabled even if there are no open files)
- debug a script activated by a script trigger
- step through scripts one step at a time
- view and track sub-scripts
- monitor fields, variables, and calculations
- disable script steps
- debug restricted-access scripts
- pause a script when script errors are encountered
- click a script error number to open a Help topic

To run scripts in debug mode:

1. Select Tools menu > Script Debugger.

The Script Debugger dialog box opens.

2. Run your script.



You can view sub-scripts when you step through scripts in the Script Debugger. For example, if Script A calls Script B, which then calls Script C, you can view the steps in all three scripts.

The Script Debugger recognizes the privileges attached to each script. A script will only appear in the Script Debugger if you have editing privileges for the script and the access privileges for the script are set to Modifiable. You can click Authenticate/Deauthenticate script to log in and edit script steps in restricted-access scripts.

In the Script Debugger window you can select more than one step from the step list, enabling you to place simultaneous multiple breakpoints on steps. If multiple steps are selected, the Set Next Step button will be disabled.

Note When you use the Script Debugger to step through scripts activated by a script trigger, you will not be able to interact with the document windows, move between fields or records, change the data, close the window, or quit. This blocking of interaction only occurs when a script is triggered via some action. When you are debugging a script that is not activated by a script trigger you can interact normally with the document windows, fields, and records. For more information about using script triggers, see Help.

Tip To enable the Script Debugger from the Manage Scripts dialog box, press Shift and click the Perform button. To disable the Script Debugger, press Ctrl (Windows) or Command (Mac OS) and click the Perform button.

Disabling script steps

You can disable and enable script steps to test portions of a script. When you run a script, disabled script steps are skipped.

To disable script steps:

1. Choose Scripts menu > Manage Scripts.
Or, choose File menu > Manage > Scripts.
2. In the Manage Scripts dialog box, double-click the script name.
Or, click the Edit button in the Script Debugger dialog box.
3. In the Edit Script dialog box, select one or more script steps, then click Disable or Enable.


For more information about debugging scripts, see Help.

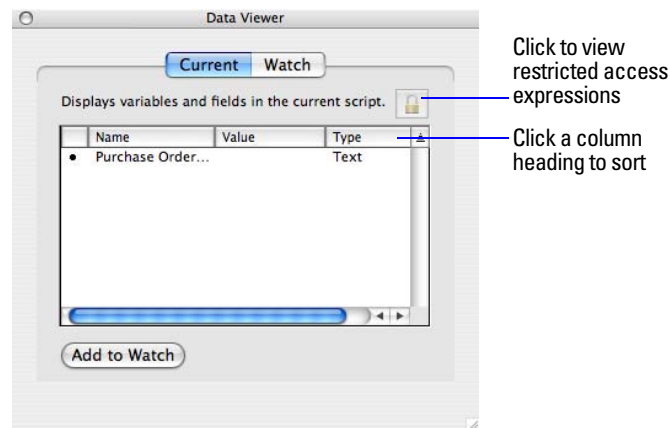
Using the Data Viewer


You can use the Data Viewer to monitor expressions like field values, local and global variables, and calculations. You can monitor these expressions while running scripts or while testing them in the Script Debugger. You can also monitor field values and variables in the database file.

The **Current** tab shows the fields and variables that are in the currently running script, fields that are referenced in calculations used in the script, and global variables. The **Watch** tab monitors expressions that you select until you remove them from the list.

To monitor fields, variables, and calculations:

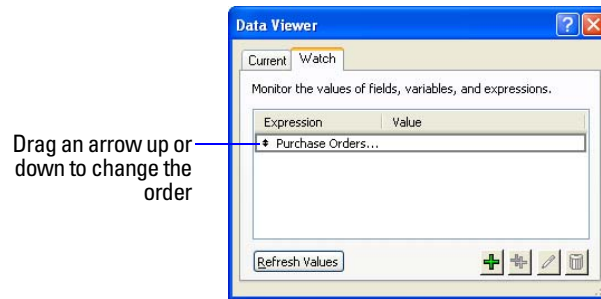
1. Choose **Tools menu > Data Viewer**, or click the Open/Close Data Viewer  button in the Script Debugger window.







2. In the **Current** tab, double-click a value to display a dialog box where you can:
 - View, edit, and copy local or global variables.
 - View (but not edit) field values.
3. To sort expressions, click a column heading. Expressions sort independently in this order: fields, global variables, local variables.
4. To add an expression to the **Watch** tab, click **Add to Watch**.
The expression is copied to the **Watch** tab, and the **Watch** tab opens.
5. To view or edit restricted-access scripts, click , then log into an account that has full access privileges.

Note If you logged in to edit restricted-access scripts in the Script Debugger, your access privileges also apply to the Data Viewer. If you logged in from the Data Viewer, your access privileges also apply to the Script Debugger. In either case, your editing privileges last until you close the Script Debugger or the Data Viewer.

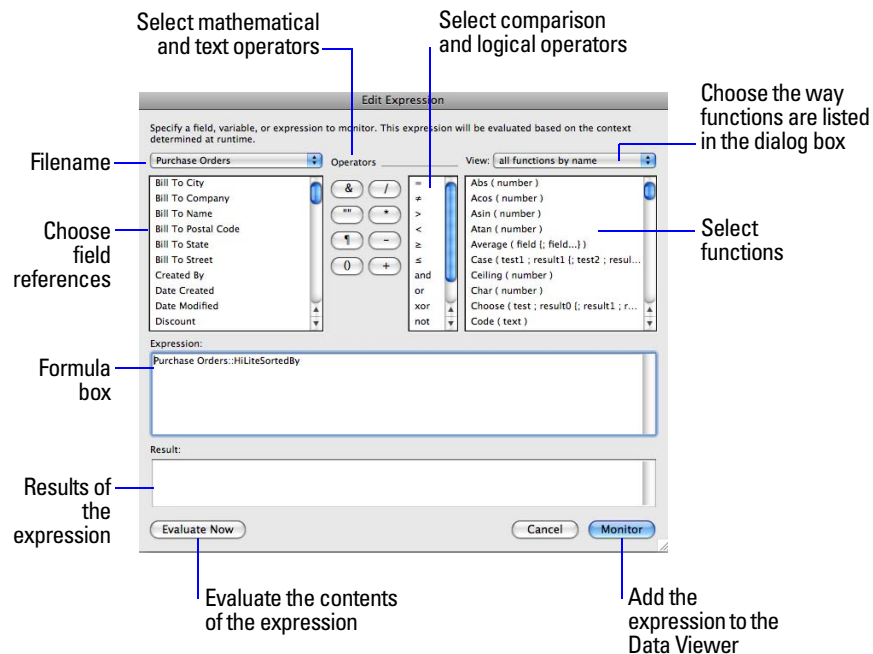
6. Click the Watch tab.



7. Choose one of the following:

To	Do this
Add an expression	Click  .
Edit an expression	Select an expression, then click  or double-click the expression.
Duplicate an expression	Select one or more expressions, then click  .
Delete an expression	Select one or more expressions, then click  .

8. In the Edit Expression dialog box, select the database file containing the expression, then build or edit the expression you want to monitor.



9. Click Evaluate Now to display the results of an expression, or click Monitor to add the expression to the watch list in the Data Viewer.

10. In the Data Viewer, click Refresh Values to refresh the calculations in the list.

For more information about using the Data Viewer, see Help.

Using the Database Design Report

Use the Database Design Report (DDR) feature to document the schema of your database and publish it to an HTML or XML file. You can choose which elements and database tables in the database you want to report. The HTML version of the report is hyperlinked and you can view or print it in a Javascript-enabled web browser.

With the Database Design Report feature you can:

- examine a textual representation of your database schema
- gather statistics on the structure of your database
- use the information in the report to recreate the structure of your database if you lose the original database files
- troubleshoot missing references, broken relationships, calculations, and more

To create a Database Design Report:

1. Open all database files for which you want to produce a Database Design Report.

You must have full access privileges for any file for which you want to produce a Database Design Report and the file must be open in FileMaker Pro Advanced. You can run a Database Design Report on local or remote files.

2. Choose Tools menu > Database Design Report.

3. In the Available Files list, clear any files that you want to exclude from the report by clearing the checkbox associated with the file.

4. If there are any files that contain tables that you want to exclude from the report, select the file in the Available Files list.

The tables in the file appear in the Include fields from tables in selected file list. You can then deselect any table in the list.

By default, all tables in all selected files are reported.

5. Clear elements that you want to exclude from the report.

By default, all elements in all selected files are reported. Each selected element, if present, will be reported on for each selected file.

6. If you prefer to publish the report in XML format instead of the default HTML, select XML in the Report Format section.

7. If you do not want the report to automatically open when done, clear the checkbox for this option in the File Handling section.

8. Click Create.

For more information about using the Database Design Report, see Help.

Chapter 5

Developing third-party FileMaker plug-ins

If you are a C or C++ programmer and familiar with calculations in FileMaker Pro and FileMaker Pro Advanced, you can create external function plug-ins that extend the feature set of the applications. The plug-ins can take advantage of recursion and looping or hook into other programming interfaces. Users can enable your plug-ins in FileMaker Pro, FileMaker Pro Advanced, and FileMaker Server and use your external functions in their calculation fields and scripts.

You can use FileMaker Server to ensure that FileMaker Pro clients always have the most current plug-in software installed on their computers. See FileMaker Server *Guide to Updating Plug-Ins*, available on www.filemaker.com/downloads.

Note See Help for detailed, comprehensive information and step-by-step procedures about using FileMaker Pro Advanced.

About external functions

FileMaker Pro Advanced includes an example plug-in project that you can modify to include your own external functions. Users can access your plug-ins through the Specify Calculation dialog box.

Follow these general steps to prepare your custom plug-ins:

1. Edit the example plug-in files to add your custom programming code.
2. Compile and test the customized plug-in.
3. Install the compiled plug-in file for your users.

To access your external functions, your users:

1. Enable your plug-in through the Preferences dialog box.
2. Configure your plug-in, if required.
3. Define or edit a calculation field.
4. In the Specify Calculation dialog box, choose `Function_Name(parameter 1 ...)` as the calculation formula.

To see all external functions, select **External functions** from the View drop-down list.

About the example plug-in

The example plug-in project is designed to illustrate what a complete plug-in looks like. You can compile the example project files to create a plug-in with several external functions that users can access through the Specify Calculation dialog box. You can examine and modify the source code of the example files in any text editor.

The plug-in example includes several external functions. See “Description of the FMExample plug-in’s external functions” on page 38.

The plug-in example files include all the source code required to compile the plug-in for the Windows and Mac OS X platforms. In addition to the plug-in source code, FileMaker Pro Advanced includes project files for Microsoft Visual Studio 2005 and Apple Xcode 3.0.

The example plug-in files are located in the English Extras\Examples\FMExample folder on the FileMaker Pro Advanced CD or electronic download. The plug-in example source code files are located in subfolders in the FMExample plug-in folder. The following tables describe some of the folders and files.

Contents of the FMExample folder

Folder	Description
Example folder	Contains all of the files that are part of the FMExample.
Headers folder	Contains function definition files for the FileMaker API. Do not distribute to users who do not have licenses for FileMaker Pro Advanced.
Libraries folder	Contains library files for the FileMaker API. Do not distribute to users who do not have licenses for FileMaker Pro Advanced.

Contents of the Example folder

File/Folder	Description
FMPluginExample.cpp	Contains code for implementation of the FMExample.
MacExample.fmplugin	Compiled plug-in for Mac OS X
MacExample.xcodeproj	Apple Xcode project file.
WinExample.sln	Microsoft Visual Studio .NET project file.
WinExample.vcproj	Microsoft Visual C++ project file, used by WinExample.sln.
WinExample.fmx	Compiled plug-in for Microsoft Windows
Support folder	Contains all additional resources and code used by FMPluginExample.cpp.

Contents of the Support folder

File/Folder	Description
FMPluginExample.rc	Contains the resources for Windows platform.
FMPluginExample.nib	Contains the resources for Mac OS X platform.
FMPluginExample.strings	Contains the strings for Mac OS X platform.
FMPluginFunctions.cpp	Contains code for implementation of external functions in FMExample.
FMPluginFunctions.h	Contains definitions for external functions, including function IDs.
FMPluginGlobalDefines.h	Contains constants used by the FMExample, including compiler directives to control code compilation.
FMPluginPrefs.cpp	Contains code for implementation of configuration dialog box in FMExample.
FMPluginPrefs.h	Contains definitions for configuration dialog box.
info.plist	Contains bundle information for Apple Xcode output.

File/Folder	Description
MacExample.plc	Contains bundle definition for Mac platform.
Resource.h	Contains definitions for resource file.

Installing, enabling, and configuring the example plug-in

External function plug-in files must be installed in the appropriate folder and enabled in FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server before they can be used. Some plug-ins must also be configured by the user.

Some plug-ins (and the libraries they reference) load only when the process is executed by a user who is logged into the system; FileMaker Server executes as a service, not as a user process. Consequently, you need to write plug-ins differently to work with FileMaker Server. Users will need to see their operating system documentation to find which libraries are typically available.

For information on installing web publishing plug-ins, see FileMaker Server Help.

To install a plug-in, drag the plug-in file into the FileMaker user's Extensions folder as follows:

On this operating system:	Store the plug-in in this folder:
Windows XP	C:\Documents and Settings\user_name\Local Settings\Application Data\FileMaker\Extensions\
Windows Vista	C:\users\user_name\AppData\Local\FileMaker\Extensions\
Mac OS X	Macintosh HD/Users/user_name/Library/Application Support/FileMaker/Extensions

In Windows, the plug-in extension must be .fmx. In Mac OS X, the plug-in extension must be .fmplugin.

To enable a plug-in:

1. Open the Preferences dialog box.

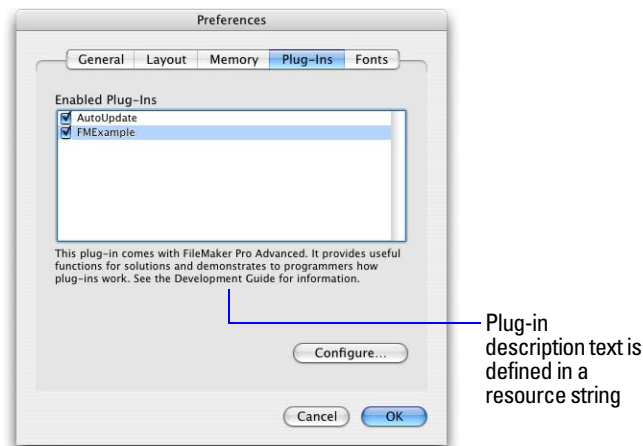
Windows: Choose Edit menu > Preferences.

Mac OS X: Choose FileMaker Pro Advanced application menu > Preferences.

2. Click the Plug-Ins tab.

3. Select the plug-in in the list.

A plug-in will appear in the list if it's installed in the correct FileMaker folder. When starting, FileMaker Pro first loads the plug-ins stored in the current user's FileMaker Extensions folder. If a particular plug-in is not found in that folder, FileMaker Pro searches for that plug-in in the Extensions folder for the FileMaker Pro application.



Select a plug-in to enable it

To configure a plug-in:

1. Select the plug-in in the Preferences dialog box.
2. Click Configure.

The Configure button is only available when the sixth character in the option string of the selected plug-in is “Y.” See “Option string syntax” on page 42.

3. Follow instructions in the configuration dialog box to configure the plug-in.
4. Click OK.

Description of the FMExample plug-in's external functions

The FMExample plug-in provided in the Microsoft Visual C++ and Apple Xcode example projects adds the following external functions to FileMaker Pro, FileMaker Pro Advanced, FileMaker Server, and FileMaker Server Advanced.

Function's name and parameter	Description of external function
XMpl_Add(number1; number2)	Adds number1 and number2 together and returns the result. The function is the same as the plus operator in the calculation engine.
XMpl_Append(textToAppend ...)	Takes a multiple list of parameters, concatenates them, and returns the result. The function is the same as the ampersand operator in the calculation engine.
XMpl_Evaluate(calcToEvaluate)	Takes a simple or complex calculation, evaluates the calculation, and returns the result. Any calculation supported by FileMaker can be passed to this function. The function is identical to the Evaluate function in the calculation engine.
XMpl_NumToWords(number)	Returns a number in bank check format. For example 44.345 returns Forty-Four Dollars and 34 Cents. All digits beyond the second decimal place and any alphabetical characters are ignored.
XMpl_StartScript(filename; scriptname)	Runs the script specified by the scriptname parameter on the file specified by the filename parameter.
XMpl_Version	Returns the version of the plug-in. It has no parameters.

Note A version function similar to the one provided in the example plug-in is required for every FileMaker Pro plug-in.

Function's name and parameter	Description of external function
XMpl_UserFormatNumber (textOrNumber)	<p>Returns the parameter as a text string formatted as specified in the configuration dialog box. Use this function to format text or numbers such as telephone numbers, postal codes, and so on.</p> <p>Formatting proceeds from right to left. Each # symbol in the format string is replaced by the next character in the parameter string. All remaining # symbols are replaced with zeros.</p> <p>This function demonstrates both client-only functionality and the plug-in configuration dialog.</p>
XMpl_FormatNumber(formatString; textOrNumber)	<p>The same as XMpl_UserFormatNumber, but formatString is provided as a parameter. This function illustrates a function that is visible in the Specify Calculation dialog for Auto Entry, and also supports calls from FileMaker Server, and Instant Web Publishing.</p>

Using the example plug-in

To access the external functions:

1. Open a file.

2. Open the Preferences dialog box.

Windows: Choose Edit menu > Preferences.

Mac OS X: Choose FileMaker Pro Advanced application menu > Preferences.

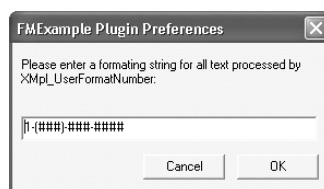
3. Click the Plug-Ins tab.

4. Select FMExample.

Because the example plug-in includes a function that requires configuration, the Configure button is enabled.

5. Click Configure.

The configuration dialog box that appears depends on how the plug-in source code was written. The XMpl_UserFormatNumber function in the FMExample plug-in displays the following configuration dialog box.



6. Click OK to use the default format or type a new format.

The “#” symbols are replaced by numbers. All other text in the format string is retained as is.

7. Click OK to close the Preferences dialog box.

8. With your file still open, in FileMaker Pro Advanced, choose Manage menu > Database > Fields tab.

9. Create a calculation field.

10. In the Specify Calculation dialog box, choose External Functions from the View drop-down list.

11. Double-click an external function to add it to the formula box.

All external function calls require the name of the external function to call and the function's parameter value, even if the value is null.

12. Replace the parameter placeholder with the required parameter or parameters for the function.**13.** Continue to build the formula then and click OK when you're done.**14.** Click OK to close the Manage Database dialog box.

Mac OS Plug-ins created for PowerPCs need to be re-compiled as universal binaries to be able to run natively when FileMaker Pro is run on Intel-based Macintosh computers. Universal binaries allow applications to run natively on both PowerPC and Intel-based Macintosh computers.

To compile the FMExample or your custom plug-ins on Mac OS X, perform a custom installation of Xcode and select the option "Cross-Development."

Customizing the plug-in example

The plug-in example in FileMaker Pro Advanced is designed to be easily modified so you can add your own custom functions. You need to modify the following items:

- version information in FMPluginExample.strings and FMPluginExample.rc
- plug-in and function names in FMPluginExample.strings and FMPluginExample.rc
- configuration function in FMPluginPrefs.cpp
- external function definitions and coding in FMPluginFunctions.cpp

Customizing the example resources

You must make the following modifications to the plug-in resource files to create a custom external function plug-in:

- Modify the version variables and strings to meet your needs.
- Revise the configuration dialog box to meet your needs.
- Specify the correct option string values.
- Edit plug-in names and description.
- Define your function names and function prototypes.

Customizing FMPluginExample.cpp

Make your modifications to the FMPluginExample.cpp in the functions listed in the following table.

Function name	Customization
Do_PluginInit	Provide your own unique plug-in ID for "pluginID." Register each function, providing its name, description, and function to be used. Call <code>fm::ExprEnv::RegisterExternalFunction</code> to register your functions.
Do_PluginIdle	Add any idle processing your plug-in needs.
Do_PluginShutdown	Revise the <code>UnRegisterExternalFunction</code> calls to reverse the registration done in <code>Do_PluginInit</code> . Call <code>fm::ExprEnv::UnRegisterExternalFunction</code> to unregister your functions.

Customizing FMPluginPrefs.cpp

This file contains the Do_PluginPrefs function for the implementation of the configuration dialog box. Revise or remove this code as needed.

Customizing FMPluginFunctions.cpp

Revise or remove the functions provided in the FMPluginFunctions.cpp file and define your own. Do_PluginInit refers to these functions when evaluating external functions in calculations.

Requirements for writing external function plug-ins

FileMaker plug-ins are most useful when they contain a single function or a set of functions with similar features. When you design your plug-in, keep in mind that developers who use your plug-in may not understand programming conventions that you take for granted. The format of each function's parameter should be understandable to the typical user.

If you are creating a FileMaker plug-in with functions that do not require any parameters, make sure the function “prototype” registered for that function does not include parentheses. For example, “DoThis” should be registered instead of “DoThis()” as the function prototype.

API code files

There are ten API code files in the Headers folder: FMXExtern.h, FMXCalcEngine.h, FMXBinaryData.h, FMXDateTime.h, FMXTextStyle.h, FMXTypes.h, FMXFixPt.h, FMXClient.h, FMXText.h, and FMXData.h. The files are not redistributable in source code (or human readable) form, cannot be modified, and are only provided to enable licensees of FileMaker Pro Advanced to compile plug-ins for use with FileMaker products. Not all the files are required to build all types of plug-ins.

The FMXExtern.h is absolutely required. The FMXExtern.h defines the parameter block (the shared data structure used by your plug-in and FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server) and some shared function calls. The function calls are used to manipulate the parameter and result handles in the parameter block.

The FMXExtern.h file defines the call-back functions for backward compatibility operations and the different kinds of plug-in events (FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server messages) sent to the plug-in in a FMExternCallSwitch definition.

FMExternCallStruct defines the structure of the parameter block. FMExternCallPtr is a pointer to that structure and gFMExternCallPtr is a global variable that should be defined in your code.

The FMXCalcEngine.h file contains the register and unregister functions. It will be used in most plug-ins, as the plug-ins will likely need to register functions.

The functionality of the remaining API code files is described in comments that are included in the files themselves.

Option string syntax

The option string must be 11 characters long for plug-ins.

The first four characters of the option string are the ID of the plug-in. The ID must be unique for each plug-in and must not begin with “F,” “FM,” or “Web.” For the Mac OS X, it is recommended that you set the creator type of the plug-in to this same value. The ID can only contain low-ASCII alphanumeric characters (such as 0-9, A-Z, and a-z).

Note So that there will be a good chance of having a unique ID, you should register the ID at the Apple Developer Support website, even if you won’t be creating a Mac OS X version of your plug-in. To register plug-in IDs as Creator codes, go to the developer pages on the Apple Inc. website at www.apple.com.

The fifth character of the option string is always “1” and the eighth, tenth, and eleventh are always “n.” Other values for these flags are reserved for FileMaker use only.

For example, “Moc31YnnYnn” is a option string for a plug-in with the ID of “Moc3” (characters 1-4) that requires configuration (character 6 = “Y”), uses the new style registration and functions callbacks (character 7 = “n”), and requires special idle time (character 9 = “Y”).

Characters in the option string	Description of characters
1-4	Characters 1-4 are the plug-in ID. Register the ID as a Creator code on the Apple Developer Support website at www.apple.com .
5	Character 5 is always “1.”
6	Set the sixth character of the option string to “Y” if you want to enable the Configure button for plug-ins in the Preferences dialog box. Use “n” if there is no plug-in configuration needed. If the flag is set to “Y,” then make sure to handle the kFMXT_DoAppPreferences message. For more information, see “FileMaker messages sent to the plug-in” on page 42.
7	Set to “n” for the new style plug-in registration and function callbacks demonstrated in the FMExample. Only set to “Y” if your plug-in requires the legacy function string list and single external callback.
8	Character is always “n.”
9	Set the ninth character of the option string to “Y” if the kFMXT_Idle message is required. For simple external functions this may not be needed and can be turned off by setting the character to “n.”
10	Character 10 is always “n.”
11	Character 11 is always “n.”

Naming conventions for external functions

The function name prefix for all of the plug-in’s external functions must be a unique value containing four or five characters and must not begin with the characters “FM” or “Web.” Four-character prefixes are reserved by FileMaker. For example, the FMPluginExample plug-in’s function name prefix is “XMpl.”

FileMaker messages sent to the plug-in

There are six possible calls that FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server can request of your plug-in. Messages sent to your plug-in are supplied in the `whichCall` field of the parameter block, `FMExternCallStruct`, defined in the `FMXExtern.h` file.

- `kFMXT_Init` — the Initialization message

- `kFMXT_Shutdown` — the Shutdown message
- `kFMXT_Idle` — the Idle message
- `kFMXT_DoAppPreferences` — the Preferences message
- `kFMXT_External` — the External Function message received by legacy plug-ins that set character 7 in the options string to “Y” and that register their functions the old external way
- `kFMXT_GetString` — the GetString message received by plug-ins that use the new style of registration when the plug-ins provide the option string, plug-in name, and description

Initialization message

The Initialization message, `kFMXT_Init`, is sent to the plug-in whenever it is enabled in FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server. This may or may not correspond with the startup of the application, depending on whether the plug-in is enabled in the Preferences dialog box.

There are two possible result values that the plug-in should return in response to the Initialization message:

- `kBadExtnVersion` should be returned if the version number passed is less than the value of `kMinExtnVersion` or greater than the value of `kMaxExtnVersion`. This prevents the plug-in from running on an API that is incompatible with the API with which it was compiled.
- `kCurrentExtnVersion` is the only other result value that should be returned. This causes the plug-in to be enabled.

For the `FMPluginExample` plug-in, the `Do_PluginInit` function is called when the Initialization message is received. The `Do_PluginInit` function first checks the version of the API that the plug-in was compiled with to verify that it’s compatible with the version of FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server that has loaded it. Then the function checks for preferences and sets them if they exist. If no preferences currently exist, it will create them with default values.

In Windows, these preferences are stored as registry entries. In Mac OS X, they are stored in a file within the Preferences folder of the System Folder. Due to the differences between the way this information is stored on the two platforms, the `Do_PluginInit` function uses preprocessor instructions to choose the correct code at compile time.

If the preferences are set properly and the API version is correct, the `Do_PluginInit` function in the `FMPluginExample` plug-in will return `kCurrentExtnVersion`.

After you set the preferences, register each external function by providing its name, description, and the function to be used. Use `fm::ExprEnv::RegisterExternalFunction` to register your functions.

Shutdown message

The Shutdown message, `kFMXT_Shutdown`, is sent to the plug-in whenever it is disabled in FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server. This may or may not correspond with the quitting of the application, depending on whether the plug-in is disabled in the Preferences dialog box.

The `FMPluginExample` plug-in does not allocate any persistent memory on the heap, and therefore does not do anything when it receives the Shutdown message. You should implement a clean-up function in your plug-in, however, to deallocate anything you have on the heap and exit from any operating system services you may be using. It’s possible for a plug-in to be enabled and disabled multiple times during a session, so it’s important for your plug-in to clean up memory.

Unregister each external function registered during the Initialization message using `fm::ExprEnv::UnRegisterExternalFunction`.

Idle message

The Idle message, `kFMXT_Idle`, is only sent to the plug-in during idle time if the idle feature flag was set to “Y” in the option string and the plug-in is currently enabled.

There are five times when this message is called by the FileMaker application.

If the `idleLevel` parameter is not zero, then the routine has been called while the application is running a script or is being controlled by the user. One of the following four messages has been sent:

Message	Meaning
<code>kFMXT_UserNotIdle = 1</code>	The user has done something within the last 30 seconds.
<code>kFMXT_ScriptPaused = 2</code>	The user is running a script that has been paused.
<code>kFMXT_ScriptRunning = 3</code>	The user is running a script.
<code>kFMXT_Unsafe = 4</code>	Same as if the <code>unsafeCalls</code> parameter is set to true.

Do not perform any lengthy, user interface, or event processing when the `idleLevel` parameter is not zero.

The Idle message will also be sent is when the application detects free time and does its own internal idle handling.

Message	Meaning
<code>kFMXT_UserIdle = 0</code>	The user hasn’t done anything within the last 30 seconds or more.

Preferences message

The Preferences message, `kFMXT_DoAppPreferences`, is sent in response to a user clicking the Configure button for the selected plug-in in the Preferences dialog box.

The plug-in should display a dialog box that will allow the user to set any specific configuration data required by the plug-in. If the plug-in requires user-definable preferences, you should implement your user interface here. The Configure button will only be enabled if the sixth character of the option string is set to “Y.” For more information, see “Option string syntax” on page 42.

Any options that need to be saved should be placed in their own registry entry (Windows) or in their own preference file (Mac OS X).

The FMExample plug-in needs to implement a configuration dialog box for the `XMpl_UserFormatNumber` function, so the flag has been set in the option string (`Xmpl1Ynnnnn`) and the function `Do_PluginPrefs` is called when the Preferences message is received.

External Function message

The External Function message, `kFMXT_External`, is a legacy message for old style plug-ins. It is no longer required for plug-ins that are registered in the new style.

GetString message

The GetString message, `kFMXT_GetString`, is sent to the plug-in when FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server want to retrieve one of the following strings from the plug-in. The plug-in developer can decide where to store the strings.

String	Meaning
<code>kFMXT_OptionsStr = 131</code>	The option string
<code>kFMXT_NameStr = 12</code>	The plug-in name
<code>kFMXT_AppConfigStr = 129</code>	The help text to display in the Preferences dialog box

Avoiding potential Mac OS X resource conflicts

Problems can occur on Mac OS X machines if your plug-in has the same ID for a resource that FileMaker Pro, FileMaker Pro Advanced, FileMaker Server, or another plug-in has for the same type of resource.

To avoid potential resource ID conflicts with your plug-in and other applications or plug-ins, follow these guidelines:

- **Use ID numbers between 23,000 and 24,999**

Use hard-coded IDs from this range for your dialog boxes, sounds, icons, and other resources to avoid conflicts with FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server resources. FileMaker does not use any of the IDs in this range for the application resources.

- **Set the current resource file to your plug-in**

To avoid conflicts with other plug-ins that use the same resource IDs, use the Mac OS X toolbox call in the Resource Manager to set the current resource file to your plug-in before getting any resource objects from the resource file.

Include the following line before any line that references or uses a resource:

```
UseResFile (pb -> resourceID);
```

When FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server loads your plug-in, the application gives the resource ID. This is located in the parameter block near the `param2` and `param3` variables in the `FMExtern.h` file. For more information, see “API code files” on page 41.

Providing documentation for your plug-in

Include an example database file with any special fields and scripts necessary to demonstrate the use of your plug-in’s external functions. In addition, make sure you provide documentation that describes each external function and its parameters.

Appendix A

Feature comparison of the runtime application with FileMaker Pro

When you double-click the FileMaker Pro application icon to start the application, the New Database dialog box opens and you can choose a database file. When you start a FileMaker Pro runtime application, the primary bound database file opens automatically.

Other key differences between the runtime application and FileMaker Pro include the following:

- All the database design features have been removed or hidden in the runtime application.
This includes Layout mode and commands on the Manage submenu.
- Custom functions and custom menus created with FileMaker Pro Advanced will work in the runtime application, although users of the runtime application cannot modify or create new custom functions or custom menus.
- Some other menu commands have been removed from the runtime application.
For example, you can't use the runtime application to create, open, or close a database. (Bound runtime database files must contain a custom button or script to close or open other files. There is no close command on a runtime database window.)
- FileMaker Pro Help is not available in the runtime application. However, you can use the custom menu feature to display customized Help text that you create.
- External function plug-ins can be enabled in the Preferences dialog box.
- Although the XML Data filter appears as an option for the Convert File script step, you can't convert XML files using this script step in a runtime application.
- FileMaker Pro File Sharing, serving a database on the web, or communicating with a Java applet requires FileMaker Pro or FileMaker Pro Advanced. You can, however, use a compatible version of FileMaker Server to serve runtime solution files.
- Apple events are supported but OLE automation is not supported in the runtime application on Windows machines.
- Runtime applications cannot be shared over a network.
- Runtime applications do not include the ability to Save/Send Records as Adobe PDF files.
- FileMaker Pro Advanced features are not available in the runtime application.
A runtime database can, however, be opened in either FileMaker Pro or FileMaker Pro Advanced. The full functionality of these applications will be enabled, except if full access privileges have been removed.
- Runtime applications don't support external SQL data sources (ESS), ODBC import, or the Execute SQL script step.

Application and document preferences

In the runtime application, some options are not available on the General tab of the Preferences dialog box.



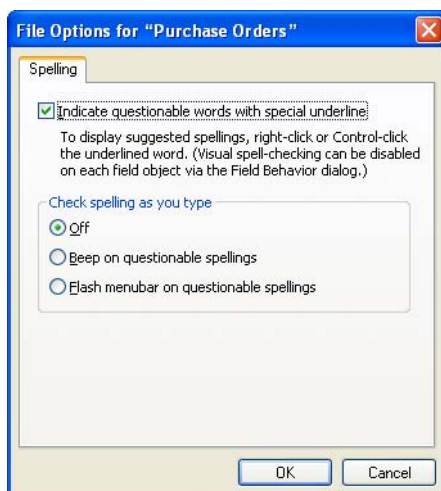
**General preferences
in a runtime
application
(Mac OS X)**

The Layout tab is changed to the Color tab in the Preferences dialog box for the runtime application.



**General
preferences in a
runtime
application
(Windows)**

The File Options dialog box in the runtime application displays only the Spelling tab.



**File options
dialog box in a
runtime
application**

Menu command comparison

The following tables show the menu commands that are available in FileMaker Pro (Pro) and in the runtime application (RT).

File Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
New Database	■		■	
Open	■		■	
Open Remote	■		■	
Open Recent	■		■	
Close	■		■	
Manage	■		■	
Sharing	■		■	
File Options	■	■	■	■
Change Password	■	■	■	■
Print Setup	■	■		
Page Setup			■	■
Print	■	■	■	■
Import Records	■	■	■	■
Export Records	■	■	■	■
Save/Send Records As	■	1	■	1
Send Mail	■	■	■	■
Send Link	■		■	
Save a Copy As	■	■	■	■
Recover	■	2	■	3
Exit	■	■		

1. You can't Save/Send Records as PDFs

2. Press Ctrl+Shift

3. Press Option+⌘

Note You can add menu items that mimic **Open**, **Close**, and **Recover** menu commands to a runtime application using custom menus based on a script or script step. For more information, see “Creating custom menus” on page 22.

Edit Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Undo/Can't Undo	■	■	■	■
Redo/Can't Redo	■	■	■	■
Cut	■	■	■	■
Copy	■	■	■	■
Paste	■	■	■	■
Paste Special	■	■		
Clear	■	■	■	■
Duplicate	■		■	
Select All	■	■	■	■
Find/Replace	■	■	■	■
Spelling	■	■	■	■
Object	■	■		
Export Field Contents	■	■	■	■
Preferences	■	■		

View Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Browse Mode	■	■	■	■
Find Mode	■	■	■	■
Layout Mode	■		■	
Preview Mode	■	■	■	■
Go to Layout	■	■	■	■
View as Form	■	■	■	■
View as List	■	■	■	■
View as Table	■	■	■	■
Status Toolbar	■	■	■	■
Customize Status Toolbar	■	■	■	■
Formatting Bar	■	■	■	■
Text Ruler	■	■	■	■
Zoom In	■	■	■	■
Zoom Out	■	■	■	■

Insert Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Picture	■	■	■	■
QuickTime	■	■	■	■
Sound	■	■	■	■
File	■	■	■	■
Object	■	■		
Current Date	■	■	■	■
Current Time	■	■	■	■
Current User Name	■	■	■	■
From Index	■	■	■	■
From Last Visited Record	■	■	■	■

Format Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Font	■	■	■	■
Size	■	■	■	■
Style	■	■	■	■
Align Text	■	■	■	■
Line Spacing	■	■	■	■
Text Color	■	■	■	■
Text	■	■	■	■

Records Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
New Record	■	■	■	■
Duplicate Record	■	■	■	■
Delete Record	■	■	■	■
Delete Found Records	■	■	■	■
Go to Record	■	■	■	■
Refresh Window	■	■	■	■
Show All Records	■	■	■	■
Show Omitted Only	■	■	■	■
Omit Record	■	■	■	■
Omit Multiple	■	■	■	■

Records Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Modify Last Find	■	■	■	■
Saved Finds	■	■	■	■
Sort Records	■	■	■	■
Unsort	■	■	■	■
Replace Field Contents	■	■	■	■
Relookup Field Contents	■	■	■	■
Revert Record	■	■	■	■

Requests Menu command (Find mode)	Windows		Mac OS X	
	Pro	RT	Pro	RT
Add New Request	■	■	■	■
Duplicate Request	■	■	■	■
Delete Request	■	■	■	■
Go to Request	■	■	■	■
Show All Records	■	■	■	■
Perform Find	■	■	■	■
Constrain Found Set	■	■	■	■
Extend Found Set	■	■	■	■
Revert Request	■	■	■	■

Scripts Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
Manage Scripts	■		■	
Save Script	■		■	
Save All Scripts	■		■	
Revert Script	■		■	
<Script names>	■	■	■	■

Note The Save Script, Save All Scripts, and Revert Script menu commands only appear when the Manage Scripts or Edit Script dialog box is active.

Window Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
New Window	■	■	■	■
Show Window	■	■	■	■
Hide Window	■	■	■	■
Minimize Window	■	■	■	■
Tile Horizontally	■	■	■	■
Tile Vertically	■	■	■	■
Cascade Windows	■	■	■	■
Arrange Icons	■	■		
Bring All To Front			■	■
<Names of open files>	■	■	■	■

Help Menu command	Windows		Mac OS X	
	Pro	RT	Pro	RT
FileMaker Pro Help	■		■	
Keyboard Shortcuts	■		■	
Resource Center	■		■	
Product Documentation	■		■	
Downloads and Updates	■		■	
Register Now	■		■	
Activate/Deactivate (Displays when the user has not activated/Displays after the user has activated)	■		■	
Send us your Feedback	■		■	
Visit our Forum	■		■	
About FileMaker Pro (or About FileMaker Pro Advanced)	■		1	
About FileMaker Pro Runtime (Displays if no custom About script is specified)		■		1
About <runtime solution> (Displays if custom About script is specified)		■		1
<Runtime solution Help script name> (Displays if custom Help script is specified)		■		■

¹ See Application Menu command table

Application Menu command (Mac OS X only)	Pro	RT
About FileMaker Pro	■	
About FileMaker Pro Runtime (Displays if no custom About script is specified)		■
About <runtime solution> (Displays if custom About script is specified)		■
Preferences	■	■
Services	■	■
Hide FileMaker Pro	■	
Hide <runtime solution>		■
Hide Others	■	■
Show All	■	■
Quit FileMaker Pro	■	
Quit <runtime solution>		■

Ignored script steps

Because some features have been removed from the runtime application, the following script steps are ignored by the runtime application:

- Open Manage Database
- Open Manage Value List
- Open Manage Data Sources
- Open Manage Scripts
- Open Sharing
- Open Help
- Set Multi-User
- New File
- Open File Options (partially available; Spell checking tab will open)
- Open Remote
- Execute SQL
- Save Records as PDF

Note Open File returns an error if the specified file has not been bound to the runtime application. A runtime solution can only perform an external script if the external file is bound to the runtime solution.

Stored registry settings or preferences

Windows registry settings

FileMaker Pro stores its registry settings at

HKEY_CURRENT_USER\Software\FileMaker\FileMaker Pro\0

FileMaker Pro Advanced stores its registry settings at

HKEY_CURRENT_USER\Software\FileMaker\FileMaker Pro\0A

The runtime application stores its registry settings at

HKEY_CURRENT_USER\Software\FileMaker\<solution name>\0

Note The filename extension for the runtime database files is registered at HKEY_CLASSES_ROOT.

Mac OS X preferences

FileMaker Pro stores its preferences in the FileMaker Pro .0 Prefs file inside the FileMaker Preferences folder.

FileMaker Pro Advanced stores its preferences in the FileMaker Pro .0A Prefs file inside the FileMaker Preferences folder. The runtime application stores its preferences in the <Solution name> Prefs file inside the FileMaker Preferences folder.

Index

A

- About layout 6
 - required contents of 7
- access keys, custom menus 23
- access privileges 7, 19
- accounts and privileges 7, 19
 - for Kiosk mode 17
 - removing Admin access 10
- Admin access
 - removing from files 10
 - removing from Kiosk solutions 18
- Apple events in runtime applications 47
- Apple Xcode 36
- ASCII characters in plug-in IDs 42
- attributes in layout themes 24
- authenticating scripts 30
- auxiliary files
 - problems with double-clicking icons 13
 - updating 17

B

- backups 16
- binding key
 - about 12
 - updating runtime database solutions 16
- binding runtime solutions 12

C

- C/C++ 35
- calculations
 - advanced 31
 - using external functions 35
- colors, layout themes 25
- commands, menu
 - available in runtime applications 49
- compression utilities for runtime databases 15
- configuring plug-ins 38, 42
- converting files from previous versions 11
- copying field or table schemas 19
- cross-platform solutions 12
- custom functions, creating 20
- custom menus
 - about 20
 - creating 22
 - example 21

- keyboard shortcuts 23
- menu items 22
- menu sets 24

D

- data sources
 - updating 16
 - updating automatically 10
- Data Viewer 31
 - Current tab 31
 - Watch tab 31
- Database Design Reports 33
- database schemas 33
 - copying or importing 19
- database statistics 33
- database structure, recreating 33
- DDR. *See* Database Design Reports
- debugging scripts 29
- delay, splash screen 12
- Developer Utilities
 - about 9
 - creating runtime solutions 10
- disabling script steps 30
- distributing runtime database solutions
 - about 13
 - distributing updates 16
 - terms and conditions 6
- Do_PluginInit function 43
- documenting
 - Database Design Reports 33
 - runtime solutions 15
- Dynamic Link Libraries (DLLs) 13

E

- Edit menu commands available in runtime applications 50
- electronic documentation 6
- elements in layout themes 27
- error codes, viewing from Script Debugger 29
- error log 10
- Execute SQL script step 54
- expressions, monitoring 31
- Extensible Markup Language (XML). *See* XML
- Extensions folder, user
 - location for plug-ins 37

extensions, filename. *See* filename extensions
 External Function message sent to plug-ins 44
 external function plug-ins 35
 enabling 37
 in runtime applications 47
 messages sent by FileMaker Pro 42
 plug-in ID 42
 external functions 35

F

fields
 copying schema 19
 monitoring 31
 File menu commands available in runtime applications 49
 File Options available in runtime application 48
 file references. *See* data sources
 FileMaker Developer. *See* FileMaker Pro Advanced
 FileMaker Pro Advanced
 documentation 5
 license agreement 6
 upgrading from earlier versions 11
 FileMaker Pro, menus available 49
 FileMaker Server 15, 35, 47
 filename extensions
 for database files
 for runtime solutions 12
 layout themes 25
 plug-ins 37
 files
 compressing runtime 15
 converting 11
 removing Admin access 10
 renaming 9
 updating 10
 FMExample plug-in 38
 fmplugin filename extension 37
 fmx filename extension 37
 folder structure
 example plug-in 36
 solution 13
 fonts
 layout themes 24
 not on user's system 14
 Format menu commands available in runtime applications 51
 formulas
 for custom functions 20
 monitoring 31

fth filename extension 25
 functions
 custom 20
 external 35
 monitoring in formulas 32

G

GetString message sent to plug-ins 45

H

Help
 menu commands available in runtime applications 53
 Help layout 15
 HTML format for Database Design Reports 33

I

icons for runtime solutions 13
 Idle message sent to plug-ins 44
 Initialization message sent to plug-ins 43
 Insert menu commands available in runtime applications 51
 installation instructions 5
 installers for runtime databases 14
 InstallShield 14
 Internet
 databases on 6
 runtime applications on 47

J

JDBC, FileMaker as data source 6

K

keyboard shortcuts in custom menu items 23
 Kiosk solutions, creating 17

L

Layout mode commands
 unavailable in runtime applications 47
 layout themes, creating 24
 legal requirements 6
 license agreement 6
 Logfile.txt 10
 logo, adding to runtime solution 12

M

- Mac OS X
 - resource conflicts 45
 - runtime application package 14
 - stored preferences 55
- MacInstallerBuilder 14
- Manage Scripts 30
- Manage submenu
 - unavailable in runtime applications 47
- menu commands
 - available in runtime applications 49
- menu separators 23
- menu sets, creating 24
- messages
 - error log 10
 - sent to external function plug-ins 42
- Microsoft Visual Studio 36
- Microsoft Windows
 - stored registry settings 55
- MindVision Installer VISE 14
- multiple tables per database file 11

N

- naming runtime database solutions 12
- networks
 - sharing solutions on 15
- new features 5
- New File script step 54
- New Layout/Report assistant 24, 26

O

- ODBC, FileMaker as data source 6
- OLE automation in runtime applications 47
- Open Define File References script step. *See* Open Manage Data Sources script step
- Open File Options script step 54
- Open File script step 54
- Open Help script step 54
- Open Manage Data Sources script step 54
- Open Manage Database script step 54
- Open Manage Scripts script step 54
- Open Manage Value List script step 54
- Open Remote script step 54
- Open Sharing script step 54
- opening files in runtime applications 47
- option string syntax for plug-ins 42

P

- passwords, required warning in About layout 7
- patterns, layout themes 25
- PDF documentation 6
- plug-ins
 - configuring 38, 42, 44
 - example project 35
 - function name prefix 42
 - IDs 42
 - in runtime applications 47
 - installing 37
 - installing web publishing plug-ins 37
 - preparing 35
 - registering with Apple 42
 - required option string syntax 42
 - resource ID conflicts (Mac OS X) 45
- preferences available in runtime application 48
- Preferences message sent to plug-ins 44
- primary file
 - connecting auxiliary files 11
 - specifying 9
 - updating 16
- Project Folder 9, 12

R

- Records menu commands available in runtime applications 51
- recovering damaged runtime files 15
- registering plug-in IDs 42
- registry, stored settings 55
- renaming files 9
- reports, database 33
- Requests menu commands available in runtime applications 52
- runtime applications
 - available menu commands 49
 - compared to FileMaker Pro 47
 - enabling plug-ins in 47
 - icon 13
 - ignored script steps 54
 - stored Mac OS X preferences 55
 - stored Windows registry settings 55
- runtime database solutions
 - About layout requirements 7, 15
 - binding files 12
 - converting 11
 - creating 9, 10
 - documenting 13
 - preparing files 10

- recovering damaged files 15
- starting 13
- updating 16
- upgrading 11
- runtime database solutions, documenting 14

S

- Save Records as PDF script step 54
- schemas, database
 - copying or importing 19
 - documenting 33
- Script Debugger 29
 - with script triggers 30
- script triggers, debugging 30
- scripts
 - authenticating 30
 - debugging 29
 - disabling script steps 30
 - steps ignored by runtime applications 54
 - unlocking 30
- Scripts menu
 - commands available in runtime applications 52
- separators, menu item 23
- Set Multi-User script step 54
- settings file 10
- shortcuts, keyboard. *See* keyboard shortcuts
- Shutdown message sent to plug-ins 43
- solution file
 - icon 13
 - problems with double-clicking icon 13
- splash screen in runtime solutions
 - closing 12
 - startup 11
- starting runtime solutions 13
- statistics, database 33
- structure, database 33

T

- tables, database
 - copying or importing schema 19
 - excluding from Database Design Report 33
 - multiple per file 11
- testing
 - database solutions 15
 - scripts 29
- text editors 24
- themes. *See* layout themes
- touch screen 17
- troubleshooting

- calculations 31
- fields 31
- Script Debugger 29
- using Database Design Reports 33
- variables 31
- tutorial, FileMaker Pro 6

U

- unlocking scripts 30
- updates
 - plug-ins 35
 - to runtime solutions 16
- upgrading runtime databases 11
- user interaction with database solution 11

V

- variables, monitoring 31
- View menu commands available in runtime applications 50

W

- web browser users 6
- web viewer 15
- Window menu commands available in runtime applications 53
- Windows
 - runtime application package 14

X

- XML
 - documents for layout themes 24
 - editors 24
 - format for Database Design Reports 33
 - output grammar for DDR 6
 - XML 1.0 specification 27
 - XML-document processing instruction 27