



>

Jazz Global Storage Plug-in

>

Updated 6 January, 2007

◇

Introduction

◆ Overview

This plugin allows storage of global values without dedicating a global field to the task. These global values are available from any FileMaker database.

◆ Shareware or Freeware

This plug-in is shareware. It will remind you of this fact periodically unless you register. Registration does not cost much, and encourages me to write more plugins. If you use this plug-in regularly, please register it (purchase a license) at <http://www.jazzmedia.com.au>.

The single license is for use by a single user. If you wish to distribute this plugin with your solution there are separate licenses available on a per-product basis. Consult the web-site for more details.

◆ Disclaimer

The plug-in is offered "as-is", and comes with no warranty of any kind. You are responsible for determining its suitability for your particular purpose or purposes, and that it performs accordingly. Please do not register the plugin until you have so-determined the plugin's suitability and effectiveness as per your requirements.

However, earlier versions of this plugin have been used in several thousand commercial CD-ROMs (Mac and Windows) for years prior to its release as shareware. It has proved to be a very valuable and reliable plugin for these products.

Conventions

◆ Definitions

The terms *function* is used to refer to an *external* function in the plugin. The terms *script* and *sub-script* will be used to refer to *Filemaker* scripts, as is consistent with *Filemaker* terminology. Occasionally the documentation may inadvertently refer to a *Filemaker* script as a *function*. If so, context should make it clear whether an *external* (plugin) function or a *sub-script* is being referenced.

◆ Calling functions

Version 1.5 (and later) of this plugin makes use of the newer *FileMaker Pro* plugin architecture. *Plugins* can take multiple values (arguments) including none, and return a single value. Although some of the functions in this plugin don't need to return any value, they will in fact return an empty string ("") and they still need to be wrapped in a script step as if they did return a value.

Plugins are designed to be used inside *FileMaker* formulas. For example, if a plugin has a function to calculate the square of a number it might be included inside a calculation like:

```
Set Field [Area; 3.1415 * Square(Radius)]
```

This calculates the radius of a circle, where 'Area' and 'Radius' are *FileMaker* fields, and 'Square' is the name of a function. The function *Square* accepts the field 'Radius' as its first and only argument (also known as its first parameter), and returns the square of this number, which is used as a formula in the 'Set Field' function.

When an external function returns a value which you will use, you must call it as shown above. However when an external function does not return a value¹ (or the value is unimportant) there are two ways you might choose to use it. The first is to create an If-End If pair of functions, with no script steps in between. The return value of the function is thus ignored. If an external were to display a dialog box of information, you could call it this way:

```
If [Display Dialog("Hello there")]  
End If
```

A more compact way (vertically) to call this function is to save the result into a dummy field. I create a global field called 'temp' which I reserve for off loading unwanted function return values. The same function call using this method would look like:

```
Set Field [temp, Display Dialog("Hello there")]
```

I prefer the second method because of its compactness, and it will be used in the examples which follow. The advantage of the first method is that no dummy field is required.

¹ Actually all external functions return an argument, irrespective of whether it is required or not. If a return value is not required the function will return the empty string ("").

General use

◆ Counting to 10

Following is a simple script to count to 10 using the *Jazz Globals* plugin and a global variable we will call “counter”. The comments before each line explain it’s purpose.

```
Enter Browse Mode []
# set a global called “counter” to 1
Set Field [temp; GlobalSet(“counter”; 1)]
Loop
  # Do something with “counter”.
  # We’ll just place it in a field called ‘value’:
  Set Field [value; GlobalGet(“counter”)]
  # exit the loop when we reach 10
  Exit Loop If [GlobalGet(“counter”) = 10]
  # add 1 to “counter”
  Set Field [test; GlobalSet(“counter”; GlobalGet(“counter”) + 1)]
End Loop
```

Of course the script doesn’t do much. You could use a *FileMaker* global field to do this also. It is just a simple example demonstrating the two most common function in *Jazz Globals*, which are *GlobalSet* and *GlobalGet*.

If you are fairly new to scripting, the last ‘Set Field’ step is a little more difficult to comprehend. First we get the value with *GlobalGet* and add one to it. This value is then used inside a *GlobalSet* call, to put the new value back into the global called “counter”.

Function Summary

◆ GlobalVersion

This function serves two purposes. With no argument the function simply returns a string describing the plugin, its registration status (if applicable) and the version number of the plugin. For example:

```
GlobalVersion
```

will return a string similar to "Jazz Globals Version 1.50". The ‘TextToNum’ function can be used to return the version number on its own (as a decimal number), from which comparisons can be made. (For this reason the version number will contain only a single decimal point, eg “1.21” and not “1.2.1”, as has become common practise today). The following example places the numeric version in the field *plugin version*.

```
Set Field [“plugin version”, TextToNum(GlobalVersion)]
```

The second purpose of this function is for registration. The serial number provided to registered users should be given to this function when a database which uses this plugin is opened. The returned string indicates whether registration was successful or not. For example, if your serial number is "12345":

```
Set Field [temp ; GlobalVersion(12345)]
```

sets field 'temp' to either "Jazz Globals Version 1.50" or "Jazz Globals Version 1.50 - Unregistered".

Even though the registration code need only be run once to register use for all open databases, it is recommended that a script containing the registration code is placed in the start-up script for every database which uses this plugin. (You assign start-up scripts in *FileMaker's* document preferences). You can check your registration status at any time by calling this function with no arguments, as shown earlier.

If you provide an invalid serial number, you will need to quit and re-launch FileMaker before the correct serial number will be accepted.

◆ GlobalSet

Sets a named global variable to the value provided. The format of the function, followed by two example of use are:

```
Set Field [temp ; GlobalSet("<label>"; <value>)]  
Set Field [temp ; GlobalSet("PI"; 3.14)]  
Set Field [temp ; GlobalSet("my name"; user_name)]
```

In the first example (second line), the number '3.14' is stored under the name (or label) 'PI'. The second example is perhaps a more common usage form, where the contents of the FileMaker field called 'user_name' is saved under the label 'my name'. The function returns the value stored if successful, otherwise it returns the empty string (eg. if not enough memory is available to save the value).

Except in the use of 'Local' or 'Frame' functions (see below), calling this function with a name (label) which already exists will cause the previous value to be overwritten.

◆ GlobalGet

Retrieves the named value. The format of the function, followed by an example are:

```
Set Field [<field>, GlobalGet(<label>)]  
Set Field ["user name", GlobalGet("my name")]
```

The example places the value previously saved under the name (or label) 'my name' into the FileMaker field called 'user name'. If more than one value has been saved under the same name (using the 'Local' or 'Frame' calls) then the most recently saved value will be returned.

◆ GlobalClear

Usage, to clear a named variable, or the entire stack (respectively):

```
Set Field [temp ; GlobalClear("<label>")]
Set Field [temp ; GlobalClear]
```

This function clears the named item (label), or clears the entire stack if no argument is provided (or if an empty string is provided). If there is more than one item by the same name (label), then the most recent value is deleted, allowing any previous values of the same name to become visible.

In the following examples, the first line clears the global variable “fred”, while the second line clears all global variables:

```
Set Field [temp ; GlobalClear("fred")]
Set Field [temp ; GlobalClear]
```

◆ GlobalUse

Usage:

```
Set Field [temp ; GlobalUse("<label>")]
```

Saves a label name for use with the *GlobalSetInUse* function. This is an alternative method to the *GlobalSet* function for saving values.

◆ GlobalSetInUse

Usage:

```
Set Field [temp ; GlobalSetInUse("<value>")]
```

Example use:

```
Set Field [temp ; GlobalUse("password")]
Set Field [temp ; GlobalSetInUse("rotund")]
```

This function is used in combination with *GlobalUse* as an alternative to the *GlobalSet* function. Here, the first line of the example names the variable to use, and the second line provides the value. Multiple calls to *GlobalSetInUse* may be made after a single call to *GlobalUse* is made. Of course subsequent calls will overwrite earlier values set.

This method of setting a global variable has no functional advantage over *GlobalSet*. The decision to use one or the other is a decision of convenience.

◆ GlobalLocal

Usage and example:

```
Set Field [temp ; GlobalLocal("<label>" ; <value>)]
Set Field [temp ; GlobalLocal("password"; "penguin45")]
```

This function creates a new value by a name (or label) provided, even if one already exists with this name. The new value conceals any others of the same name until it is cleared or "unframed". Usage is otherwise the same as for *GlobalSet*.

Practical usage is to create temporary 'local' variables within a FileMaker script. To create local variables, you must first ensure that *GlobalFrame* is called on entry to a function and *GlobalUnframe* is called on exit. In between these calls, any call to *GlobalLocal* creates (sets) a local variable by the name provided. The variable can be retrieved using *GlobalGet*, just as for global variables. If more than one variable exists with the same name, the more recently created variable will be accessed.

Note that strictly these variables created are only local to the containing script if surrounding calls to *GlobalFrame* and *GlobalUnframe* have been made. If you neglect to unframe, for example, the variables will not go out of scope, until *GlobalUnframe* is finally called from somewhere else. See *GlobalFrame* and *GlobalUnframe* for a description of this mechanism.

The Jazz Media plugin *Jazz Params* provides a more elegant solution for using local variables, an for the passing and returning of named parameters between *FileMaker* scripts.

◆ GlobalFrame

Usage:

```
Set Field [temp ; GlobalFrame]
```

This function places what is known as a *frame* on the stack (the stack is where the labels and values are stored). A frame is a nothing more than a marker. Any new values saved (meaning any values with a label not already on the stack, or any values saved using the *GlobalLocal* function) are placed after this marker. All these variables can be removed, along with the marker in a single call to *GlobalUnframe*.

Calls to *GlobalFrame* and *GlobalUnframe* are normally paired, and often you will use *GlobalLocal* to save temporary values in-between these calls. Multiple *frames* (markers) may be placed on top of each other (nested within). Each call to *GlobalUnframe* will remove down to and including only the most recent frame.

The following code (with notes) should explain some of the intricacies of this system.

```
Set Field [temp ; GlobalSet("password"; "rotund")]
Set Field [temp ; GlobalSet("person"; "fred")]
Set Field [temp ; GlobalFrame
Set Field [temp ; GlobalSet("password"; "abracadabra")] *1
Set Field [temp ; GlobalLocal("person"; "charlie")] *2
Set Field [temp ; GlobalLocal("PI", 3.141592)]
Set Field [temp ; GlobalGet("person")] *3
Set Field [temp ; GlobalGet("password")] *4
Set Field [temp ; GlobalUnframe] *5
Set Field [temp ; GlobalGet("person")] *6
Set Field [temp ; GlobalGet("password")] *6
Set Field [temp ; GlobalGet("PI")] *6
```

- *1 password 'abracadabra' overwrites the previous value ('rotund')
- *2 'charlie' is saved with the label 'person', obscuring but not overwriting the previous value, because of the use of *GlobalLocal*.
- *3 places the text "charlie" into the FileMaker field 'temp'
- *4 places the text "abracadabra" into the FileMaker field 'temp'
- *5 deletes all names created since the frame, which is 'PI' and the second instance of 'person'
- *6 places into the FileMaker field 'temp' (in succession):

"fred"	old value of 'person',
"abracadabra"	recall that it overwrote the first password
""	PI was destroyed because it was created after <i>GlobalFrame</i> .

◆ GlobalUnframe

Usage:

```
Set Field [temp ; GlobalUnframe]
```

This function is called to delete all *new* variables created since the most recent call to *GlobalFrame*. Each call to *GlobalUnframe* should pair an earlier call to *GlobalFrame*. See *GlobalFrame* for further description and sample of use.

▶ How it works

This plugin is implemented by creating a linked list of named elements, held in application memory. Each element can contain a value (a text string of any length) and a name (a string of up to 31 characters). Although there is an inherent order of the elements, the order is inconsequential for normal use.

More to be written.

Appendices

◆ Version History

- 1.06 Released to the public as shareware (June 2002)
- 1.07 Fixed bug where no result was returned when the shareware dialog was displayed. Decreased dialog frequency.
Corrected version string version number in *Global Version* function.
- 1.08 Fixed bug where setting a parameter or variable to the empty string failed to replace a previous value.
- 1.09 Fixed a bug where setting a global to the empty string had no effect. It now clears the value.
- 1.50 Reworked plugin for Intel Macs, using newer plugin architecture. Added unicode compatibility. Mac release (Jan 2007).
- 1.51 Fixed a bug where the shareware reminder caused an incorrect result to be returned. (Feb 2007)

◆ About Jazz Media

Jazz Media is an Australian-based company, who's main business is in creating MultiMedia products for clients. It specialises in CD-ROMs and dynamic web sites requiring a database back-end. Use of an in-house database engine for CD-ROMs provides extremely fast access to data, and means that clients can distribute solutions royalty free.

In the course of normal business plugins have been developed, either to solve specific developmental problems, or to add functionality to programs and bundled as part of the commercial products. After years of successful operation, these plugins have been repackaged and released to the public as shareware or freeware.