



>

Jazz Global Storage Plug-in

>

Updated 10 September, 2005

◇

Introduction

◆ Overview

This plugin provides a mechanism whereby values can be passed between different FileMaker databases (related files).

◆ 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 at <http://www.jazzmedia.com.au>.

The simple 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, this plugin has 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*, *command* and *routine* are used interchangeably in this document, and refer to an *external* function in the plugin. While technically there is a difference between a *function* and a *command*, *FileMaker* makes no such distinction, so the terms can be interchanged without confusion.

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*, *command* or *routine*. If so, context should make it clear whether an *external* (plugin) function or a *sub-script* is being referenced.

◆ Calling functions

This extension makes use of the FileMaker plugin architecture. Plugin functions are designed to take exactly one argument (a single string of text) and return a single argument¹. As some of the functions of this plugin require multiple or no arguments, and some of them don't return any value, some encoding of the information needs to take place to conform to the FileMaker plugin requirements.

Plugins are designed to be used inside formulas. A simple function to calculate the square of a number could be included inside a calculation like:

```
Set Field ["Area", "3.1415 * External("Square", Radius)"]
```

This script step calculates the radius of a circle, where *Area* and *Radius* are *FileMaker* fields, and *Square* is the name of the function. In this example, the function accepts the field *Radius* as its argument, and returns the square of this number, which is used in the formula as shown.

When an external function does not return a value² (or the value is unimportant) there are two easy ways to use it. The first is to create an *If-End If* pair of commands, with no script steps in-between. Because the body of the *If-End If* statement is empty, the return value of the function is ignored. A *display dialog* function could be called this way:

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

An more compact way to achieve the same result is to create a global field called *temp* (for example) which is reserved for offloading unwanted function results. The same *display dialog* function would now be called like this:

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

This is the method I prefer, because of its compactness, and will be used in the examples throughout this document.

General use

◆ example use

To be written.

¹ Actually plugins accept two arguments (strings), where the first one is the name of the function, and the second one is the data. It is more convenient to consider the first string simply as the name of the function, and thus the second string is the only data passed to this command. When we talk about a single argument being passed to a function, we are referring to this second string.

² Actually all external functions return an argument, irrespective of whether it is required or not. If a return value is not required, by convention the function will return an empty string ("").

Function Summary

◆ Global Storage Version

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

```
Set Field ["temp", "External("Global Storage Version", "")"]
```

will return a string similar to "Jazz Media Global Storage Routines 1.00". 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(External("Global Storage Version", ""))"]
```

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", "External("Global Storage Version", 12345)"]
```

sets 'temp' to either "Jazz Media Global Storage Routines 1.00" or "Jazz Media Global Storage Routines 1.00 - 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 (enabled in *FileMaker's* document preferences).

If you accidentally perform a script with a (non-empty) invalid serial number, you will need to quit and re-launch FileMaker before the correct serial number will be accepted. You can check your registration status at any time by providing an empty string to this function, as shown earlier.

◆ Global Set

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

```
Set Field ["temp", "External("Global Set", "<label>=<value>")"]  
Set Field ["temp", "External("Global Set", "PI=3.14")"]  
Set Field ["temp", "External("Global Set", "my name=" & user name)"]
```

In the first example, the value "3.14" is stored under a label named '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).

Spaces after the label but before the equals sign are ignored. Spaces after the equals sign will be saved with the string (value).

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

◆ Global Get

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

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

The example places the value previously saved under the label of '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.

◆ Global Clear

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

```
Set Field ["temp", "External("Global Clear", "<label>")"]  
Set Field ["temp", "External("Global Clear", "")"]
```

This function clears the named item (label), or clears the entire stack if no argument is provided (an empty string). 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", "External("Global Clear", "fred")"]  
Set Field ["temp", "External("Global Clear", "")"]
```

◆ Global Use

Usage:

```
Set Field ["temp", "External("Global Use", "<label>")"]
```

Saves a label name for use with the *Global Set In-Use* function. This is an alternative method to the *Global Set* function for saving values.

◆ Global Set In-Use

Usage:

```
Set Field ["temp", "External("Global Set In-Use", "<value>")"]
```

Example use:

```
Set Field ["temp", "External("Global Use", "password")"]  
Set Field ["temp", "External("Global Set In-Use", "rotund")"]
```

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

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

◆ Global Local

Usage and example:

```
Set Field ["temp", "External("Global Local", "<label>=<value>")"]  
Set Field ["temp", "External("Global Local", "password=rotund")"]
```

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 *Global Set*.

Practical usage is to create temporary 'local' variables within a FileMaker script. To create local variables, you must first ensure that *Global Frame* is called on entry to a function and *Global Unframe* is called on exit. In between these calls, any call to *Global Local* creates (sets) a local variable by the name provided. The variable can be retrieved using *Global Get*, 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 not local to the containing script, they only appear as such due to the enclosing calls to *Global Frame* and *Global Unframe*. If you neglect to unframe, for example, the variables will remain accessible from all scripts thereafter, until *Global Unframe* is finally called, possibly within another script. See *Global Frame* and *Global Unframe* for a description of this mechanism.

The Jazz Media plugin 'Jazz Params' provides local variables using a more elegant system, which also allows the passing and returning of named parameters between *FileMaker* scripts.

◆ Global Frame

Usage:

```
Set Field ["temp", "External("Global Frame", "")"]
```

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 *Global Local* command) are placed after this marker. All these variables can be removed, along with the marker in a single call to *Global Unframe*.

Calls to *Global Frame* and *Global Unframe* are normally paired, and often you will use *Global Local* to save temporary values in-between these calls. Multiple *frames* (markers) may be placed on top of each other (nested within). Each call to *Global Unframe* 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", "External("Global Set", "password=rotund)"]
Set Field ["temp", "External("Global Set", "person=fred)"]
Set Field ["temp", "External("Global Frame", "")"]
Set Field ["temp", "External("Global Set", "password=abracadabra)"] *1
Set Field ["temp", "External("Global Local", "person=charlie)"] *2
Set Field ["temp", "External("Global Local", "PI=3.141592)"]
Set Field ["temp", "External("Global Get", "person)"] *3
Set Field ["temp", "External("Global Get", "password)"] *4
Set Field ["temp", "External("Global Unframe", "")"] *5
Set Field ["temp", "External("Global Get", "person)"] *6
Set Field ["temp", "External("Global Get", "password)"] *6
Set Field ["temp", "External("Global Get", "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 *Global Local*.
- *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 *Global Frame*.

◆ Global Unframe

Usage:

```
Set Field ["temp", "External("Global Unframe", "")"]
```

This function is called to delete all *new* variables created since the most recent call to *Global Frame*. Each call to *Global Unframe* should pair an earlier call to *Global Frame*. See *Global Frame* 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 Updated internal codebase.

◆ 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 re-packaged and released to the public as shareware or freeware.