



>

## Jazz Utils Plugin

>

Updated 30 November, 2003

◇

# Introduction

## ◆ Overview

This plugin is a collection of miscellaneous utilities which I written on an ad-hoc basis, based on personal or project requirements. They have been written with specific requirements in mind, and are not guaranteed to work under all circumstances.

As I choose to do all development work on a Macintosh computer (PPC), most of the functions in this plugin are Macintosh only functions. In cases where a routine is cross-platform, it is likely only the Macintosh version has had significant testing and verified operational to my personal satisfaction. Similarly, while the Macintosh plugin is FAT (68k and PPC code), it is likely that only the PPC code has received significant testing.

Windows-capable functions include:

Jazz Media-Version	Jazz-Char2Num
Jazz-Strip HTML	Jazz-Num2Char
Jazz-Mac2Win Text	Jazz-Win Activate ( <i>Windows only</i> )
Jazz-Win2Mac Text	

## ◆ Conventions

The word 'function' and 'command' are used interchangeably in this document. While technically there is a difference between the two, FileMaker only provides a single method of access so differentiation between the terms is is not important.

## ◆ Shareware or Freeware

This plugin is freeware. You may copy the plugin and give it away at no cost. You may not sell this plugin, but you may include it free-of-charge in products you develop at your own risk. If this plugin is used in any products you develop, a credit would be appreciated, but is not required.

## ◆ Disclaimer

These plugin are provided 'as is', and no warranty or guarantee is made of any kind that it operation as expected, or documented. Use of the plugins is entirely at your own risk. You are entirely responsible for determining the suitability of the plugin(s) for your purposes.

## Calling functions

This extension makes use of the FileMaker plugin architecture. Plugins are designed to take exactly one argument (a single string of text) and return a single argument<sup>1</sup>. As some of the functions of this plugin require multiple or no arguments, and some of them don't return any value, some trickery needs to take place to facilitate this requirement.

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

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

could calculate the radius of a circle, where 'Area' and 'Radius' are FileMaker fields, and 'Square' is the name of the function. This function accepts the field 'Radius' as its argument, and returns the square of this number, which is used in the formula of the 'Set Field' command.

When an external function does not return a value<sup>2</sup> (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. 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 ["External("Display Dialog", "Hello there)"]  
End If
```

An more compact way to call this function is to create a global variable called 'temp' (or whatever you choose) which I reserve for off loading unwanted function return values. The same function call using this method would look like:

```
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 to follow.

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

<sup>2</sup> 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 ("").

## Function Summary

### ◆ Jazz Media-Version

Returns a string describing the plugin, its registration status (if applicable) and the version number of the plugin. The 'TextToNum' function is guaranteed to return the version number on its own (as a decimal number), from which comparisons can be made.

For example, in version 1.04:

```
Set Field ["temp", "External("Jazz Media-Version", "")"]
```

sets 'temp' to either "Jazz Media Utilities 1.04".

### ◆ Jazz-Get Clipboard

Usage:

```
External("Jazz-Get Clipboard", <optional replacement value>)
```

Example:

```
Set Field ["clip1", "External("Jazz-Get Clipboard", "")"]  
Set Field ["clip1", "External("Jazz-Get Clipboard", clip1)"]
```

The first example places the plain-text held in the clipboard into the field "clip1". This is the most common form. If the clipboard does not contain text then the empty string will be returned.

If an argument is provided, as in the second example, then this value will be placed on the clipboard after the current value is retrieved. This example swaps the contents of the clipboard with the contents of the FileMaker field 'clip1'.

Be aware of unexpected results if using the clipboard to exchange information with other programs, if this command is called while FileMaker is not the front-most application. To understand clipboard behaviour in these circumstances you should be aware that applications maintain their own clipboard, which is only transferred to and from the system clipboard when applications are moved to and from the foreground.

### ◆ Jazz-Set Clipboard

Usage:

```
External("Jazz-Set Clipboard", <new value>)
```

Example:

```
Set Field ["temp", "External("Jazz-", "123")"]
```

This example sets the clipboard with the plain text string "123". Any existing clipboard contents will be lost.

Be aware of unexpected results if using the clipboard to exchange information with other programs, if this command is called while FileMaker is not the front-most application. To understand clipboard behaviour in these circumstances you should be aware that applications maintain their own clipboard, which is only transferred to and from the system clipboard when applications are moved to and from the foreground.

## ◆ Jazz-Strip HTML



Usage:

```
External("Jazz-Strip HTML", <tagged text>)
```

Example:

```
Set Field ["plain text", "External("Jazz-Strip HTML", html text)"]
```

This function removes *many* HTML tags surrounding text, and applies minor reformatting to make it readable as plain text. It is by no means comprehensive in operation. I wrote this function when I needed a field to contain HTML text, but wanted the text to be legible at the same time.

An example of the tags removed and processed are:

<P>,  	converted to two paragraph characters (two returns)
<B>, <I>, <U>...	just removed
<Hn>	two paragraph returns appended after the heading
<PRE>	removed, but enclosed contents is not modified
<LI>	removed, and a bullet character and space prepended.

## ◆ Jazz-Clip to HTML



Usage:

```
External("Jazz-Strip HTML", "")
```

Example:

```
Go to Field [Select/perform, "styled text"]  
Copy [Select]  
Set Field ["html text", "External("Jazz-ClipToHTML", "")"]
```

This function creates simple character-formatted HTML text from styled text held on the clipboard. The original problem was to be able to retain simple character-style information, such as bold and italic, when exporting as tab-delimited text.

This is not possible.

The work around is to copy styled text to the clipboard and have a plugin convert this information into the equivalent tagged text. By placing the function result into another field, and looping through all the records, the desired result can be achieved.

Only bold, italic and underline formatting is tagged. There are plans to extend this to font name, size and colour at a future date, with a small level of conversion customisation (such as which font sizes are mapped to the HTML sizes, and similarly for font names).

## ◆ Jazz-Debug BBEdit



Usage:

```
External("Jazz-Debug BBEdit", <Debug print text>)
```

Example:

```
Set Field ["temp", "External("Jazz-Debug BBEdit", "Entering counting script)"]  
Set Field ["temp", "External("Jazz-Debug BBEdit", "Value of i = " & i)"]
```

This is a Macintosh-only routine. The routine is designed to aid debugging, by allowing a string to be printed to a window. This string can contain the contents of FileMaker fields, or any other contents sent to the plugin function.

To avoid reinvention of the wheel, use is made of the shareware text editor *BBEdit*. The free version called *BBEdit Lite* can be downloaded from *BareBones* web site at <http://www.barebones.com>. The function *Jazz-Debug BBEdit* sends the argument contents (plus a terminating return) to *BBEdit's* front-most window, using *Apple events*. The application must be open with a document window available for the debug string. Requires *AppleEvents*.

## ◆ Jazz-Debug STE



Usage:

```
External("Jazz-Debug STE", <Debug print text>)
```

Example:

```
Set Field ["temp", "External("Jazz-Debug STE", "Entering counting script)"]  
Set Field ["temp", "External("Jazz-Debug STE", "Value of i = " & i)"]
```

This is a Macintosh-only routine, identical to *Jazz-Debug BBEdit*, except that the results are sent to the application *Simple Text Editor*, provided on (almost) every Macintosh from *Mac OS7* thru *OS9*. The function sends the argument contents (plus a terminating return) to *Simple Text Editor's* front-most window, using *Apple events*. The application must be open with a document window available for the

debug string. Requires *AppleEvents*.

## ◆ Jazz-Debug STE Slow



Usage:

```
External("Jazz-Debug STE Slow", <Debug print text>)
```

Example:

```
Set Field ["temp", "External("Jazz-Debug STE Slow", "Entering counting script)"]  
Set Field ["temp", "External("Jazz-Debug STE Slow", "Value of i = " & i)"]
```

This routine requires *AppleScript*. It is identical to 'Jazz-Debug STE', except that it performs the operation using *AppleScript*. It was the predecessor to 'Jazz-Debug STE', and was found to be painfully slow, so the function was rewritten to use *Apple events* directly, thus also not requiring *AppleScript* to be loaded.

While superceded, this routine is retained deliberately because slowing down an application can sometimes be beneficial during the debugging process.

## ◆ Jazz-Get Mouse



Usage:

```
External("Jazz-Get Mouse", "")
```

Example:

```
Set Field ["temp", "External("Jazz-Get Mouse", "")"]  
Set Field ["x coord", "External("Jazz-Get Mouse X", "")"]  
Set Field ["y coord", "External("Jazz-Get Mouse Y", "")"]
```

This routine is designed to be used in combination with other routines in this plugin, as demonstrated in the example, above. This routine records the current mouse location (with respect to the top-left corner of the FileMaker layout) and returns immediately. The routine always returns an empty string, but the mouse location is remembered for later retrieval using the routines "Jazz-Get Mouse X" and "Jazz-Get Mouse Y".

## ◆ Jazz-Get Mouse Click



Usage:

```
External("Jazz-Get Mouse Click", "")
```

Example:

```
Set Field ["temp", "External("Jazz-Get Mouse Click", "")"]
```

```
Set Field ["x coord", "External("Jazz-Get Mouse X", "")"]
Set Field ["y coord", "External("Jazz-Get Mouse Y", "")"]
```

This routine is designed to be used in combination with other routines in this plugin, as demonstrated in the example, above. This routine waits for the next click of the mouse, and records the click location (with respect to the top-left corner of the FileMaker layout). This mouse click will *not* get passed on to FileMaker or the operating system.

The routine will not return until the mouse is clicked, and the computer might seem non-operational until in the interim. The routine always returns an empty string, but the mouse click location is remembered for later retrieval using the routines "Jazz-Get Mouse X" and "Jazz-Get Mouse Y".

This routine was written to obtain (x,y) coordinates of towns on maps (pictures) which were contained in a FileMaker database. Use of this routine allowed the coordinates to be automatically entered into the database. (One database held the towns and recorded locations, while a related one held the maps). You could possibly think of many other uses for this function.

While the behaviour with scrolled windows is beyond the control of the plugin, it turns out that a scrolled window returns a location with respect to the top-left of the form, and not the top-left of the window — which I consider to be a more useful value. Clicks outside the FileMaker window are allowed, and corresponding values (which in some cases may be negative) will be returned.

## ◆ Jazz-Get Mouse X



Usage:

```
External("Jazz-Get Mouse X", "")
```

Example:

```
Set Field ["mouse x", "External("Jazz-Get Mouse X", "")"]
```

This routine is used to return the x-location recorded at the most recent call of 'Jazz-Get Mouse' or 'Jazz-Get Mouse Click'. See these routines for further explanation.

## ◆ Jazz-Get Mouse Y



Usage:

```
External("Jazz-Get Mouse Y", "")
```

Example:

```
Set Field ["mouse y", "External("Jazz-Get Mouse Y", "")"]
```

This routine is used to return the y-location recorded at the most recent call of 'Jazz-Get Mouse' or 'Jazz-Get Mouse Click'. See these routines for further explanation.

### ◆ **Jazz-Mac2Win Text**



Usage:

```
External("Jazz-Mac2Win", <mac text>)
```

Example:

```
Set Field ["win text", "External("Jazz-Mac2Win", mac text)"]
```

This routine converts Macintosh text into the Windows equivalent ascii codes. This is only relevant for *extended* ascii codes, such as é and ö. Be aware that FileMaker does its own conversions when displaying and importing/exporting data across platforms. Thus need for this routine is rarer than one might expect.

If I recall correctly, text sent or received from a plugin is not automatically translated appropriately, which is one area where this routine becomes valuable.

### ◆ **Jazz-Win2Mac Text**



Usage:

```
External("Jazz-Win2Mac", <win text>)
```

Example:

```
Set Field ["mac text", "External("Jazz-Win2Mac", win text)"]
```

This routine converts Windows text into the Macintosh equivalent ascii codes. This is only relevant for *extended* ascii codes, such as é and ö. Be aware that FileMaker does its own conversions when displaying and importing/exporting data across platforms. Thus need for this routine is rarer than one might expect.

If I recall correctly, text sent or received from a plugin is not automatically translated appropriately, which is one area where this routine becomes valuable.

### ◆ **Jazz-Cn2Mac Text** and **Jazz-CnX2Mac Text**



These routines were written for a particular project, to convert text between a custom font (with non-standard characters in non-standard positions) and the closest equivalent Macintosh text. The font was created especially for the project, so this routine will have no value outside this project.

These routines may be removed from the plugin in future versions.

## ◆ Jazz-Win Activate

Usage:

```
External("Jazz-Win Activate", <application name>)
```

Example:

```
Set Field ["temp", "External("Jazz-Win Activate", "readme.txt - NotePad)"]  
Set Field ["temp", "External("Jazz-Win Activate", "Windows NT Task Manager)"]  
Set Field ["temp", "External("Jazz-Win Activate", "Wine)"]
```

This is a Windows-only routine. It was written when a Windows database was required to activate (bring to the front) another application. (The Macintosh version used *AppleScript*, thus it was not required for the Macintosh).

Not being a *Window* guru, I find myself unable to describe the exact parameter requirements for this function, except that it is one of the names for the application you wish to activate. Further more, an incorrect parameter can cause the FileMaker application to be terminated prematurely (crash). However use of the correct application name (once discovered) proved stable and reliable.

The three examples shown above were found to be reliable to activate the running applications. The first is the file *readme.txt* when open within the *Note Pad* application. The second is the *Windows NT Task Manager*, and the last is an application built with *Macromedia Director 7*, called *Wine.exe*. You will need to experiment to find the appropriate name for the application you wish to activate.

## ◆ Jazz-CharToNum

Usage:

```
External("Jazz-CharToNum", <single character>)
```

Example:

```
Set Field ["temp", "External("Jazz-CharToNum", "C)"]
```

This routine converts a character to its ascii equivalent. While so trivial as to be used in sample code everywhere, I find this routine invaluable. This routine is available for Windows, however *extended* ascii will presumably return the corresponding value from the Macintosh character table (since I believe these are the codes passed to the plugin). The routine *Jazz-Mac2Win* can be used to correct this behaviour if this proves to be a problem.

The example above will return the ascii value for the upper-case letter 'c', which is 67. If more than one character is passed to the routine, the ascii value of the first character will be used.

## ◆ Jazz-NumToChar



Usage:

```
External("Jazz-NumToChar", <ascii value>)
```

Example:

```
Set Field ["temp", "External("Jazz-NumToChar", 67)"]
```

This routine is the opposite of 'CharToNum', above. It turns an ascii value to the equivalent character. If the value is  $\geq 256$  then the character will be based on the value mod 256. If the ascii value is  $< 0$ , the result is undefined. The example above returns the upper-case character 'C'.

## ◆ Jazz-Common Words



Usage:

```
External("Jazz-Common Words", <lines to compare>)
```

Example:

```
Set Field ["temp", "External("Jazz-Common Words", field 1 & "¶" & field 2)"]
```

This routine compares the words on the first and second lines of the . It returns a count of the number of words appearing on both lines. The routine is case insensitive, so "abc" and "aBc" are considered the same word. For example, if the two lines (the whole string provided) contain the following:

the quick brown

Quick brown themes

then the routine will return the number '2' — two words match. (If only one line is provided, the routine will return an empty string).

---

## Appendices

### ◆ Version History

I've been a little lazy with versioning so far. This is version 1.04. There was only one version released previously, so there shouldn't be cause for confusion. The previous version contained only the first five functions listed in this documentation.

### ◆ About Jazz Media

*Jazz Media* is a small company, with close ties to an earlier company, *Primal Screen Multimedia*. Most of the combined work to date has been creating multimedia CD-ROMs for clients, all of which involve access to large quantities of data (>50,000 records in one title). A slow migration to web-based products is in-hand.

The *James Halliday Interactive Wine Companion* by *HarperCollins Publishers* (Australia) has been one of these titles, which includes a substantial FileMaker runtime database. The routines of this and other plugins have come about as direct or indirect requirements of this and other titles.

Presently we have developed approximately 15 commercial titles, and near 20 plugins for *FileMaker* and *Macromedia Director*, plus a few stand-alone utilities. In time more of these may be refined and released to the public as shareware or freeware.