RegEx PlugIn

RegexPlugIn is a Plug-In for FileMaker which enables FileMaker users to work with regular expressions.

It works with FileMaker 8 or higher on Microsoft Windows or Mac OS X on Intel (requires 10.4 or higher), both 32-bit and 64-bit.

+ D o w n l o a d + + + + + + +
+
RegexPlugIn is available as source and binary ZIP archives from GitHub. The current version is **0.3.0**.

+ L i c e n s e + + + + + + +
+

+ S u p p o r t + + + + + + +
+
Support for **RegexPlugIn** is available via **GitHub issues**. If you have any questions or if you want to report bugs, *please* use this mailing list and don’t email us directly. Thanks.

+ I n s t a l l a t i o n + + + + + + +
+
Unpack the ZIP archive. The Windows plugin is *`RegexPlugIn.fmx`* (32-bit), the Mac plugin is *`RegexPlugIn.fmplugin`* (64-bit). Put the appropriate file into the *`Extensions`* folder of FileMaker. Restart FileMaker.

If the plug-in isn’t recognized by FileMaker on Windows (i.e. if it doesn’t show up in FileMaker’s *Preferences* dialog), chances are good that you are missing Microsoft’s C++ runtime library *`msvcr80.dll`*. Most people will have it already as it is automatically installed by many other applications, but if you don’t, you can get it from here.

+ E x a m p l e s + + + + + + +
+
An example FileMaker database which explains the usage of **RegexPlugIn** interactively is available here (.fmp12) and here (.fp7).

+ R e g u l a r E x p r e s s i o n S y n t a x + + + + + + +
+

The regular expressions used in **RegexPlugIn** are based on the CL-PPCRE library and are thus compatible with Perl. For more information see https://perldoc.perl.org/perlre and http://edicl.github.io/cl-ppcre/.

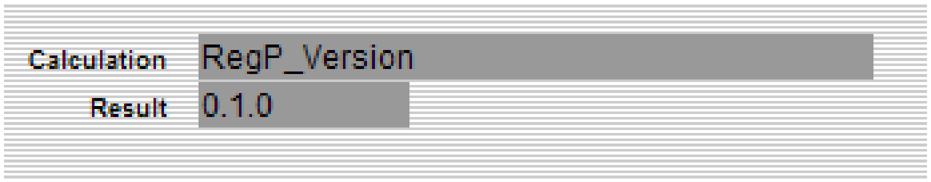
+ C o n f i g u r a t i o n + + + + + + +
+
On Windows you can configure (via FileMaker's *Preferences* dialog) whether **RegexPlugIn** should cache regular expressions (this is the default setting) or not. It is almost always advisable to do that unless you have a large database where the regular expression is variable, i.e. where it is different for each record.

+ T e c h n i c a l   i n f o r m a t i o n + + + + + + +
+
The **RegexPlugIn** was written in Common Lisp using the LispWorks development environment and the FM-PLUGIN-TOOLS toolkit. The regex engine used is CL-PPCRE.

The source code for **RegexPlugIn** is now part of FM-PLUGIN-TOOLS.

+ F u n c t i o n s + + + + + + +
+
RegexPlugIn provides the following *external* functions which can be used in FileMaker calculations:

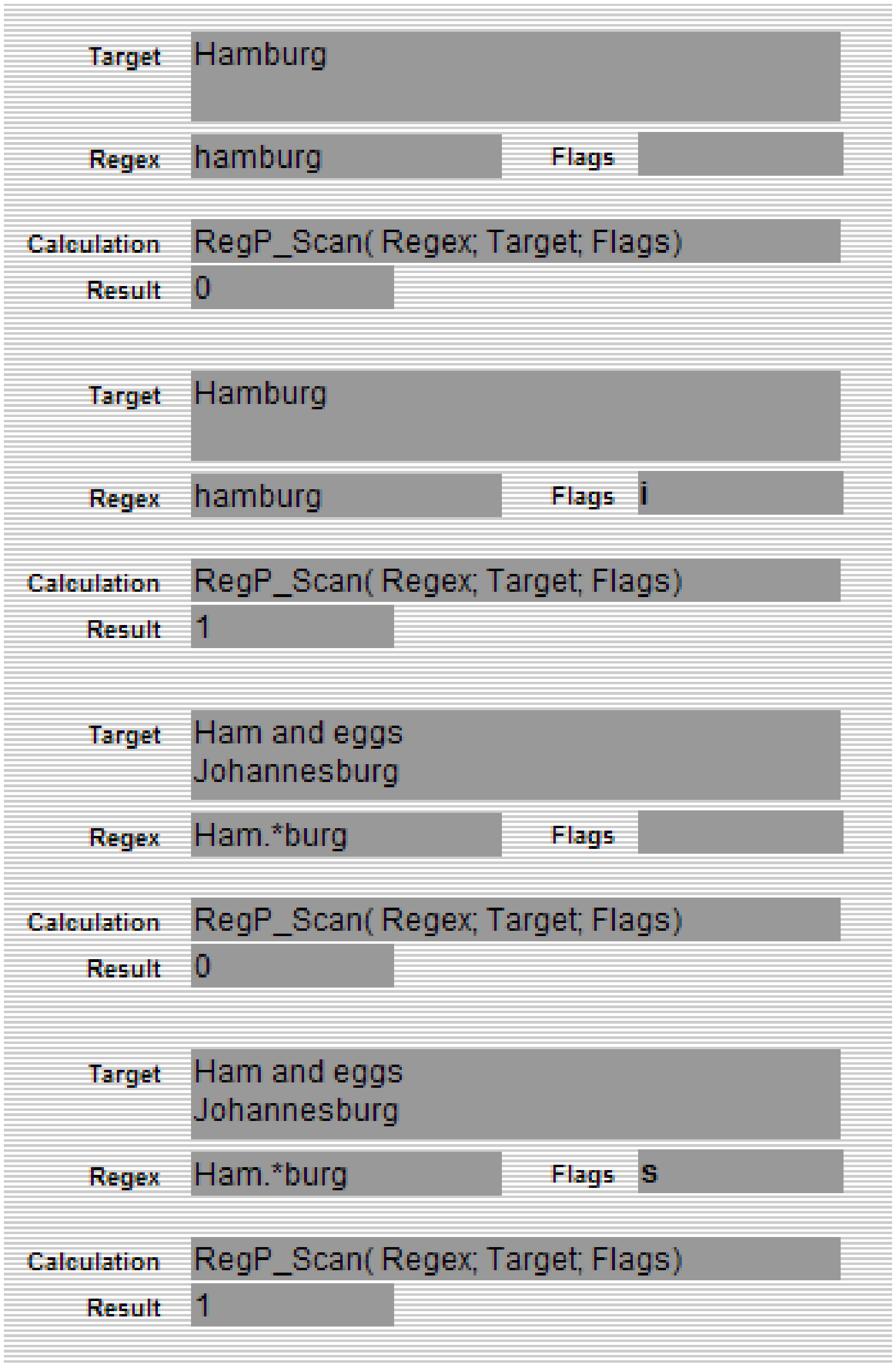
+ RegP_Version
+
This function simply returns the version of **RegexPlugIn** as a string.



Calculation	RegP_Version
Result	0.1.0

+ RegP_Scan (regex ; target {; flags})
+
Searches the text `target` from start to end and tries to match the regular expression `regex`. Returns `True` (1) on success, `False` (0) otherwise.

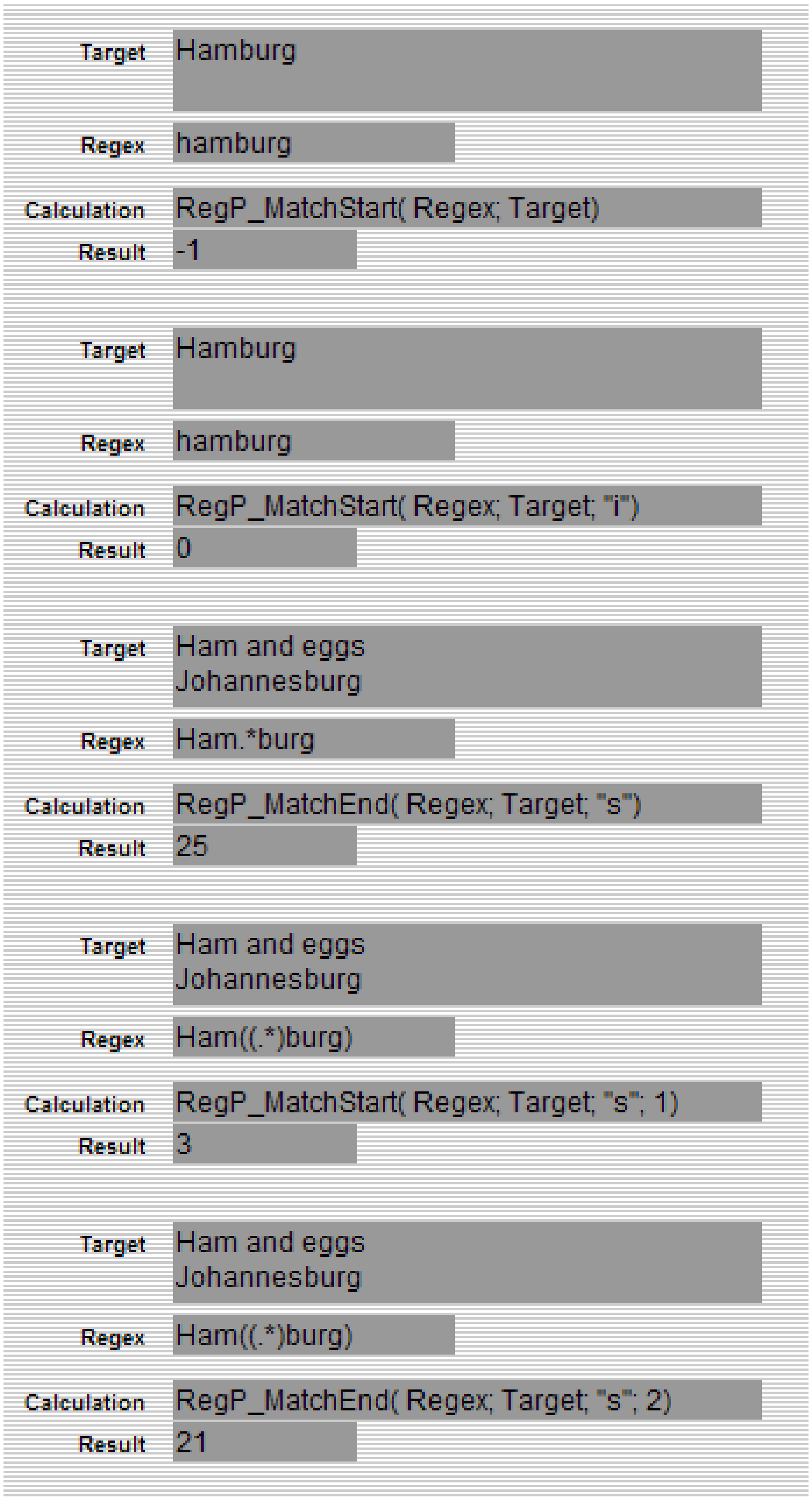
The optional `flags` argument is a string. Upon invocation of the function it is searched for the occurrence of the characters `i`, `s`, `m`, `x`. These characters have the same meaning as the modifiers in Perl, so for example if `flags` is `"is"` or `"si"` (or even `"iS"` or `"SiSi"`), then the regular expression will ignore case and work in "single-line mode."



Target	Hamburg		
Regex	hamburg	Flags	
Calculation	RegP_Scan(Regex; Target; Flags)		
Result	0		
Target	Hamburg		
Regex	hamburg	Flags	i
Calculation	RegP_Scan(Regex; Target; Flags)		
Result	1		
Target	Ham and eggs Johannesburg		
Regex	Ham.*burg	Flags	
Calculation	RegP_Scan(Regex; Target; Flags)		
Result	0		
Target	Ham and eggs Johannesburg		
Regex	Ham.*burg	Flags	s
Calculation	RegP_Scan(Regex; Target; Flags)		
Result	1		

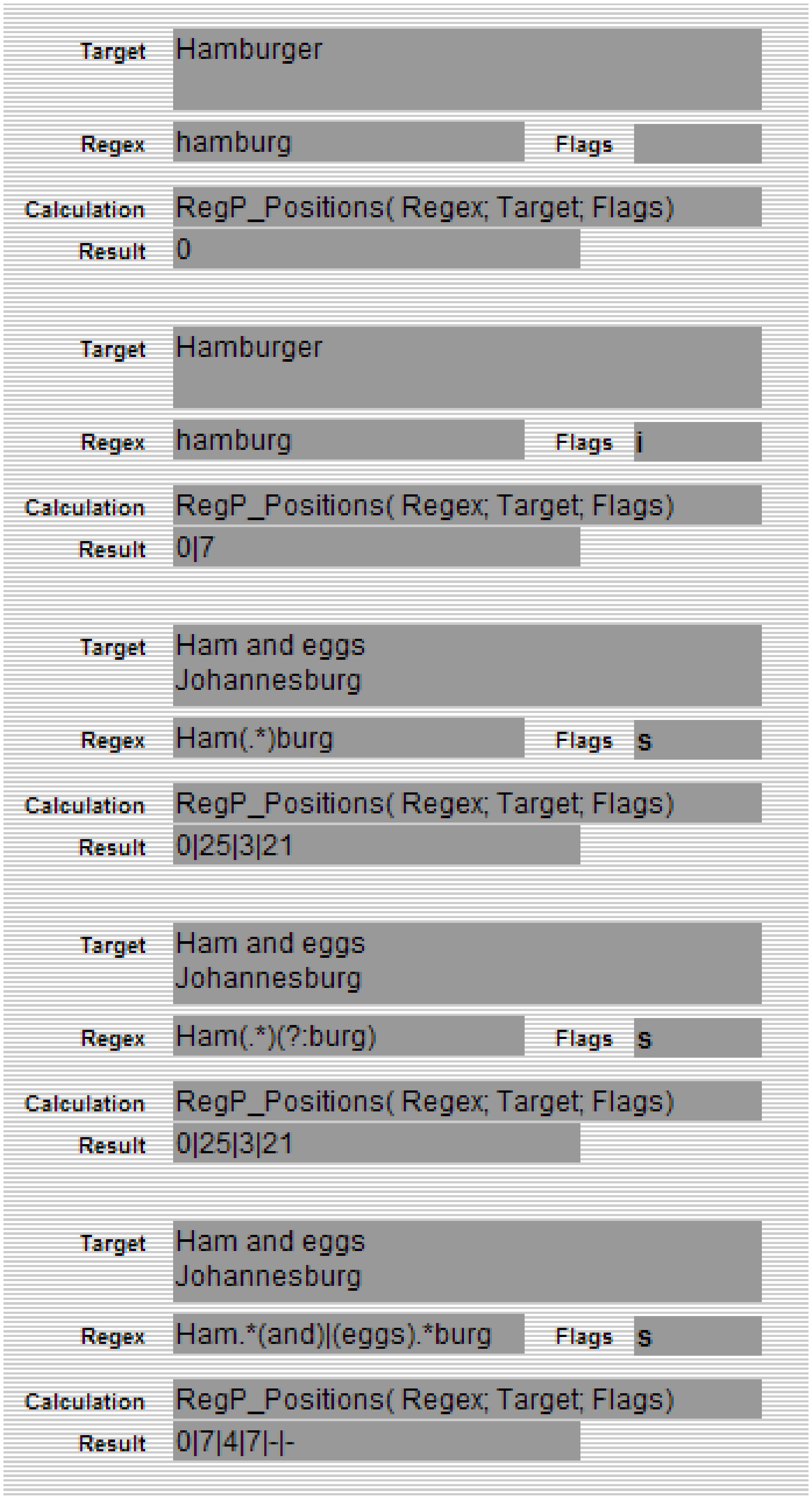
+ RegP_MatchStart (regex ; target {; flags; register})
+ RegP_MatchEnd (regex ; target {; flags; register})
+
These functions return the start and end positions of the matching substring within *`target`* if the regular expression matches, otherwise they return -1. The *`flags`* argument is used as above.

The second optional argument, *`register`*, can be the number of a register group within the regular expression in which case the start and end positions of this group are returned. Note that the first register has the number 1 and *not* 0.



Target	Hamburg
Regex	hamburg
Calculation	RegP_MatchStart(Regex; Target)
Result	-1
Target	Hamburg
Regex	hamburg
Calculation	RegP_MatchStart(Regex; Target; "i")
Result	0
Target	Ham and eggs Johannesburg
Regex	Ham.*burg
Calculation	RegP_MatchEnd(Regex; Target; "s")
Result	25
Target	Ham and eggs Johannesburg
Regex	Ham((.*)burg)
Calculation	RegP_MatchStart(Regex; Target; "s"; 1)
Result	3
Target	Ham and eggs Johannesburg
Regex	Ham((.*)burg)
Calculation	RegP_MatchEnd(Regex; Target; "s"; 2)
Result	21

+ RegP_Positions (regex ; target {; flags ; outerPairDelimiter ; innerPairDelimiter })
+
This function kind of combines `RegP_MatchStart` and `RegP_MatchEnd` (see above). If there is no match, it returns `False` (0). Otherwise it returns a string of the start and end positions of the match itself followed by the start and end positions of all register groups. The individual numbers are separated with the optional delimter arguments which default to `|` character. If a register group doesn't match, two hyphens (`-`) instead of two numbers are inserted into the result string.



Target	Hamburger		
Regex	hamburg	Flags	
Calculation	RegP_Positions(Regex; Target; Flags)		
Result	0		

Target	Hamburger		
Regex	hamburg	Flags	i
Calculation	RegP_Positions(Regex; Target; Flags)		
Result	0|7		

Target	Ham and eggs Johannesburg		
Regex	Ham(.*)burg	Flags	s
Calculation	RegP_Positions(Regex; Target; Flags)		
Result	0|25|3|21		

Target	Ham and eggs Johannesburg		
Regex	Ham(.*)(?:burg)	Flags	s
Calculation	RegP_Positions(Regex; Target; Flags)		
Result	0|25|3|21		

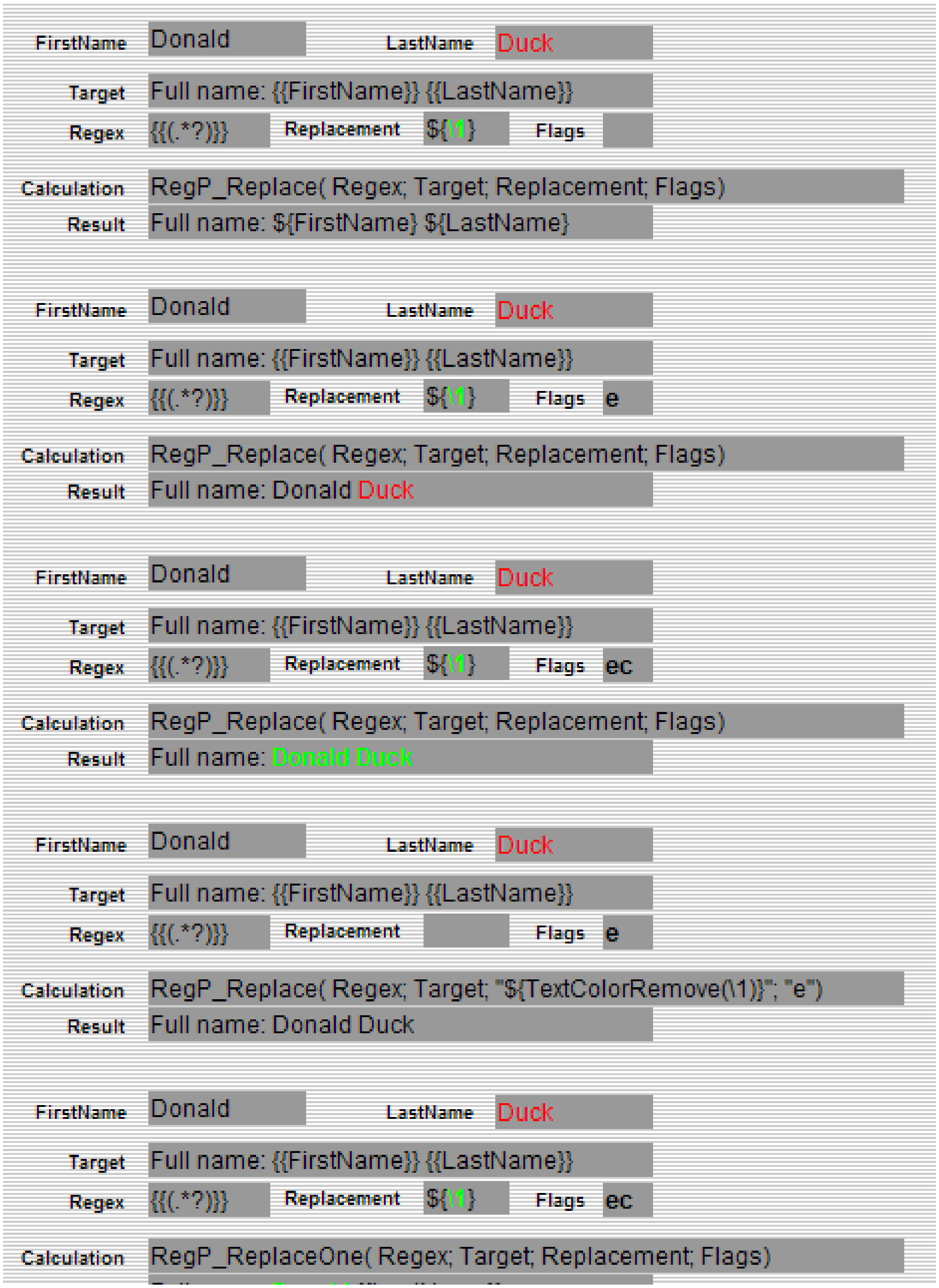
Target	Ham and eggs Johannesburg		
Regex	Ham.*(and)|(eggs).*burg	Flags	s
Calculation	RegP_Positions(Regex; Target; Flags)		
Result	0|7|4|7|-|-		

+ RegP_Replace (regex ; target ; replacement {; flags})
+ RegP_ReplaceOne (regex ; target ; replacement {; flags})
+
`RegP_Replace` returns the text `target` with all substrings that match `regex` replaced with `replacement`. `replacement` can contain the special substrings `\&` for the whole match, `` \` `` for the part of `target` before the match, `\'` for the part of `target` after the match, or `\N` or `\{N}` for the *N*th register group where *N* is a positive integer.

`RegP_ReplaceOne` is similar but replaces only the first match.

The `flags` argument is used as above. Furthermore, if the `flags` argument contains an `e`, then the replacement result is searched for sequences that look like `${...}` where `...` can be anything that doesn't contain the `}` character. These sequences will be replaced by the result of evaluating `...` in FileMaker as with the function `Evaluate`.

`RegP_Replace` and `RegP_ReplaceOne` will usually conserve character styles, i.e. if the first four characters of the target string are red and bold, then the first four characters of the replacement result will also be red and bold unless they are replaced. However, if the `flags` argument contains a `c`, then the special sequences like `\1` or `\&` in `replacement` determine the character style, i.e. if `\1` is in italics and green, then what will be substituted for `\1` will also be in italics and green. In addition, if `flags` also contains an `e` (see last paragraph), then the character style of the `${...}` sequence determines the character style of the evaluation result.



FirstName	Donald		LastName	Duck	
Target	Full name: {{FirstName}} {{LastName}}				
Regex	{{(.*?)}}	Replacement	${\1}	Flags	
Calculation	RegP_Replace(Regex; Target; Replacement; Flags)				
Result	Full name: ${FirstName} ${LastName}				

FirstName	Donald		LastName	Duck	
Target	Full name: {{FirstName}} {{LastName}}				
Regex	{{(.*?)}}	Replacement	${\1}	Flags	e
Calculation	RegP_Replace(Regex; Target; Replacement; Flags)				
Result	Full name: Donald Duck				

FirstName	Donald		LastName	Duck	
Target	Full name: {{FirstName}} {{LastName}}				
Regex	{{(.*?)}}	Replacement	${\1}	Flags	ec
Calculation	RegP_Replace(Regex; Target; Replacement; Flags)				
Result	Full name: Donald Duck				

FirstName	Donald		LastName	Duck	
Target	Full name: {{FirstName}} {{LastName}}				
Regex	{{(.*?)}}	Replacement		Flags	e
Calculation	RegP_Replace(Regex; Target; "${TextColorRemove(\1)}"; "e")				
Result	Full name: Donald Duck				

FirstName	Donald		LastName	Duck	
Target	Full name: {{FirstName}} {{LastName}}				
Regex	{{(.*?)}}	Replacement	${\1}	Flags	ec
Calculation	RegP_ReplaceOne(Regex; Target; Replacement; Flags)				

+ RegP_GetText (regex ; target {; flags ; registers })
+
extracts the first match of pattern regex in target string

+ RegP_GetTextAll (regex ; target {; separationstring ; flags ; registers })
+
extracts ALL matches of pattern regex in target string separated by ‘separationstring’.