King?Macro Guide

In the program you can use various macros to generate content automatically or modify things. Please note that you can use the variable itself from the project used anywhere. E.g. if you defined “Your E-Mail” you can use it in the project like %your e-mail%.
The usage is recursive and allows you to use macros in the content that a macro creates.

Inhaltsverzeichnis

Spin Syntax explained:

A spin syntax always begins with a { and ends with a }.
Multiple items are separated with a |. A random item is used when creating the content.

Example:	*This {is|can be}a test.*
Output 1:	*This is a test.*
Output 2:	*This can be a test.*

As you see above the resulting content can have two variations and a random one is used here. However, you can also use fixed data when using so called *spintags* as in the example below.

Example:	*{#TAG1 I love {this|your} forum|#TAG2 Hello there}. {#TAG1 Best forum ever.|#TAG2 How are you?}*
Output 1:	*I love this forum. Best forum ever.*
Output 2:	*Hello there. How are you?*

In no way would it ever create content with “I love this forum. How are you?” or “Hello there. Best forum ever.”. And as you see in the example above you can also use nested spin syntax.

Another example is the use of URL domain binding like the following...

Example:	*{#.de Hallo Fremder!|#.com Hello Stranger!}*
Output 1:	For a German Site it will write: *Hallo Fremder!*
Output 2:	For a Site ending with .com it will write: *Hello Stranger!*

#file_links[<filename or url>,<number of lines>,<line output>]

This macro will read a random line from a file or URL and put them together according to the <line output> mode. The line output can have the following values:

S	Joins the lines with a single space.
L	Joins the lines with a new link
N	Joins the lines with nothing inbetween.

SP	Same as “S” but reads the number of lines not randomly but one after the other.
LP	Same as “L” but reads the number of lines not randomly but one after the other.
NP	Same as “N” but reads the number of lines not randomly but one after the other.

You can also use a syntax like **(<min lines>,<max lines>)** or **[<min lines>..<max lines>]** for the parameter <number of lines>. This would not always use the same number of lines but chose a random one between the given range (e.g. *#file_links[names.dat,**(2,10)**,S]*).

Example: *I like: #file_links[names.dat,2,S] and all on this forum who contribute to it.*
Output: *These are my friends: Devin Ozz and all on this forum who contribute to it.*

The macro reads two random names from the file “names.dat” and put a space between them.

#file[<filename or url>] or **#file=<filename or url>**

This macro reads the content of a file or URL.

Example: *Something about myself: #file[about_yourself.dat]*
Output: *Something about myself: I am a poor coder and need your money ;)*

The macro reads the whole content of the file “about_yourself.dat” and inserts it into the text.

#grabbed or **#grabbed[<line output>]**

This will try to insert content of grabbed content from the present website. The file xgrab.txt can hold the definitions on how to get the content in the following way...

```
<h1>[...]</h1>
<title>[...]</title>
```

The content is collected (first match on the page only) and joined together the way that line output is defined.

S	Joins the lines with a single space (default).
L	Joins the lines with a new link
N	Joins the lines with nothing in between.

Example: *I read “#grabbed[N]” and had to smile :)*
Output: *I read “Welcome to my Forum” and had to smile :)*

Assuming the webpage has a title like “*Welcome to my Forum*” and your *xgrab.txt* file has something defined like “*<title>[...]</title>*” you get that output. Unfortunately I couldn't find any decent *xgrab.txt* file to play with so you have to create your own or copy it from an other program that also uses this.

#gennick[XYZ,<min length>,<max length>]

This macro generates a random nick name with a character length between min length and max length. The first parameter is ignored for now. Just put anything you want for it.

Example:	*Hello #gennick[XYZ,5,10]*
Output:	*Hello DevinSan*

In the example above the macro created a nickname with a length of 8 characters and inserted it in the text.

#random[<range>]

This inserts a range of characters in the text. A range can have the following syntax:

#random[a,b,c]	This will either insert a, b or c into the text.
#random[0..9,A..Z]	This will insert a number or an upper case character into the text.

Example:	*My password is: test#random[0..9]#random[-,_]#random[a..z,A..Z]*
Output:	*My password is: test3-H*

The example above inserted 3 characters according to the used ranges in the macro.

#err[<number>]

This macro is not inserting anything but indicates that the whole text for this content should be modified with spell/type errors. The number can go from 1 to 10000 and indicates how often spell/type errors should be created for a word. If you use 10000 you will have almost every word in the text spelled wrong.
But why you might ask yourself now is this of any use? Normally you want to create perfectly written content and not have your text with spell errors. Some people prefer this to not create duplicate content and have it look more natural as spell errors happen to everyone of us.

Used spelling errors are:

skip character	*example* becomes *exaple* (missing the m)
double character	*example* becomes *exxample* (x repeated)
reverse two characters	*example* becomes *exmaple* (a and p swapped)
use next character on keyboard	*example* becomes *exsmple* (s is next on keyboard to a)
additional character on keyboard	*example* becomes *exsample* (s is next on keyboard to a)

#trans … #notrans

This macro tries to translate the text between #trans and #notrans to the language used on the site where it is submitting to. Please note that the language detection on the site might not always work and that this translation is done online. If the translation fails, it will use the original text. The original text can be in any language as the translation tries to auto detect the language, but I would suggest you to use English here.

Example: *#trans The minute that we are born we begin to die. #notrans*
Output: *Die Minute, die wir geboren werden, beginnen wir zu sterben.*

The example above is translating the sentence "*The minute that we are born we begin to die.*" into "*Die Minute, die wir geboren werden, beginnen wir zu sterben.*" while we submit to a German site like *http://www.something-german.de/*

%random-<min>-<max>%

This is generating a random number between min and max.

Example: *I was born on %random-1960-1980%.*
Output: *I was born on 1968.*

The example above generated a random number between 1960 and 1980 and finally inserted 1968.

%columnspinfile-<filename>-<column>%

This macro is reading a random line from the file and taking the set column (seperated by a , from it. If you use the macro with the same file name but a different column, than that exact line is used and not again a new random line. This makes it possible to e.g. have a file with address data and use the same content that belongs to one address instead of unrelated data.

Example:
I life in %columnspinfile-address_data.dat-1% - %columnspinfile-address_data.dat-3%.

Output:
I life in Australia – Amaroo.

Lets have a quick look at the file "*address_data.dat*":

```
...
Australia,ACT,Amaroo,2914,37 Parkes Road,(02) 6114 7945,
United States,OK,Oklahoma City,73109,4310 Meadow Drive,405-990-7523,
…
```

As you see it has one address each line and seperated by a ",".
The content is: *Country,State,City,ZIP,Street,Phone*.
Now the program takes a random line out of that file and uses that content to insert the

different columns into the text where *%columnspinfile-address_data.dat-1%* is the country and *%columnspinfile-address_data.dat-3%* is the city.

%spinfolder-<folder>%

%spinfolder2-<folder>%

%spinfolderdelete-<folder>%

This macro is using a random file from the folder and using that as content.
Also note that the program is trying to match the same file name or file number when locating a file in a different folder. E.g. if *%spinfolder-c:\my_articles\%* is using file "*article6.txt*" than it tries to use "*summary6.txt*" (or anything with the number 6 in file name) from the macro *%spinfolder-c:\my_article_summaries\%*.

%spinfolder2% is not keeping the previously used content in mind and %spinfolderdelete% is deleting the file after using it to not use it again.

Example:	*%spinfolder-c:\my_articles\%*
Output:	*This is the content of article 6 which I have put in a folder....*

%spinfile-<filename>%

This is reading a random line from the file. Though the same line is used if this macro is used more than once on a submission with the same file.

Example:	*My name is %spinfile-names.dat% %spinfile-lnames.dat%.*
Output:	*My name is Earl Grey.*

The example above is using the macro two times but on different files (so it is always using a random line, just when you later use the *%spinfile-names.dat* or *%spinfile-lnames.dat%* on the same site it is using the same content).

%file-<filename>%

This is the same as using the #file[<filename or url>] macro but it is not accepting an URL.

Example:	*%file- c:\my_articles\article6.txt%*
Output:	*This is the content of article 6 which I have put in a folder....*

%keyword%

This will insert a random keyword that you have defined in your project into the text. Please don't use %keywords% (note the s) as that would insert all keywords into the text and not just one.

Example: *My site is all about %keyword% and %keyword%*
Output: *My site is all about division and joy .*

The example above is taking two times a random keyword (in this case joy and division) from your project and inserts it.

%keyword<number>%

This macro would insert not a random keyword but take the exact order from the project setting. %keyword1% is taking the first defined keyword, %keyword2% the second and so on.

Example: *My site is all about %keyword1% and %keyword2%*
Output: *My site is all about joy and divistion.*

Assuming you have "joy, division, music, ..." defined in your project as keywords you would get the above output as it takes %keyword1% which is joy and %keyword2% which is division.

%source_url%

This is replaced with the URL that the program "clicked" to get to the current page. It can be seen as a referrer page used in HTML headers e.g. to indicate where the User came from.

Example: *I clicked on a link from %source_url% and came here.*
Output: *I clicked on http://www.gsa-onlione.de and came here.*

If the current URL is now http://www.gsa-online.de/contact.php but was previously on http://www.gsa-onlione.de you would get the output from above.

%targeturl% or %blogurl% or %blog_url%

This is replaced with the current URL the program is on.

Example: *I clicked on a link from %source_url% and now I am on %targeturl%.*
Output: *I clicked on http://www.gsa-onlione.de and now I am on http://www.gsa-online.de/contact.php.*

This example uses the previous %source_url% macro as well to demonstrate what the content of the macros might be.

%targetsubdomain%

This macro inserts the full domain of the site we submit to.

Example: *Nice to be on % targetsubdomain%.*
Output: *Nice to be on forum.gsa-online.de.*

%targetdomain%

This macro will insert the main domain without the suddomain as in the macro above.

Example: *Nice to be on % targetdomain%*
Output: *Nice to be on gsa-online.de.*

%targethost%

This macro will insert the host together with http:// or https:// into the text.

Example: *Now the whole site %targethost% is great.*
Output: *Now the whole site http://www.gsa-online.de is great.*

%targetpath%

This macro is just taking the path from the URL the program is on.

Example: *I am surfing on %targethost%%targetpath%.*
Output: *I am surfing on http://www.gsa-online.de/contact.php*

In this sample we used two macros as this macro alone is not making much sense in most cases.

%targetparameters%

This macro is just taking the parameters of an URL (including the ?-Sign).
Also note that if there is no parameter used for the URL it is left empty.

Example: *I am surfing on %targethost%%targetpath%%targetparameters%.*
Output: *I am surfing on http://www.gsa-online.de/contact.php?lang=de*

In this case we used all tree macros from above.

%original_targeturl%

%original_targethost%

%original_targetpath%

%original_targetparameters%

These macros are referring to the very first URL used when the submission started. The output is the same as the one without the *original_* prefix.

%url_domain%

This is replaced with the domain of your URL that you want to submit.

Example: *I am working a lot on my site at %url_domain% lately.*
Output: *I am working a lot on my site at gsa-online.de lately.*

%url_path%

This is replaced with the path of the URL that you want to submit.

%url_parameters%

This is replaced with the parameter of the URL that you want to submit.

%random_email% or %email%

This is generating a random email address with a correct syntax and an existing domain. The program tries to reuse the value used from "names.dat" (random names) to generate an email that looks related to your name.

Example: *My email is %random_email%.*
Output: *My email is blah@hotmail.com.*

%emailuser%

This macro will insert the content of an email that you see before the @ sign.

Example: *My email is %random_email% so the thing before the @ is %emailuser%.*
Output: *My email is blah@hotmail.com so the thing before the @ is blah.*

The example above might not make a lot sense when using in your project content but it might be required for scripting things together.

%emailhost%

This macro will insert the content of an email that you see after the @ sign.

Example: *My email is %random_email% so the thing after the @ is %emailhost%.*
Output: *My email is blah@hotmail.com so the thing before the @ is hotmail.com.*

Again this example might not make much sense when using in your project content but it might be required for scripting things together.

%name%

This generates a random name (first name only) and would be the same as when using the macro %spinfile-names.dat%.

Example: *My name is %name%.*
Output: *My name is Earl.*

%website% or **%url%**

This will insert the URL you are trying to submit.

Example: <a href="%url%">%anchor_text%</a>
Output: <a href="http://www.gsa-onlione.de/">GSA</a>

Assuming that the URL is defined as http://www.gsa-online.de/ and the anchor text as GSA you would get the above content.

%blogtitle% or **%blog_title%**

The title of the current URL you are about to submit to (everything between html tag <title></title>) is used here.

Example: *Wow "%blogtitle%" is indeed an interesting topic.*
Output: *Wow "How to do things." is indeed an interesting topic.*

Assuming the website you submit to has *<title>How to do things.</title>* in there html (the one you see in top of your browser) you would get the above content.

%meta_keyword%

A random meta keyword from the site you are submitting to.

Example: *Nice to see someone writing about %meta_keyword%.*
Output: *Nice to see someone writing about SEO.*

Assuming that the website you submit to has
<meta name="Keywords" content="Google Ranking, SEO, Website Submission" />
you would get the above content.

%image_title%

This is used for image comment engines only and will have the image title in it (or "this image" if not found).

Example: *Very nice image. I think "%image_title%" is the best of all.*
Output: *Very nice image. I think "Sunset in Rostock" is the best of all.*

Assuming that the image you want to place a comment on is called "*Sunset in Rostock*" you would get the above content.

%random_url%

This will insert a random URL from your project setting. The difference between %url% is that it is purely random and not saved during submission. This makes it possible to insert not only one URL in an article, but also another one from your project.

Example: *My Website is %url% and %random_url% as well as %random_url%*

Output: *My Website is http://url1.com and http://url1911.com as well as http://url23.com*

If you would have used %url% only, you would have three of the same URLs in the output.

%random_anchor_text%

This will work the same way as %random_url%. However it tries to stay connected tot he last %random_url% used. This might not always work so handle with care.

%random_video% %random_image%

This will insert a random image or video to your content (e.g. in article) to make it look more natural. It uses the sites defined in content_search.dat to search for content related to your article (using your keywords/anchor text).

Please note that even though a proper author and source is added to the image/video it might still not be enough in terms of copyright notes.

%url<number>%

Uses the exact URL from your project (first is %url1%). If you have defined less URLs in the project, it will use a random one.

%anchor_text<number>%

Uses the exact anchor text from the URL in your project. If no anchor_text was defined with that URL it is handled the same as %anchor_text%.

%verified_url%

Uses a random URL from your verified URLs.

%verified_anchor_text%

Uses the anchor text of the last used verified URL that was used from %verified_url%

%verified_domain%

Uses a random domain from your verified URLs.

%verified_host%

Uses a random host from your verified URLs.

%first_paragraph-<varname>%

This will take the content of the the variable and only insert the first paragraph.

%cookie-<cookie_name>[;<URL>]%

This will get you the cookie value of the cookie_name. If you leave that URL out, you get the data from the current URL. The path is ignored from the URL.

%datetime-<format_string>%

This will get you the current date time according to the format string. Below are some exambles:

y	= Year last 2 digits
yy	= Year last 2 digits
yyyy	= Year as 4 digits
m	= Month number no-leading 0
mm	= Month number as 2 digits
mmm	= Month using ShortDayNames (Jan)
mmmm	= Month using LongDayNames (January)
d	= Day number no-leading 0
dd	= Day number as 2 digits
ddd	= Day using ShortDayNames (Sun)
dddd	= Day using LongDayNames (Sunday)

ddddd	= Day in ShortDateFormat
dddddd	= Day in LongDateFormat
c	= Use ShortDateFormat + LongTimeFormat
h	= Hour number no-leading 0
hh	= Hour number as 2 digits
n	= Minute number no-leading 0
nn	= Minute number as 2 digits
s	= Second number no-leading 0
ss	= Second number as 2 digits
z	= Milli-sec number no-leading 0s
zzz	= Milli-sec number as 3 digits
t	= Use ShortTimeFormat
tt	= Use LongTimeFormat
u	= Unix time stamp

am/pm	= Use after h : gives 12 hours + am/pm
a/p	= Use after h : gives 12 hours + a/p
ampm	= As a/p but TimeAMString,TimePMString
/	= Substituted by DateSeparator value
:	= Substituted by TimeSeparator value

Important : if you want to see characters such as dd in the formatted output, placing them in " marks will stop them being interpreted as date or time elements.

In addition to this formatting, various of the above options are affected by the following variables, with their default values :

DateSeparator	= /
TimeSeparator	= :
ShortDateFormat	= dd/mm/yyyy
LongDateFormat	= dd mmm yyyy
TimeAMString	= AM
TimePMString	= PM
ShortTimeFormat	= hh:mm
LongTimeFormat	= hh:mm:ss
ShortMonthNames	= Jan Feb ...
LongMonthNames	= January, February ...
ShortDayNames	= Sun, Mon ...

LongDayNames	= Sunday, Monday ...
TwoDigitYearCenturyWindow	= 50

%data_url-<file>%

This will insert the content of a file encoded in base64 and also adds the rest required to use this for DATA URI SCHEME.

Example: <img src="%data_url-c:\images\profile.jpg%" />

Output: <img src="data:image/png;base64,iVBORw0KGgoAA==" />

%spinfolder_data_url-<folder>%

Same as %data_url% macro but using a random file from that folder.