

## Notes on Using the GAMS Put\_utility Command

Bruce McCarl  
Regents Professor of Agricultural Economics  
Texas A&M University  
April, 2008

GAMS currently does not allow one to easily manipulate strings however put file commands allow one to write out strings composing them based on set element text. There are cases where when executing external programs or in reading and writing information to external files that such string manipulation can be valuable. For example one might want to read in a large number of GDX files in a similar manner or execute external programs with altering parameters. The `put_utility` or `put_utilities` commands in GAMS allow such functions.

In general the `put_utility` language feature works in the following way. Two lines need to be generated with the `put_utility` for each command to be executed

- The first line tells what type of `put_utility feature` to use where the allowed features and their functions are given below.
- The second line gives the `arguments` to use with that `feature`
- The general syntax is

```
File nameoffiletouse  
Put nameoffiletouse  
Put_utility localname 'feature1' / 'arguments' ;  
Put_utility 'feature2' / 'arguments' ;
```

or

```
put_utility 'key' / 'arguments' / 'key' / 'arguments' /.... ;
```

### Notes

The `localname` is optional and gives the local name of the file that is to be used to pass instructions.

In the `put_utility` command the `"/` separates the lines as in ordinary put commands. Note an ending `"/` is not used although one may stack features as shown in the example below

The names of the `features` or `keys` above that can be used and nature of associated `arguments` are

<code>Feature</code> name	Nature of General Function	Nature of associated <code>arguments</code>
exec	Causes a command to be passed to the operating system for execution	Command to be executed with arguments
shell	Causes a command to be passed to the command	Command to be processed by shell processor then

	shell processor that in turn is passed to operating system for execution	passed on in processed form to the operating system (Note distinctions between shell and exec are technical and can be operating system specific. They typically involve the ability to use redirect of standard input output and the error console)
gdxin	Causes subsequent GDX file loaded by execute_load or execute_loadpoint to come from a specified file name	Name of the GDX file that data will be loaded from
gdxout	Causes subsequent GDX file unloaded by execute_unload to come from a specified file name	Name of the GDX file to which data will be unloaded
ren	Causes put file output to be directed to a named external file	File name to use for subsequent put operations
inc	Causes the contents of a file to be incorporated into the currently active put file	File name whose contents are to be included
click	Causes a clickable file reference to be added to the process window	File name for file to which reference will point
msg	Causes a message to be placed in the listing file	Text of message
log	Causes a message to be placed in the log file (also process window)	Text of message
msglog	Causes a message to be placed in both the log and listing files	Text of message
title	Causes the title on the DOS window to be changed	New name for window
glb	Used by GAMS to aid in building model library – not intended for users	--
ren	Used by GAMS to aid in building model library – not intended for users	--

The arguments are typically in quotes and limited to 255 characters although in the messages, exec and shell cases multiple quoted elements can be included each up to a maximum of 255 characters and will be included in information passed on.

The advantage in all of these is that you can assemble the command line at run time (e.g. selecting the filename or command line arguments based on some set elements).

## Examples

\*write stuff to different files

```
loop(i,  
    random = uniformint(0,100);  
    put_utility 'shell' / 'echo ' random:0:0 ' > ' i.tl:0;  
);
```

\*Put data in several.gdx files then reloads it

```
file fx2;  
put fx2;  
set ij / 2005*2007 /;  
scalar random;
```

\*put out the data to multiple GDX files

```
loop(ij,  
    put_utility 'gdxout' / 'data' ij.tl:0;  
    random = uniform(0,1);  
    execute_unload random;  
);
```

\*Load the data from multiple GDX files

```
loop(ij,  
    put_utility 'gdxin' / 'data' ij.tl:0 ;  
    execute_load random; display random;  
);
```

```
file dummy; dummy.pw=2000; put dummy;
```

\*here I execute some commands

```
put_utility 'exec' / 'gams sets' /  
            'shell' / 'dir *.gms' ;
```

\*here I enter a clickable link

```
put_utility 'click' / 'sets.gms' ;
```

\*here I vary where the put file output goes

```
loop(i,  
    put_utility 'ren' / i.tl:0 '.output' ;  
    put "output to file " i.tl:0 " with suffix output " /;  
);
```

\*here i put messages in the LOG and LST files

```
put_utility 'msg' / 'message to lst file' /  
            'log' / 'message to log file' /  
            'msglog' / 'message to log and lst file' ;
```

\*here i put some text in the put file

```
file junk;  
put junk;  
put_utility 'inc' / 'addit.txt' ;  
put_utility 'inc' / 'sets.gms' ;
```