SnippetManager for PowerGUI

(C) 2011 Denniver Reining / bytecookie.wordpress.com / v1.0.1

1. What it is

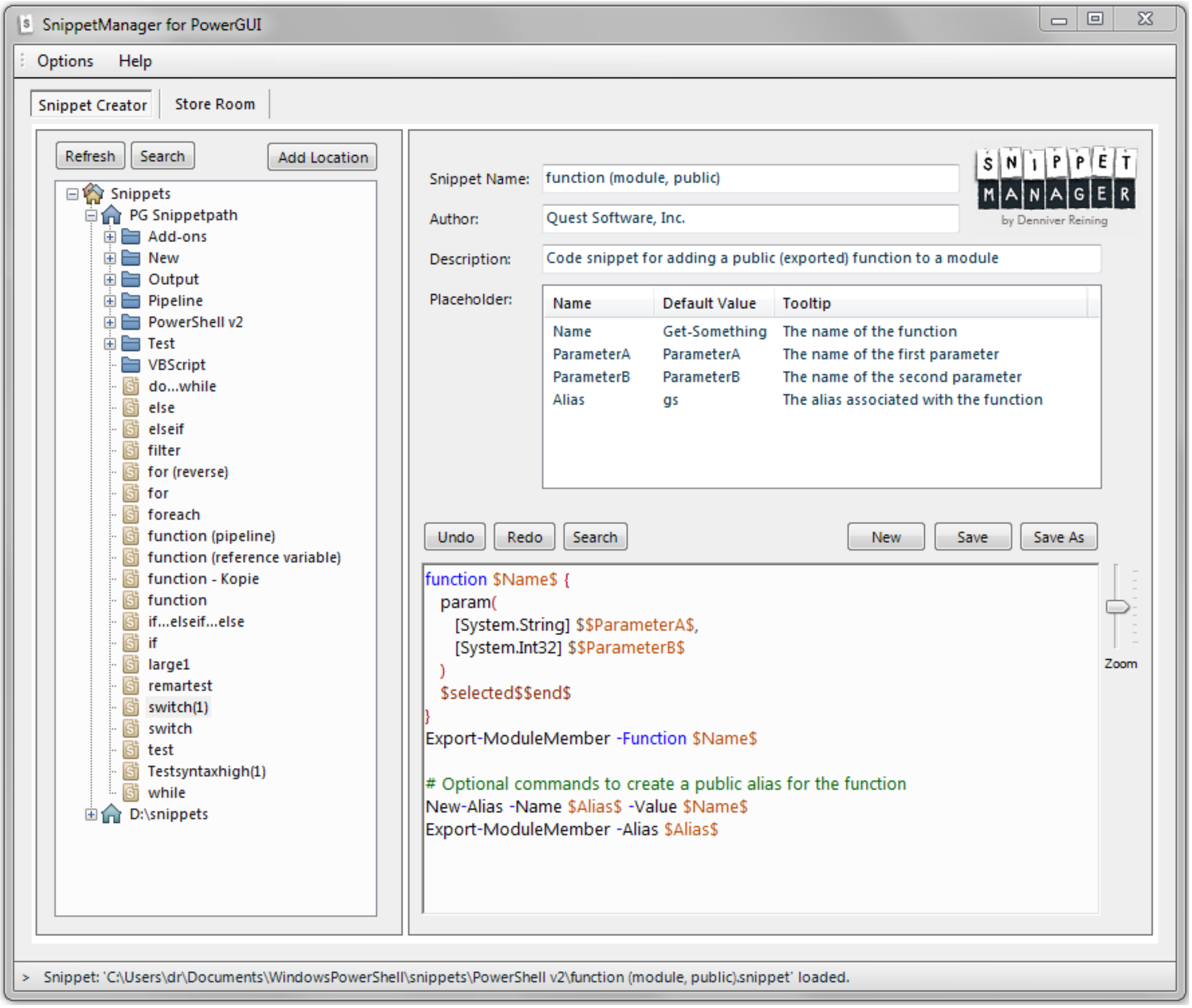
With the SnippetManager add-on you can manage your code snippets. You can easily create new snippets, edit existing ones, create new categories and much more.

Snippets?

If you never worked with snippets before, here is a short explanation:

Snippets are code fragments or short scripts that you save for later, so that you can use them again in another script. Once you assigned a name to your snippet, you can easily paste them in a script, using the "snippets"-menu entry in the context menu of PowerGui Script Editor.

In addition to that, you can replace parts of the snippet code (e.g. function names, variables or anything that you probably would change when you use that code again) with a placeholder. This placeholder will be highlighted when you paste the snippet to your script, and you can simply put in a new value for the placeholder.



Snippet Name: function (module, public)

Author: Quest Software, Inc.

Description: Code snippet for adding a public (exported) function to a module

Placeholder:

Name	Default Value	Tooltip
Name	Get-Something	The name of the function
ParameterA	ParameterA	The name of the first parameter
ParameterB	ParameterB	The name of the second parameter
Alias	gs	The alias associated with the function

```
function $Name$ {
  param(
    [System.String] $$ParameterA$,
    [System.Int32] $$ParameterB$
  )
  $selected$$end$
}
Export-ModuleMember -Function $Name$

# Optional commands to create a public alias for the function
New-Alias -Name $Alias$ -Value $Name$
Export-ModuleMember -Alias $Alias$
```

> Snippet: 'C:\Users\dr\Documents\WindowsPowerShell\snippets\PowerShell v2\function (module, public).snippet' loaded.

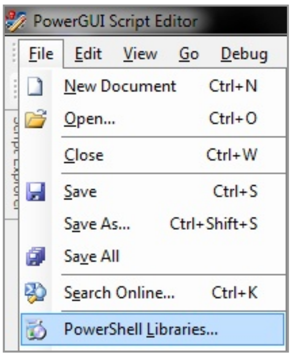
2. Installation

- **PowerGUI Version 2.3** (or higher) **is required** for this version (1.0.1) of Snippet Manager

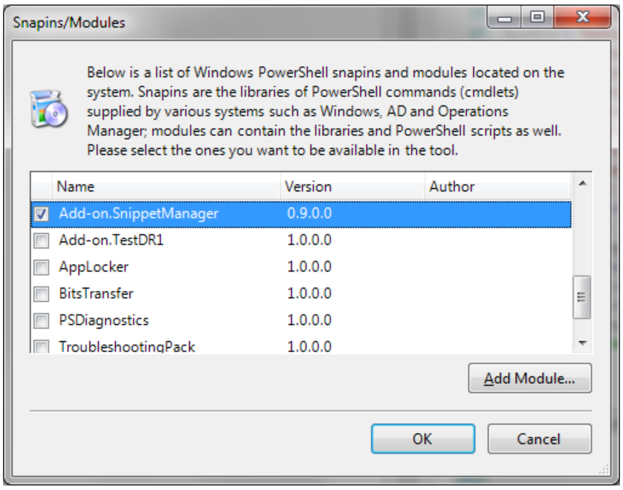
- **Download** and **run** the SnippetManager.msi-installer**.** All necessary files and folders will be created automatically.

- **Start** your PowerGUI Script Editor.

  - Click on "***File***" in the menu bar and then "*Powershell Libraries…*".



  - In the "*Snapins/Modules*"-window, locate the "*Add-on.SnippetManager*"-entry, select the **checkbox** and click "*OK*".


Snapins/Modules

Below is a list of Windows PowerShell snapins and modules located on the system. Snapins are the libraries of PowerShell commands (cmdlets) supplied by various systems such as Windows, AD and Operations Manager; modules can contain the libraries and PowerShell scripts as well. Please select the ones you want to be available in the tool.

Name	Version	Author
☑ Add-on.SnippetManager	0.9.0.0	
□ Add-on.TestDR1	1.0.0.0	
□ AppLocker	1.0.0.0	
□ BitsTransfer	1.0.0.0	
□ PSDiagnostics	1.0.0.0	
□ TroubleshootingPack	1.0.0.0	

- **Done!**

3. About the snippet paths (Important!)

PowerGUI stores its (shipped) **snippets** in a subfolder of its **installation folder**. Usually, that is:

- *C:\Program Files\PowerGUI\snippets*

You may know, as a **security measure** in Windows 7 and Vista, no program (or user for that matter) without admin privileges is allowed to write to the programs dir. If you try it by hand, you get an elevation prompt. If a program tries it, on some systems you get an error, on some systems the *folder redirection* kicks in and redirects every change you made to a folder called virtual store. This causes a whole bunch of problems.
To avoid these problems and still be able to change the shipped snippets, there was an **elevation prompt** popping up, every time you started Snippet Manager. That was really annoying.

Since version **2.3**, PowerGUI uses a **second snippet path** per default:

- *C:\Users\[yourname]\Documents\WindowsPowerShell\snippets*)

This path, being in the **user profile**, is fully accessible to the user. So when you first run Snippet Manager, it asks you if you want to move these snippets from the installation folder to the user profile. You then have the following **options**:

- You can **move them**. That means:
  - You will be able to **edit and deactivate/reactivate all of the snippets**.
  - Only the user profile snippet path will be added automatically to the snippet explorer in SnippetManager.
  - If SnippetManager detects snippets in the PowerGUI install directory (e.g. after an PowerGUI update) you will be asked again.

- **Or** you can **leave them** in the installation folder. That means:
  - Both paths (install dir and user profile) will be added automatically to the snippet explorer in SnippetManager and **you can only edit or deactivate/reactivate the snippets in the user profile**.
  - **But**, and this is very important(!), **both paths will be used by PowerGUI** when you open the *"Insert Snippet*"-menu. If there are snippets with equal names in both folders, the one in the user profile will **take precedence** over the other one.
  - You won't be asked again about this. But if you should later decide to move the snippets after all, i recommend not doing it by hand, but by clicking the menu item "*HELP\Reset all settings…*", because in this case you want SnippetManager to watch the "install dir/snippets"-folder for changes in the future.

I highly recommend the first option.

4. How it works

You can **access SnippetManager** via the new menu entry "Tools/ *SnippetManager*".

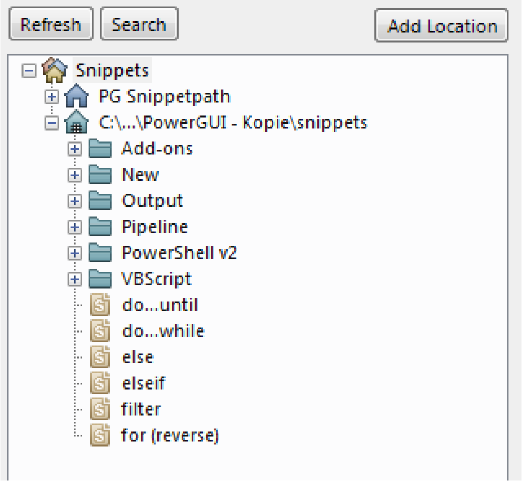
4.1 The Snippet Explorer

- The snippet explorer is on the left side of the main window. Here you have standard explorer (right-click) functionality (Copy, Paste, Delete, Rename etc.) except for two limitations: no drag and drop and no multi-selection of items. (This will be implemented in a later version of the add-on though.)

  In order to compensate for that, there is the "***Show in Windows Explorer***"-context menu item, where you can go directly to the item and make any kind of mass actions via Windows Explorer. If you get back to the SnippetManager just click "***Refresh***" to see your changes.

- To **open a snippet**, just double click on it.

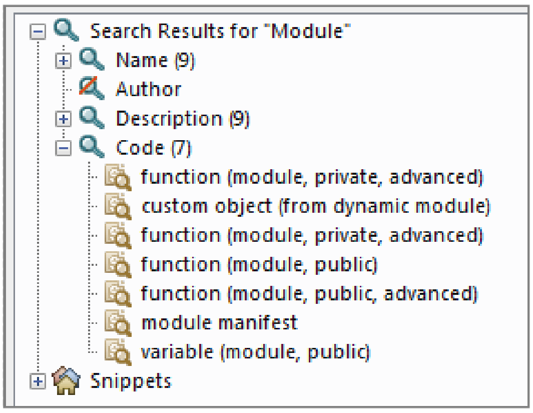
- You can **add** other snippet folders ("**Locations**") to the snippet explorer by clicking "Add Location".
  - You should **add folders with snippets** directly, instead of adding e.g. whole drives, cause this can slow down file operations considerably.
  - If you add a location you have no write access to, the location-icon will have bars in front of the door to remind you of that fact. :)
  - You can add UNC-paths manually by clicking on "*Options\Add Network Location...*"



- You can **remove an location** by right-clicking on the location root and choosing "Remove..."

4.1.1 Search for Snippets

- Via the **search** button, you can search in every part of every snippet the snippet explorer. Search results will be displayed as a new tree in the explorer for easy access.


Search Results for "Module"
Name (9)
Author
Description (9)
Code (7)
function (module, private, advanced)
custom object (from dynamic module)
function (module, private, advanced)
function (module, public)
function (module, public, advanced)
module manifest
variable (module, public)
Snippets

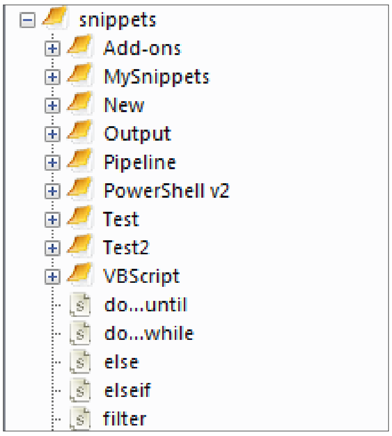
4.2 File Names, Snippet Names and Folders

- **File Names**: this is how the snippet *file* is named and what shows up in the snippet explorer. They **do not appear** in the snippet-menu in PowerGUI.

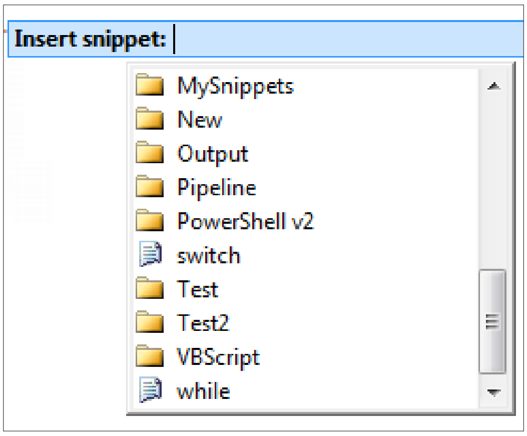
- **Snippet Name**: The snippet name is shown (and can be changed) on the right side of the main window. This name is what **you see** in the snippet menu in PowerGUI.

- **Folders**: The folders that are shown (and can be created and edited) in the snippet explorer, **also act as categories** for the snippet menu in PowerGUI. A good way to group your snippets and keep the snippet menu tidy.

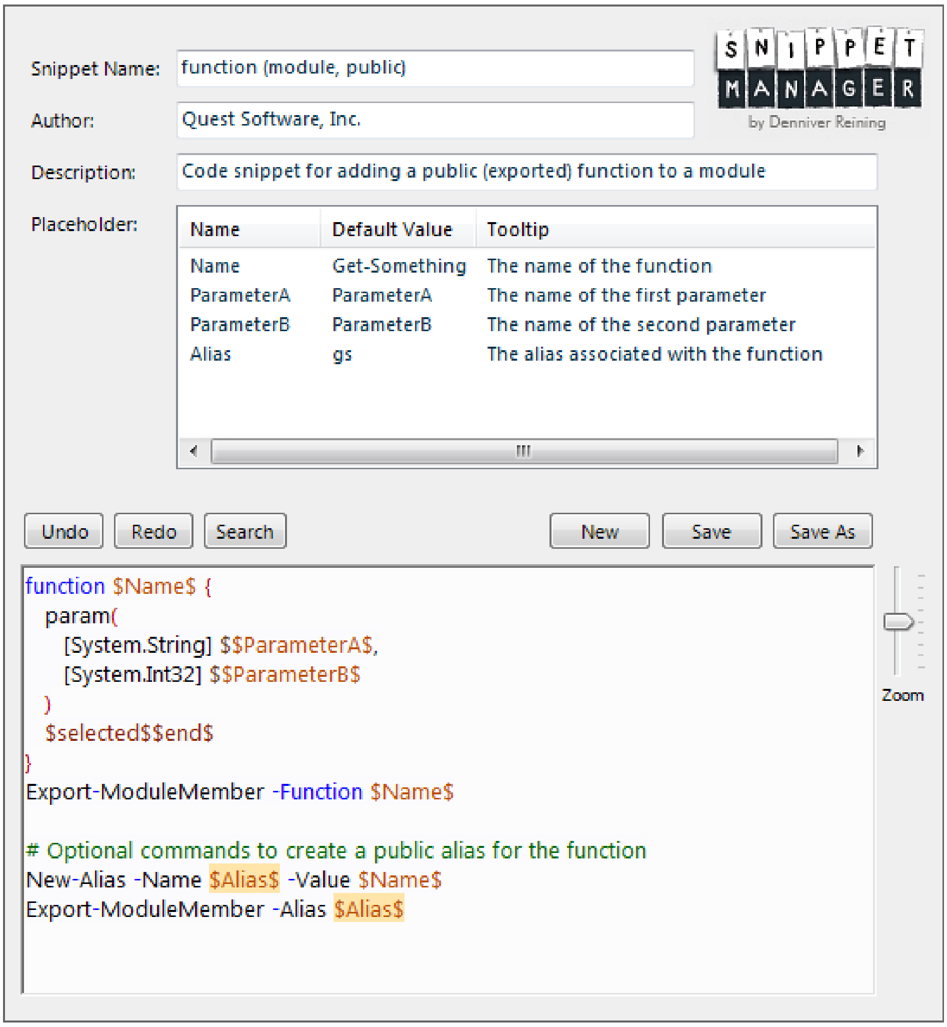
Snippet Explorer:



Snippet menu in PowerGUI:



4.3 The Snippet Window



Snippet Name: function (module, public)

Author: Quest Software, Inc.

Description: Code snippet for adding a public (exported) function to a module

Placeholder:

Name	Default Value	Tooltip
Name	Get-Something	The name of the function
ParameterA	ParameterA	The name of the first parameter
ParameterB	ParameterB	The name of the second parameter
Alias	gs	The alias associated with the function

```
function $Name$ {
  param(
    [System.String] $$ParameterA$,
    [System.Int32] $$ParameterB$
  )
  $selected$$end$
}
Export-ModuleMember -Function $Name$

# Optional commands to create a public alias for the function
New-Alias -Name $Alias$ -Value $Name$
Export-ModuleMember -Alias $Alias$
```

- The ***Snippet Name*** is what you see in the snippet menu in PowerGUI, its mandatory for every snippet.

- ***Author and Description*** are optional, but a description is a good idea, it shows up as tooltip in the snippet explorer, and as well in the snippet menu in PowerGUI.
TIP: You can add a **standard** value for the **Author** field under *"Options/Auto Author" and* every time you hit *"New"*, the Author-field will be filled in automatically..

- The **Code box** is where the snippets code sits. (surprise!) There are **two ways to paste code** to this box, either by pressing *CTRL+V* or by the "paste"- *context menu* item. The **difference** is, when you paste it via the shortcut, Snippet Managers syntax highlighting (which is basic) does not apply, instead you see the original syntax highlighting from PowerGUI (if you copied the code from there). Decide for yourself what suits you better.

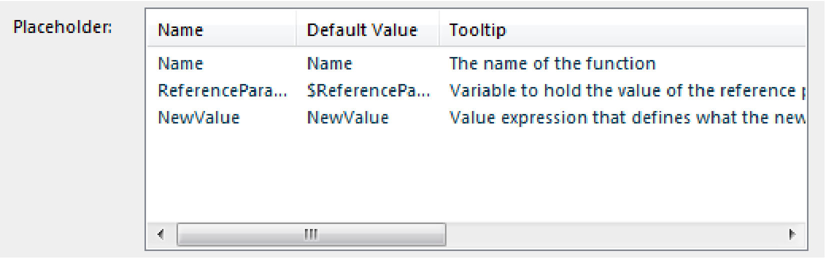
- The **search button** lets you search for anything in the code box and every found instance gets highlighted.

- I think i dont need to explain the ***Undo***, ***Redo***, ***Save*** and ***Save As-*** Buttons; with the ***New*** - button you can clear the snippet window in order to create a new snippet.

4.4 Placeholder?

You can replace parts of the snippet code (e.g. function names, variables or anything that you probably would change when you use that code again) with a placeholder. This placeholder will be highlighted when you paste the snippet to your script, and you can simply put in a new value for the placeholder.

- A standard placeholder consist of a ***Name***, a ***Default Value*** and an *optional* ***ToolTip***



Placeholder:

Name	Default Value	Tooltip
Name	Name	The name of the function
ReferencePara...	$ReferencePa...	Variable to hold the value of the reference ı
NewValue	NewValue	Value expression that defines what the new

   - The ***Name*** is what identifies the placeholder, it will be displayed in your snippet code as *$myPlaceholderName$*. It has no other purpose.
     ! A placeholder will later only be recognized, if its enclosed by "$"-chars in the snippet code.

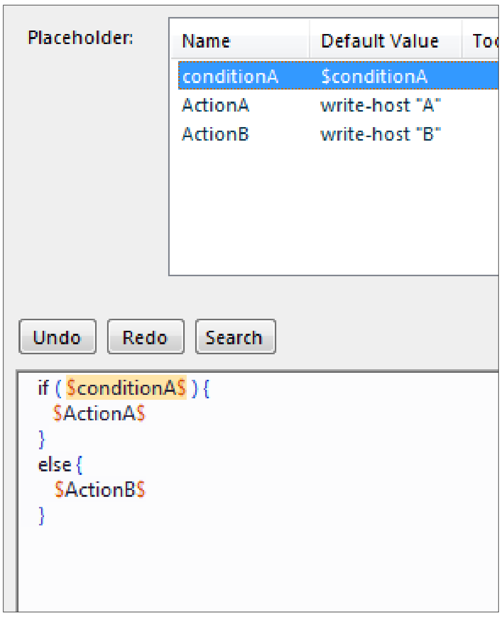
   - When you insert a snippet in your script, the ***Default Value*** will be put in every place where you put $<myPlaceholderName>$ in the snippet code. You can then change one instance and this will be replicated to **every instance**, or you can leave the Default Value. Therefore this should be a valid value.

   - The ***ToolTip*** will appear in PowerGUI and is optional.

   - If you **singleclick** on a placeholder *name* or *default value* in the **placeholder list**, all instances of it will be highlighted in the code.

   - If you **doubleclick** on a placeholder *name*, *default value* or *tooltip* in the **placeholder list**, you can edit it. If you edit the *name*, every instance of the placeholder in the snippet will be changed accordingly.

Example of a placeholder:

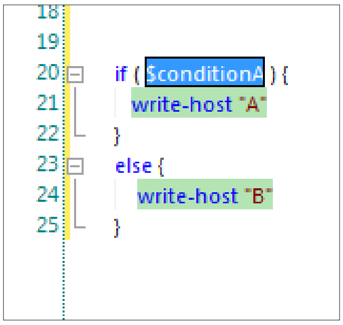


Placeholder:

Name	Default Value	To
conditionA	$conditionA	
ActionA	write-host "A"	
ActionB	write-host "B"	

```
if ( $conditionA$ ) {
  $ActionA$
}
else {
  $ActionB$
}
```

and how it appears in PowerGUI:

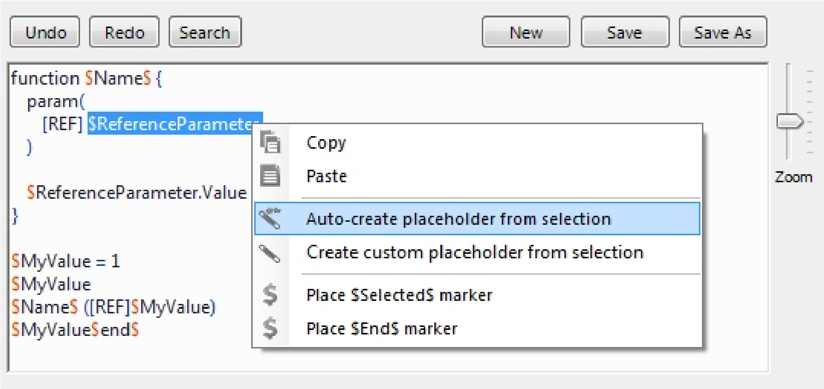


```
if ( $conditionA ) {
   write-host "A"
}
else {
   write-host "B"
}
```

4.5 Create a placeholder

There are **three ways** to create a placeholder:

- **The first two** you can access from **within the code box**. Just highlight a command or variable that you want to turn into a placeholder, right click and select "*Auto-create placeholder*" or "*Create custom placeholder*".



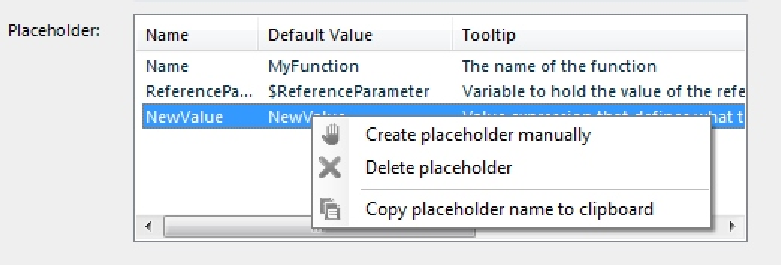
```
function $Name$ {
   param(
      [REF] $ReferenceParameter
   )

   $ReferenceParameter.Value
}

$MyValue = 1
$MyValue
$Name$ ([REF]$MyValue)
$MyValue$end$
```

  - "**Auto-create ...**" will turn the selection into a placeholder name and default value and will automatically replace **all** occurrences of it with **$**<*YourSelelection*>**$**.
  - "**Create Custom ...**" works basically the same, but you can first set a different name for the placeholder.
  - You can turn literally any part of the code into a placeholder. Only for the name itself, tabs, spaces and line breaks will be removed, so that its easier to read. Be careful: because you can turn **anything** into a placeholder, the auto-replacement will replace **any** occurrence of it, even inside another word. So i suggest not to turn e.g. a single character into a placeholder.
  - Every auto created instance will be highlighted for you to **review them**. I suggest to do that.

- **The third way** is to create a placeholder **manually**.



Placeholder:

Name	Default Value	Tooltip
Name	MyFunction	The name of the function
ReferencePa...	$ReferenceParameter	Variable to hold the value of the refe
NewValue	NewV	hat t

Create placeholder manually
Delete placeholder
Copy placeholder name to clipboard

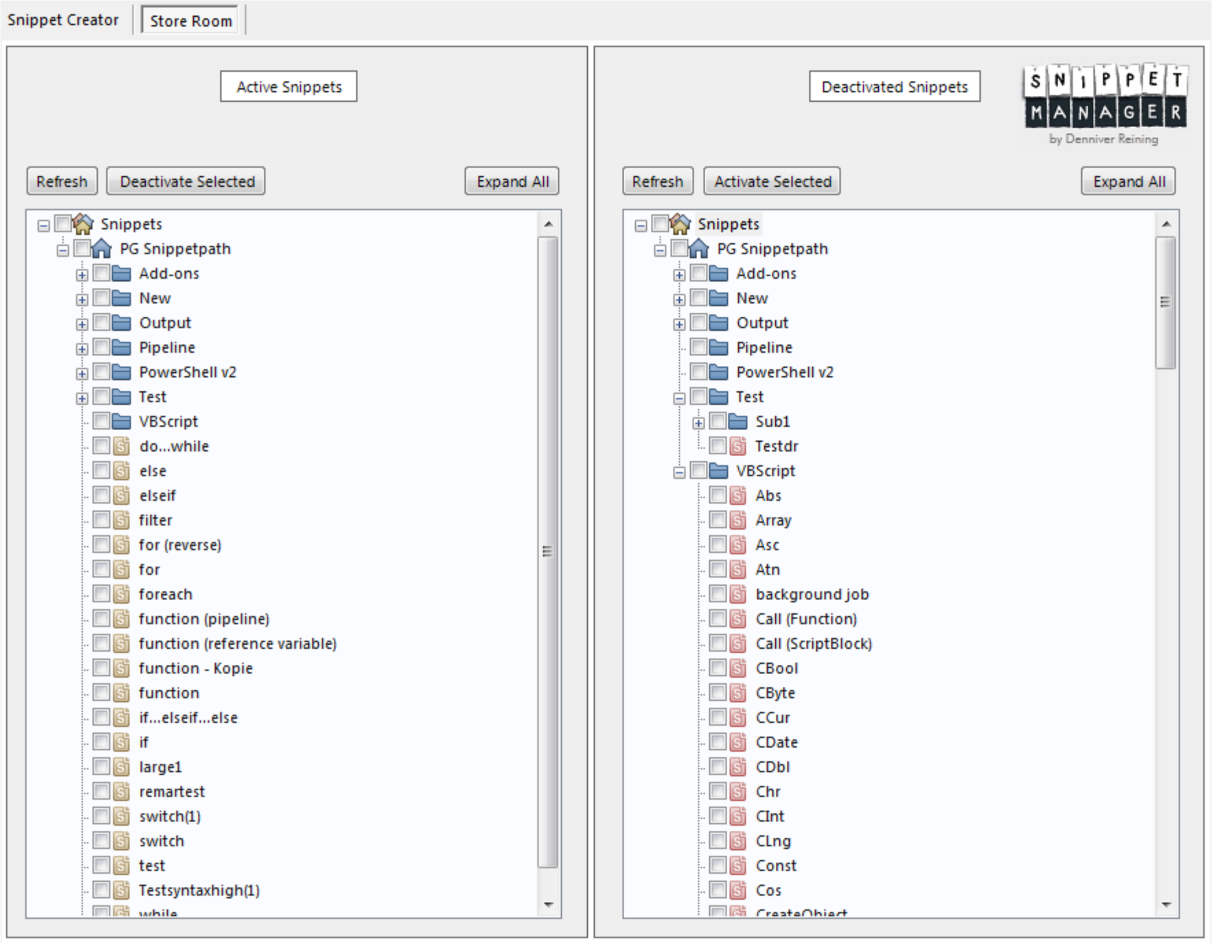
1. To do that, **right click** in the **placeholder list** and choose "*Create placeholder manually*".
2. **Insert** a name and a default value for your new placeholder.
3. The name of the placeholder has now been copied to the clipboard (along with the $...$ identifiers), so now simply **paste the new placeholder** to every point in the code where you want it to appear later.
   - You can simply recall the $...$-name of every placeholder by right-clicking on a placeholder and select "**Copy...to clipboard**", for easy pasting it to the snippet.

4.6 Markers

There are two special placeholders: **END** and **$SELECTED$** which you can **insert from the context menu** of the code box.
(And although you could just type them in, i suggest you do it via the context menu, because in this case Snippet Manager automatically **checks** if you already have the chosen marker somewhere in the code and if yes, **highlights** it for you.)

- The **END** marker marks the place where your cursor sits after you inserted the snippet to your script.
- When you select something (a string, a command, whatever) in your script and then insert the snippet, the **$SELECTED$** marker will be replaced by the selected text. In other words, the snippet will be "wrapped around" your selection.

5. The Store Room (activate and deactivate snippets)



In the Store Room you can **deactivate (and reactivate) snippets** you don't need at the moment, so that they don't take up space in (the rather small) *"Insert Snippet"*-menu in PowerGUI.

- If there are e.g. shipped snippets that you probably never use but you don't want to delete them, or you have sets of snippets for different projects, just **check** them (separately or the whole folder) **and click** "Deactivate Selected" on the left side.
- To **reactivate** snippets, just do the same on the right side and click "Activate Selected".

- If you deactivate **all snippets in a folder**, the folder itself will **still show up** everywhere in Snippet Manager, but in PowerGUI it won't.
  This is because **deactivated snippets stay where they are**, they will just have a different file extension so that PowerGUI ignores them. So don't be surprised, if you try to delete a seemingly empty folder and Snippet Manager warns you that there are deactivated snippets in it.

6. Addendum

- If you have any **suggestions, questions, problems, feature ideas** please send me an email to snipman@gmx.net.

Have a nice day,
Denniver Reining