Go

- Home
- About Me

Posts Comments

- Uncategorized

← More Granular Options for CWDIllegalInDllSearch Needed
Achieve Radically Better Firewire 800 Performance on Windows 7/2008/Vista x64 →

Copy and Paste with Clipboard from PowerShell

September 3, 2010 6 Comments

Many times I want to use PowerShell to manipulate some text that I have in something else like email or a text editor or some such or conversely, I do something at the command line and I want to paste the output into a document of some kind.

I'm not sure why there is no cmdlet for outputting to clipboard.

```
PS> get-command out-*

CommandType     Name            Definition
-----------     ----            ----------
Cmdlet          Out-Default     Out-Default [-InputObject <PSObject>] [-Verbose]...
Cmdlet          Out-File        Out-File [-FilePath] <String> [[-Encoding] <Stri...
Cmdlet          Out-GridView    Out-GridView [-InputObject <PSObject>] [-Title <...
Cmdlet          Out-Host        Out-Host [-Paging] [-InputObject <PSObject>] [-V...
Cmdlet          Out-Null        Out-Null [-InputObject <PSObject>] [-Verbose] [-...
Cmdlet          Out-Printer     Out-Printer [[-Name] <String>] [-InputObject <PS...
Cmdlet          Out-String      Out-String [-Stream] [-Width <Int32>] [-InputObj...
```

Fortunately, though Windows ships with a native clip.exe for piping text to clipboard. You can alias it to out-clipboard or just pipe to "clip".

```
new-alias  Out-Clipboard $env:SystemRoot\system32\clip.exe
```

Now you can just pipe to Out-Clipboard or clip just like the other Out-* cmdlets.

The larger issue is that PowerShell doesn't expose a way to paste text from the clipboard. However, the WinForms API of the .NET Framework has a GetText() static method on the System.Windows.Forms.Clipboard class. There is a catch, though:

> The Clipboard class can only be used in threads set to single thread apartment (STA) mode. To use this class, ensure that your Main method is marked with the STAThreadAttribute attribute.

This is unfortunate because PowerShell runs in a multithreaded apartment by default which means that the workaround is to spin up a new PowerShell process which is slow.

```
function Get-ClipboardText()
{
    $command =
    {
        add-type -an system.windows.forms
        [System.Windows.Forms.Clipboard]::GetText()
    }
    powershell -sta -noprofile -command $command
}
```

A slightly less obvious approach is to paste the text from the clipboard into a non-visual instance of the System.Windows.Forms.Textbox control and then emit the text of the control to the console. This works fine in a standard PowerShell process and is substantially faster than spinning up a new process.

```
function Get-ClipboardText()
{
    Add-Type -AssemblyName System.Windows.Forms
    $tb = New-Object System.Windows.Forms.TextBox
    $tb.Multiline = $true
    $tb.Paste()
    $tb.Text
}
```

The TextBox method is about 500 times faster than the sub-process method. On my system, it takes 0.501 seconds to do spin up the sub-process and emit the text from the clipboard versus 0.001 seconds to use the TextBox control in the current process.

I also alias this function to “paste”. The commands I actually type and remember are “clip” and “paste”.

When using paste, I generally am setting a variable or inserting my pasted text into a pipeline within parens.

CAVEAT: Whenever you capture a value without any {CR}{LF}, the clipboard will add it. You may need to trim the newline to get the behavior you expect when pasting into a pipeline.

Here’s some one-liners to illustrate. I have dig from BIND in my path and I have Select-String aliased to grep. Notice how when I paste the IP address I captured to the clipboard from the one-liner gets pasted back with an extra blank line.

```
PS> ((dig google.com) | grep '^google\.com') -match '(\d+\.\d+\.\d+\.\d+)$' | out-null; $matches[0]
72.14.235.104
PS> ((dig google.com) | grep '^google\.com') -match '(\d+\.\d+\.\d+\.\d+)$' | out-null; $matches[0] | clip
PS> paste
72.14.235.104

PS> ping (paste)
Ping request could not find host 72.14.235.104
. Please check the name and try again.
PS> ping (paste).trim()

Pinging 72.14.235.104 with 32 bytes of data:
Reply from 72.14.235.104: bytes=32 time=192ms TTL=51
Reply from 72.14.235.104: bytes=32 time=208ms TTL=51
Reply from 72.14.235.104: bytes=32 time=212ms TTL=51
Reply from 72.14.235.104: bytes=32 time=210ms TTL=51

Ping statistics for 72.14.235.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 192ms, Maximum = 212ms, Average = 205ms
```

Lust zu Chatten??
vor 1 Minute

Hi!
Mir ist langweilig...
Bist Du auch Single?
Bock auf ein Date?
...!!!?
Heut schon was vor?

Share this: Twitter 1 0

Like this: Like Be the first to like this post.

Filed under Uncategorized Tagged with powershell, utilities

6 Responses to *Copy and Paste with Clipboard from PowerShell*

1. Pingback: .debug » Goo.gl + PowerShell

2. *Dave Regal* says:
March 7, 2011 at 10:08 pm

Hey, Thanks for sharing what you found. I found a way to stay within the realms of Powershell. I need to enhance it so it takes the text as an input variable, but here’s what I got so far.

```
##########################################################################################
# Copy-Text.ps1
#
# The script copies a string variable to system clipboard.
# Credit: http://blogs.msdn.com/b/powershell/archive/2009/01/12/copy-console-screen-to-system-clipboard.aspx

$text = “This is my text”

# Load System.Windows.Forms assembly.
$null = [Reflection.Assembly]::LoadWithPartialName(“System.Windows.Forms”)

# Create data object.
```

```
$dataObject = New-Object windows.forms.dataobject

# Add generated strings to data object.
$dataObject.SetData([Windows.Forms.DataFormats]::UnicodeText, $true, $text)

# Put data object in system clipboard.
[Windows.Forms.Clipboard]::SetDataObject($dataObject, $true)

'The text has been copied to system clipboard.'
'You can now paste it to any application that supports text.'
```

Reply

3. *Dave Regal* says:
   March 7, 2011 at 10:09 pm

   Oh yeah, you have to run it differently.

   > powershell -noprofile -sta -command .\Copy-Text.ps1

   Reply
4. *Justin Dearing* says:
   March 16, 2011 at 3:04 pm

   Thanks for the alias and the script!!!

   I opened up a connect bug to add native cmdlets and a PSDrive for accessing the clipboard. If you think that would be a good idea you can vote on the issue: https://connect.microsoft.com/PowerShell/feedback/details/651777/implement-cmdlets-to-get-data-in-and-out-of-the-windows-clipboard

   Reply
5. Pingback: How to copy files through a remote desktop connection using clipboard and PowerShell « Second Life of a Hungarian SharePoint Geek
6. *Stephen* says:
   November 29, 2011 at 8:21 pm

   This is a very useful powershell feature, and your usage of Dig and Grep just blew my mind a bit. It suddenly stopped looking like a PoSh command line and more like Bash.

   Reply

Leave a Reply

Enter your comment here...

Recent Posts

- Delete Unwanted Windows Apps from Launch Pad with VMWare Fusion
- Size an MKMapView to Fit its Annotations in iOS Without Futzing with Coordinate Systems
- Targeted Marketing Considered Harmful
- How to Create an Xcode 4.0-style Window-based Application in Xcode 4.2
- "Bins" for Windows 7 is *Bleeping* Awesome
- How to Infuriate Your Custom Software Client in Three Easy Steps
- Windows 8 Metro + Desktop: Give it a Chance
- SUA Deprecated in Windows 8
- A Better Telnet for Windows
- Let's Play Spot the Scam

Top Posts

- How to Create an Xcode 4.0-style Window-based Application in Xcode 4.2
- Mount SkyDrive Natively in Windows
- Pin Eclipse Helios to Windows 7 Taskbar
- FIX: VirtualBox Host-Only Network Adapter Creates a Virtual "Public Network" Connection That Causes Windows to Disable Services
- Cool PowerShell Script Replicates Telnet
- Copy and Paste with Clipboard from PowerShell

- Custom JsonResult Class for ASP.Net MVC to Avoid MaxJsonLength Exceeded Exception
- Size an MKMapView to Fit its Annotations in iOS Without Futzing with Coordinate Systems
- SUA Deprecated in Windows 8
- Upgrade and Zenburn the Console Window

Tags

.net adobe acrobat android applocker asp.net asp.net mvc boot camp boot camp 3.1 bug c# chrome compatibility cryptography debugging development email ethics extension google google chrome h.264 internet internet explorer iOS iphone java javascript macbook microsoft nexus one performance powershell privacy security testing troubleshooting utilities virtualbox virtual machine virtualpc visual studio visual studio 2010 windows windows 7 windows vista

Archives

- April 2012 (1)
- March 2012 (2)
- November 2011 (1)
- October 2011 (2)
- September 2011 (4)
- August 2011 (1)
- June 2011 (3)
- May 2011 (7)
- April 2011 (2)
- March 2011 (2)
- February 2011 (9)
- January 2011 (6)
- December 2010 (6)
- November 2010 (2)
- September 2010 (9)
- August 2010 (24)
- July 2010 (12)
- June 2010 (1)
- March 2010 (8)
- February 2010 (4)
- January 2010 (12)
- December 2009 (5)
- November 2009 (2)