# TEC US 2011: PowerShell Deep Dive: Sean Kearney – Integrating PowerShell with Legacy Environments

These are the scripts Sean was using. You can also find his:

- Slides here: http://dmitrysotnikov.wordpress.com/2011/08/18/slides-from-deep-dive-sessions/
- Video recording here: http://dmitrysotnikov.wordpress.com/2011/08/11/deep-dive-video-integrating-powershell-with-legacy-environments-sean-kearney/

## close-file-function.ps1

```
function global:close-file ($name) {
	$DATA=(& '.\Handle Program\handle.exe' $name)
	Foreach ($FILE in 5..(($DATA.Count)-1))
	{
		$MYPID=$DATA[$FILE].substring(24,7).trim()
		$MYHANDLE=$DATA[$FILE].Substring(41,14).trim()
		(& '.\Handle Program\handle.exe' -c $MYHANDLE -p $MYPID -y)
	}
}
```

## get-dirsize.ps1

```
$size=[int]((cmd.exe1 /c 'dir c:\users\administrator /s | find "File(s)"'
)[($_.Count)-1].substring(24).trim()).split(" ")[0]
If ($Size) {Return $Size}
else { Exit 42 }
```

## get-proc.ps1

```
################# Check-PrivateBuild
### Script Copyright Stephen Croft
### Written for the 2011 Scripting Games Beginner challange 1
###
### Instructions;
### check-privatebuild takes four possible paramaters;
### -app "c:\windows\system32\calc.exe" --> Full path of application to check
### -complist computers.txt --> text file containing multiple computer names,
each on a single line, to check
### -computer DC01 --> name of remote (or local) computer to check
### -outputcsv --> name of output csv location
###
### Examples
###
```

```
### check-privatebuild
### checks for privatebuild flag on Notepad.exe on local machine, outputs to
%temp%\output.csv
###
### check-privatebuild -computer DC01
### checks for privatebuild flag on Notepad.exe on DC01, outputs to
%temp%\output.csv
###
### check-privatebuild -complist "c:\scripts\computers.txt" -outputcsv
"c:\scripts\output.csv"
### checks for privatebuild flag on computers listed in
c:\scripts\computers.txt, outputs to c:\scripts\output.csv
###
### check-privatebuild -app "c:\windows\system32\calc.exe"
### checks for privatebuild flag on calc.exe on localhost, outputs to
%temp%\output.csv
###
### please note that specifing both a single computer name and a list file
will only return the list file computers.
################################################################
### Version History
### v1.0.0 - Initial Version
### v1.0.1 - Amended function name and added version history.
### v1.0.2 - Removed Company info in Copyright
### v1.0.3 - Added advanced param handling/defauls, replaced FSO text file
handing with clear-content and add-content
### Added examples in instructions and renamed function (again) as I didnt
like what it was called o.O
### v1.0.3.1 - Code Cleanup (removed old FSO remarked line)
################################################################
########################
#Pre-Function gubbings (Param handling, file opening)
########################
Param(
[string]$app = "c:\Windows\System32\notepad.exe",
$complist,
[string]$computer = "localhost",
[string]$outputcsv = $env:temp + "\output.csv")
$app = $app -replace ":", "$" #replace the : with # so it works no matter the
target machine
clear-content $outputcsv
########################
#Define Functions
########################
function get-IsPrivateBuild
{param ($computername, $apppath)
#Cleanly get Hostname if localhost
if ($computername -eq "localhost")
{$hostname = $env:computername}
elseif ($computername -eq $null)
{$hostname = $env:computername}
else
{$hostname = $computername}
#Open $app properties and select the two we care about (IsPrivateBuild,
FileDescription)
$pb = (Get-ChildItem $apppath).VersionInfo | Select-Object IsPrivateBuild,
FileDescription
```

```
#amend csv variable with
Add-Content $outputcsv ($pb.FileDescription + "," + $hostname + "," +
$pb.IsPrivateBuild)
}
########################
#End Functions
########################
########################
#Core Script - the magic happens within!
########################
#start writing csv- this should run once to put CSV headers in place
Add-Content $outputcsv "Process, computer, PrivateBuild"
#start calling get-privatebuild with correct pass-thru variables
if($complist -eq $null) #checks for lack of import file
{
if ($computer -eq $null) #checks to see if any computername passed - if not
works on localhost
{$fullapp = "\\localhost" + "\" + $app
get-IsPrivateBuild $computer $fullapp}
else #uses passed $computer to check
{$fullapp = "\\" + $computer + "\" + $app
get-IsPrivateBuild $computer $fullapp
}
}
else #parses import file and runs get-privatebuild on each object
{get-Content $complist | foreach-Object {
$fullapp = "\\" + $_ + "\" + $app
get-IsPrivateBuild $_ $fullapp
}}

################ Check-PrivateBuild
### Script Copyright Stephen Croft
### Written for the 2011 Scripting Games Beginner challange 1
###
### Instructions;
### check-privatebuild takes four possible paramaters;
### -app "c:\windows\system32\calc.exe" --> Full path of application to check
### -complist computers.txt --> text file containing multiple computer names,
each on a single line, to check
### -computer DC01 --> name of remote (or local) computer to check
### -outputcsv --> name of output csv location
###
### Examples
###
### check-privatebuild
### checks for privatebuild flag on Notepad.exe on local machine, outputs to
%temp%\output.csv
###
### check-privatebuild -computer DC01
### checks for privatebuild flag on Notepad.exe on DC01, outputs to
%temp%\output.csv
###
### check-privatebuild -complist "c:\scripts\computers.txt" -outputcsv
"c:\scripts\output.csv"
### checks for privatebuild flag on computers listed in
c:\scripts\computers.txt, outputs to c:\scripts\output.csv
###
```

```
### check-privatebuild -app "c:\windows\system32\calc.exe"
### checks for privatebuild flag on calc.exe on localhost, outputs to
%temp%\output.csv
###
### please note that specifing both a single computer name and a list file
will only return the list file computers.
#################################################################
### Version History
### v1.0.0 - Initial Version
### v1.0.1 - Amended function name and added version history.
### v1.0.2 - Removed Company info in Copyright
### v1.0.3 - Added advanced param handling/defauls, replaced FSO text file
handing with clear-content and add-content
### Added examples in instructions and renamed function (again) as I didnt
like what it was called o.O
### v1.0.3.1 - Code Cleanup (removed old FSO remarked line)
#################################################################
########################
#Pre-Function gubbings (Param handling, file opening)
########################
Param($thisapp = 'c:\Windows\System32\notepad.exe', $complist,
[string]$computer = "localhost", [string]$outputcsv = $env:temp +
"\output.csv")
$app = $app -replace ":", "$" #replace the : with # so it works no matter the
target machine
clear-content $outputcsv
########################
#Define Functions
########################
function get-IsPrivateBuild
{param ($computername, $apppath)
#Cleanly get Hostname if localhost
if ($computername -eq "localhost")
{$hostname = $env:computername}
elseif ($computername -eq $null)
{$hostname = $env:computername}
else
{$hostname = $computername}
#Open $app properties and select the two we care about (IsPrivateBuild,
FileDescription)
$pb = (Get-ChildItem $apppath).VersionInfo | Select-Object IsPrivateBuild,
FileDescription
#amend csv variable with
Add-Content $outputcsv ($pb.FileDescription + "," + $hostname + "," +
$pb.IsPrivateBuild)
}
########################
#End Functions
########################
########################
#Core Script - the magic happens within!
########################
#start writing csv- this should run once to put CSV headers in place
Add-Content $outputcsv "Process, computer, PrivateBuild"
#start calling get-privatebuild with correct pass-thru variables
if($complist -eq $null) #checks for lack of import file
{
```

```
if ($computer -eq $null) #checks to see if any computername passed - if not
works on localhost
{$fullapp = "\\localhost" + "\" + $app
get-IsPrivateBuild $computer $fullapp}
else #uses passed $computer to check
{$fullapp = "\\" + $computer + "\" + $app
get-IsPrivateBuild $computer $fullapp
}
}
else #parses import file and runs get-privatebuild on each object
{get-Content $complist | foreach-Object {
$fullapp = "\\" + $_ + "\" + $app
get-IsPrivateBuild $_ $fullapp
}}
```

## handledemo.ps1

```
.\stupidseandiditagain.ps1
& '.\Handle Program\handle.exe' docx
& '.\Handle Program\handle.exe' -c xxx -p xxx
$DATA=(& '.\Handle Program\handle.exe' docx)
$DATA
$DATA | GET-MEMBER

$MYPID[0]
$MYPID[1]
$MYPID[2]
$MYPID[3]
$MYPID[4]
$MYPID[5]
$MYPID=($DATA[5].substring(24,7)
$MYPID=($DATA[5].substring(24,7).trim()
$MYHANDLE=($DATA[5].Substring(41,14)).trim()

(& '.\Handle Program\handle.exe' -c $MYPID -p $MYHANDLE -y)

Demo lauching first

./close-file-function.ps1

close-file docx
close-file xlsx
```

## launchcmd.ps1

```
CLEAR-HOST
WRITE-HOST
WRITE-HOST
WRITE-HOST '$value1="Hello Kitty"'
WRITE-HOST
```

```
WRITE-HOST '$value2=[datetime]'7/1/2011''
WRITE-HOST
WRITE-HOST '$value3=GET-CREDENTIAL -Credential CONTOSO\Mister.E'
WRITE-HOST
WRITE-HOST 'cmd.exe /c C:\TEC2011\testparam.cmd $value1 $value2 $value3'
WRITE-HOST
WRITE-HOST 'WRITE-HOST "The Exit Code returned from the CMD Script was
$LASTEXITCODE"'
WRITE-HOST
$CONTINUE=READ-HOST

$value1='Hello Kitty'

$value2=[datetime]'7/1/2011'

$value3=GET-CREDENTIAL -Credential CONTOSO\Mister.E

cmd.exe /c C:\TEC2011\testparam.cmd $value1 $value2 $value3

WRITE-HOST The Exit Code returned from the CMD Script was $LASTEXITCODE

$CONTINUE=READ-HOST

CLEAR-HOST

WRITE-HOST
WRITE-HOST
WRITE-HOST '$value1=("Hello Kitty").tostring()'
WRITE-HOST
WRITE-HOST '$value2=([datetime]'7/1/2011').toshortdatestring()'
WRITE-HOST
WRITE-HOST '$value3=(GET-CREDENTIAL -Credential CONTOSO\Mister.E).Username'
WRITE-HOST
WRITE-HOST 'cmd.exe /c C:\TEC2011\testparam.cmd $value1 $value2 $value3'
WRITE-HOST
WRITE-HOST 'WRITE-HOST "The Exit Code returned from the CMD Script was
$LASTEXITCODE"'

$CONTINUE=READ-HOST

$value1=('Hello Kitty').tostring()
$value2=([datetime]'7/1/2011').toshortdatestring()
$value3=(GET-CREDENTIAL -Credential CONTOSO\Mister.E).Username

cmd.exe /c C:\TEC2011\testparam.cmd $value1 $value2 $value3

WRITE-HOST The Exit Code returned from the CMD Script was $LASTEXITCODE
```

## launchcmd.vbs

```
dim objShell

Set objShell = WScript.CreateObject( "WScript.Shell" )
```

```
dim test1

dim test2


test1=objShell.Exec("cmd.exe")

test2=objshell.Exec("powershell.exe -command dog")

wscript.echo test1

wscript.echo test2
```

## LaunchPS.CMD

```
@echo off

cls

REM LaunchPS.BAT

Echo.

Echo SET Param1='Snoopy Dance'

Echo.

Echo SET Param2=4/1/2011

Echo.

Echo SET Param3=1

Echo.

Echo Powershell.exe -executionpolicy RemoteSigned -file c:\tec2011\TestParam.PS1 %%Param1%%
%%Param2%% %%Param3%%

pause

SET Param1="Snoopy Dance"

Echo.

SET Param2=4/1/2011

Echo.
```

```
SET Param3=1

Echo.

Powershell.exe -executionpolicy RemoteSigned -file c:\TEC2011\TestParam.PS1 %Param1% %Param2%
%Param3%

Echo.

Echo The Result code from the script is %ERRORLEVEL%

Echo.

pause

@echo off

cls

REM LaunchPS.BAT

Echo.

Echo SET Param1='Snoopy Dance'

Echo.

Echo SET Param2=4/1/2011

Echo.

Echo SET Param3=1

Echo.

Echo Powershell.exe -executionpolicy RemoteSigned -file c:\tec2011\TestParamType.PS1 %%Param1%%
%%Param2%% %%Param3%%

pause

SET Param1="Snoopy Dance"

Echo.

SET Param2=4/1/2011

Echo.

SET Param3=1
```

Echo.

Powershell.exe -executionpolicy RemoteSigned -file c:\TEC2011\TestParamType.PS1 %Param1% %Param2% %Param3%

Echo.

Echo The Result code from the script is %ERRORLEVEL%

Echo.

## listapps.ps1

```
$list=get-childitem
REGISTRY::HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\

Foreach ($App in $list)
{

(get-itemproperty registry::$App).DisplayName

}
```

## listpc.ps1

```
$list=net sessions
$Systemcount=($list.count)
foreach ($counter in
4..$systemcount){$IP=$list[$counter].substring(2,21).trim();$user=$list[$counter].substring(23,43).trim();$computername=(nbtstat -A
$ip)[14].substring(4,15).trim();Write-host $user,$computername}
```

## locationofdevices.txt

HKLM\System\CurrentControlSet\Enum

Level1 - Device Category

## shownonpresentdevices.txt

set devmgr_show_nonpresent_devices=1

## stupidseandiditagain.ps1

```
INVOKE-ITEM 'C:\tec2011\If Sean Keeps doing this we won.docx'
```

```
INVOKE-ITEM 'C:\tec2011\demodocbadsean.docx'
INVOKE-ITEM 'C:\tec2011\Why does Sean always leave his files open.docx'
INVOKE-ITEM 'C:\tec2011\FakeSheetforSean.xlsx'
INVOKE-ITEM 'C:\tec2011\StupidSean.xlsx'
```

## testparam.cmd

```
@echo off

ECHO ECHO %%1

ECHO.

ECHO ECHO %%2

ECHO.

ECHO ECHO %%3

ECHO.

ECHO EXIT 1001001

ECHO.

pause

ECHO.

ECHO %1

ECHO.

ECHO %2

ECHO.

ECHO %3

ECHO.

EXIT 1001001
```

## testparam.ps1

```
WRITE-HOST '$val0=($args[0])'
WRITE-HOST
WRITE-HOST '$val0.gettype()'
WRITE-HOST 'WRITE-HOST $val0'
WRITE-HOST
WRITE-HOST '$val1=($args[1])'
```

```
WRITE-HOST
WRITE-HOST '$val1.gettype()'
WRITE-HOST 'WRITE-HOST $val1'
WRITE-HOST
WRITE-HOST '$val2=($args[2])'
WRITE-HOST
WRITE-HOST '$val2.gettype()'
WRITE-HOST 'WRITE-HOST $val2'
WRITE-HOST
WRITE-HOST 'EXIT 42'

$CONTINUE=READ-HOST

$val0=($args[0])
WRITE-HOST
$val0.gettype()
WRITE-HOST $val0

$val1=($args[1])
WRITE-HOST
$val1.gettype()
WRITE-HOST $val1

$val2=($args[2])
WRITE-HOST
$val2.gettype()
WRITE-HOST $val2
WRITE-HOST
EXIT 42
```

## testparam.vbs

```
Value0=wscript.arguments(0)

Value1=wscript.arguments(1)

Value2=wscript.arguments(2)

wscript.Echo Value0

wscript.Echo Value1

wscript.Echo Value2

wscript.quit 2112
```

## testparam1.cmd

```
WRITE-HOST '$val0=($args[0])'

WRITE-HOST '$val0.gettype()'
```

```
WRITE-HOST 'WRITE-HOST $val0'
WRITE-HOST
WRITE-HOST '$val1=($args[1])'
WRITE-HOST '$val1.gettype()'
WRITE-HOST 'WRITE-HOST $val1'
WRITE-HOST
WRITE-HOST '$val2=($args[2])'
WRITE-HOST '$val2.gettype()'
WRITE-HOST 'WRITE-HOST $val2'
$CONTINUE=READ-HOST
$val0=($args[0])

$val0.gettype()
WRITE-HOST $val0

$val1=($args[1])

$val1.gettype()
WRITE-HOST $val1

$val2=($args[2])

$val2.gettype()
WRITE-HOST $val2
$CONTINUE=READ-HOST
```

```
CLEAR-HOST
WRITE-HOST '$type0=[string]($args[0])'
WRITE-HOST '$type0.gettype()'
WRITE-HOST 'WRITE-HOST $type0'
WRITE-HOST
WRITE-HOST '$type1=[datetime]($args[1])'
WRITE-HOST '$type1.gettype()'
WRITE-HOST 'WRITE-HOST $type1'
WRITE-HOST
WRITE-HOST '$type2=[BOOLEAN]($args[2])'
WRITE-HOST '$type2.gettype()'
WRITE-HOST 'WRITE-HOST $type2'
WRITE-HOST
$CONTINUE=READ-HOST
$type0=[string]($args[0])
$type0.gettype()
WRITE-HOST $type0

$type1=[datetime]($args[1])
$type1.gettype()
WRITE-HOST $type1

$type2=[BOOLEAN]($args[2])
$type2.gettype()
WRITE-HOST $type2
```

$CONTINUE=READ-HOST

WRITE-HOST 'EXIT 42'

EXIT 42

## testparamtype.ps1

```
WRITE-HOST '$type0=[string]($args[0])'
WRITE-HOST
WRITE-HOST '$type0.gettype()'
WRITE-HOST 'WRITE-HOST $type0'
WRITE-HOST
WRITE-HOST '$type1=[datetime]($args[1])'
WRITE-HOST
WRITE-HOST '$type1.gettype()'
WRITE-HOST 'WRITE-HOST $type1'
WRITE-HOST
WRITE-HOST '$type2=[BOOLEAN]($args[2])'
WRITE-HOST
WRITE-HOST '$type2.gettype()'
WRITE-HOST 'WRITE-HOST $type2'
WRITE-HOST
WRITE-HOST 'EXIT 42'
$CONTINUE=READ-HOST
$type0=[string]($args[0])
WRITE-HOST
$type0.gettype()
WRITE-HOST $type0

$type1=[datetime]($args[1])
WRITE-HOST
$type1.gettype()
WRITE-HOST $type1

$type2=[BOOLEAN]($args[2])
WRITE-HOST
$type2.gettype()
WRITE-HOST $type2
$CONTINUE=READ-HOST
EXIT 42
```