The next generation command line scripting

Presented by
Bob McCoy, CISSP/ISSAP, MCSE
Microsoft Services
Cyber Security Forum 07/18/2007

Windows PowerShell

Command Line Interface

- cmd.exe and command.com
  - Lack of scriptable functionality
  - Lack of documentation
- Windows Script Host (WSH)
  - Not integrated with the shell
  - Vulnerable
- Cscript
  - Not integrated with the shell
- Administrative tasks and automation to dependent on proprietary GUI based tools
- Scriptable wrappers difficult to implement

Background

- Gottfried Leibniz's monadology (1704)
  - Universe is a composite of fundamental elements (cmdlets) integrated in a pre-established harmony
- In place of CMD and WSH
- Microsoft GUI wrappers
- Supported on XP, Server 2003, Vista and Longhorn Server (x86, x64, IA64)
- Foundation for Exchange Server 2007
- Built on .NET Framework

- Object Orientation
- .NET Based (*requires .NET 2.0*)
- CLI & Scripting Language (C# like, Perl like)
- Cmdlets
- Platform Independent Utilities
- Users
  - Administrators
  - Scripters
  - System integrators

Windows PowerShell

- Strong guidelines
- Aliasing
- Tab-completion & partial parameter usage
- Command line editing
- Pipelining
- Object utilities
- Better documentation
- Security

Windows PowerShell

Feature Highlights

- Cmdlets
  - Object oriented
  - Extensible
  - Easily discoverable
- Access to objects
  - ADO
  - .NET
  - WMI
  - COM
  - Etc…
- PSDrives

Typing

- Loose
  - $d="7/18/2007"
  - Function foo() {$args[0]}
- "Strong"
  - [datetime]$d="4/20/2005"
  - function foo([datetime]$date) {$date}
  - [int] [char] "a"
- Extensible
  - add-member $d Note Description "CSF Briefing"
  - $d.Description
- PowerShell works on any .NET type - not a fixed set of "scripting types"

- Basic Shell usage
  - Discovering PS commands and functionality
  - Get-Command
  - Get-Alias
  - Get-Help
  - Get-Member
- Other Useful cmdlets not in Demo
  - Select-Object
  - ForEach-Object
  - Where-Object
  - Sort-Object

DEMO – Basic Usage

- Access WMI data
- Use of alternate credentials
- Get-WmiObject win32_share
- Get-WmiObject win32_share | Format-List [a-z]*
- GWmi win32_share | Format-Table Name,Path -hide
- Binds to root\cimV2 namespace on the local host by default, params

Demo – WMI

- A data store location
- Access it just like a file system drive
- New-Psdrive
- Get-Psdrive
- Set-Location (aliased to cd)
- Help About_Provider

Demo - PSDrive

- Debug
  - Programmer-level information
- ErrorAction
  SilentlyContinue | *Stop* | *Continue* | *Inquire*
  - What to do if an error occurs?
- ErrorVariable *VariableName*
  - Assign errors to a variable
- Verbose
  - Additional information about the activities being performed

Ubiquitous Parameters

Interactivity

- Commands with side-effects support:
  - Whatif
    - gps | where {$_.handles –ge 500} | stop-process -WhatIf
  - Confirm
    - Stop-process S* -Confirm
  - Verbose
    - Stop-Process [a-x]*[q]*[r-t] -Verbose

Security – Execution Policy

- Get-ExecutionPolicy
- Set-ExecutionPolicy
  Restricted | *AllSigned* | *RemoteSigned* | *Unrestricted*
  - Policy can only be set by Admin
- Use of code signing
- Self-signing cert
  http://www.hanselman.com/blog/SigningPowerShellScripts.aspx
  - Root ► Personal Cert ► Trusted Publisher
- Does not implicitly execute scripts from current directory

```
"hello" | get-member

# SIG # Begin signature block
# MIIEMwYJKoZIhvcNAQcCoIIEJDCCBCACAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# gjcCAQSgWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUN0zmm7MbQLHr3Ay6MaRZ30DF
# ywCgggI9MIICOTCCAaagAwIBAgIQ/OlnVY3d169DGW1clWeFqjAJBgUrDgMCHQUA
# MCwxKjAoBgNVBAMTIVBvd2VyU2hlbGwgTG9jYWwgQ2VydGlmaWNhdGUgUm9vdDAe
# Fw0wNzA3MTYxMzM1NDZaFw0zOTEyMzEyMzU5NTlaMBoxGDAWBgNVBAMTD1Bvd2Vy
# U2hlbGwgVXNlcjCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAoMd3ZkVZ0cAc
# 4kB2z5zIq7ttKsb+nK1Zv+WnuoIL89aWlXzZqqrXU5sAbBrLu1aq84I7lnTR/LxH
# 0Ts3EwaxXXdzwX45u4Dnhnp7IR6pdoVaOnOaKu6SU68sQD5CV5HKbBzbRyxBuDxX
# 57QgWHhsLwHN05+mG3G41XMgfLLI0WcCAwEAAaN2MHQwEwYDVR0lBAwwCgYIKwYB
# BQUHAwMwXQYDVR0BBFYwVIAQwH7/oqgShF/e6q4m27AtX6EuMCwxKjAoBgNVBAMT
# IVBvd2VyU2hlbGwgTG9jYWwgQ2VydGlmaWNhdGUgUm9vdIIQCztDjdeS64tDd5vS
# 3g4NNTAJBgUrDgMCHQUAA4GBALdwm6EP3CoT4qjG5yNBvtNtO447duj7LlDjWBIo
# np5C3CpV3uxmFlOh2ejhQRa8kfqfe7t+S0dchQv1rTLf7m65TwZeRf5yLLvOaKI5
# myZIDymYGBLE2Hl9iqWm8IdiVDaVCRi/H2XB/BtbkkfS7iekce3tF/4SL/vNSRMW
# 1Tj6MYIBYDCCAVwCAQEwQDAsMSowKAYDVQQDEyFQb3dlclNoZWxsIExvY2FsIENl
# cnRpZmljYXRlIFJvb3QCEPzpZ1WN3devQxltXJVnhaowCQYFKw4DAhoFAKB4MBgG
# CisGAQQBgjcCAQwxCjAIoAKAAKECgAAwGQYJKoZIhvcNAQkDMQwGCisGAQQBgjcC
# AQQwHAYKKwYBBAGCNwIBCzEOMAwGCisGAQQBgjcCARUwIwYJKoZIhvcNAQkEMRYE
# FITgShAtGf7rHeEmHnBR40TpbGerMA0GCSqGSIb3DQEBAQUABIGAOiR6OP3ixsjE
# Lks/xqcSHeYwjXfgIODFUGUnzBJr8nm/tsFvbfGEkgWYu0M8MqCvt7W90vIQT1Mq
# iF8eEr+8e17mj0puGvForlwRR3aP9kAcPhVC946t87ymtx/+RLbmxbKM5/iJOagW
# EDkfVHuVHbXzCHe6ZS8j3HbjNV0HKLA=
# SIG # End signature block
```

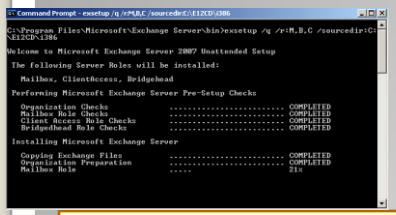
Code Signing Sample

Security – Passwords

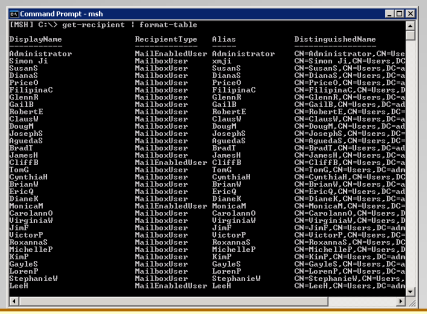
- Read-Host -assecurestring
- ConvertTo-SecureString
- ConvertFrom-SecureString
- Encryption
  - No Key – uses Triple-DES
  - With Key – uses Rijndael
  - Windows Data Protection (DPAPI) http://msdn2.microsoft.com/en-us/library/ms995355.aspx
  - http://www.leeholmes.com/blog/SecureStringsInPowerShell.aspx
  - get-help ConvertFrom-SecureString

E2007 Management Architecture

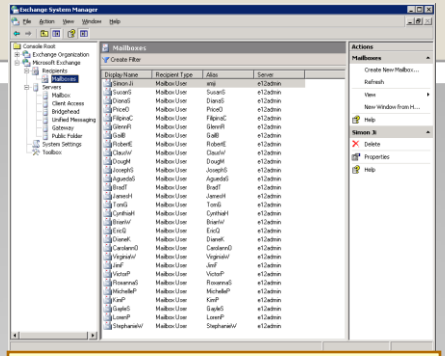
Setup

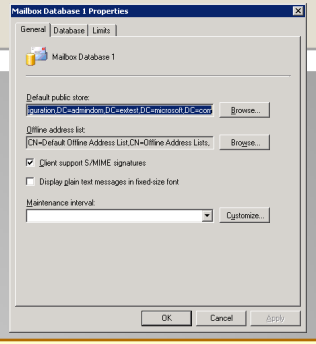


CLI



GUI





WinForms

ADO.Net

PS Provider

WinForms

Early-bound objs

PS Engine

Exchange cmdlets

Configuration Data Access

Process boundary

Wrap Up

- PowerShell is extremely powerful (and very cool)
- The price is right for Windows users
- Documentation is extensive
- Scripting experience much richer at both design and execution time
- Strong community support (blogs, repositories, books)

Resources

- Official PowerShell site
  http://www.microsoft.com/powershell
- TechNet Scripting Center
  http://www.microsoft.com/technet/scriptcenter/hubs/msh.mspx
- PowerShell Team Blog
  http://blogs.msdn.com/PowerShell/
- Sapien Press Blog
  http://blog.sapien.com/
- The PowerShell Guy
  http://thepowershellguy.com/
- Books
  “PowerShell Step-by-Step” (MSPress)
  “PowerShell TFM” (Sapien Press)
  “Windows PowerShell in Action” (Manning Pub)
  “Windows PowerShell Quick Reference” (O’Reilly)

Demo Script

Get-process

Ps | where { $_.handles –gt 500 }

Ps | where { $_.handles –gt 1000 } | sort ws –descending | select-object –first 5

Ps | gm

$g = ps notepad; $g.WaitForExit()

$g = ps notepad; $g.Kill()

Get-service

Get-service | where { $_.status –eq “running” }

Get-service | format-table –groupby status

Get-service | sort status | format-table –groupby status

Get-service | sort status,name | format-table –groupby status

---

WMI

Get-WMIObject Win32_Share

Get-WMIObject -list | where { $_.name –like “win32*” }

gwmi win32_share | format-table name,status,description -auto -hide

---

Regex

("foo" -match "o")

("foo" -match "^o")

("foo" -match "o$")

("foo" -match "[a-l]")

("foo" -match "[a-l].*o$")

("foo" -match "[a-l].*x$")

```
[regex]$rx = "[a-l].*o$"

"foo" -match $rx          "fox" -match $rx
```

Date and Time

```
$d = get-date

$d.dayofweek

$d.adddays(90)

$d.getdatetimeformats()

$d.touniversaltime().tolocaltime()

$bd = get-date -year 2007 -month 6 -day 25

$d - $bd

($d - $bd).days
```

Registry

```
$r = get-item 'HKLM:\software\Microsoft\Windows\CurrentVersion\Run'

$r.getvaluenames()

$r.getvalue("Zune Launcher”)

$reg = get-item “HKCU:\Software\Microsoft\Windows\CurrentVersion”

$r = $reg.OpenSubKey(“run”,$true)

$r.setvalue(“foo”,”bar”)
```