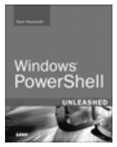
Introduction To: Microsoft Windows PowerShell



Created and Presented by:
Rand Morimoto, Ph.D., MCSE, CISSP
Tyson Kopczynski, MCSE, CISSP
Slides and Video of seminars up on
http://www.cco.com/online.htm

Presented by

Agenda

- Introduction
- What is PowerShell?
- Understanding PowerShell
- Scripting with PowerShell
- PowerShell Best Practices
- Questions and Answers

Presented by

Rand Morimoto
President,
Convergent Computing

- Lead Author for Sams Publishing "Unleashed"- series (books on Windows, Exchange, SharePoint, ISA, MOM)
- Series Editor for Sams Publishing "Administration and Management"-series (books on Vista, SQL 2005, OpsMgr 2007)
- Head Judge – Imagine Cup 2005, 2006, 2007, and now 2008! (Int'l IT competition for college students (www.imaginecup.com))

Tyson Kopczynski
Senior Consultant,
Convergent Computing

- Author of Windows PowerShell Unleashed
- Contributing author in a numbers of books
- NetworkWorld blogger
- Practice Lead – CCO Security Solutions
- Developed the CCO Scripting Practice
- Specialist in Active Directory, Group Policy, Windows scripting, Windows Rights Management Services, PKI, and IT security practices
- CISSP, GSEC, GCIH, and MCSE Security

Presented by

Convergent Computing (CCO)

- **Entering our 22nd year in business**
- **65-consultants (32 published authors)**
- **Work with technology products 2-3 yrs before the products release**
- **Design planning, migration and implementation assistance, knowledge transfer, and support**

CCO Publications
A few of the books we've written lately…

Presented by

What is Windows PowerShell?

Presented by

Windows PowerShell Overview…

- Microsoft's new command line shell and scripting language
- Built on the .NET Framework
- Easy to use, adopt, and learn
- Scripting language is Perl-ish and C#-ish
- Meant to be readable
- Designed from the ground up to simplify routine tasks without the overhead of the GUI…

Presented by

The future of PowerShell…

- Already has been adopted by a number of product groups
  - Exchange Server 2007
  - System Center Operations Manager 2007
  - System Center Data Protection Manager V2
  - System Center Virtual Machine Manager
  - *** Windows Server 2008 ***
- Many more product groups are working to adopt and integrate PowerShell into their applications

Presented by

Windows Management Dark Ages

The Frustrated Admin

A bunch of different scripting languages	A bunch of different command line tools	A bunch of different management GUIs

Windows and Server Products

WMI	COM	APIs	.NET	…

Task Completion

Presented by

Windows Management Renaissance

The Happy Admin

Still has the old interfaces	**PowerShell Console**	**PowerShell Driven MMC**	**PowerShell Scripts**

Windows and Server Products

WMI	COM	APIs	.NET	…

Task Completion

What is the difference?

Presented by

PowerShell is the Difference!

PowerShell will become the foundation for all Windows and Server product management…

MMC	Scripts	Console	GUIs

PowerShell

Common Interfaces	.NET Framework

Windows and Server Products

Presented by

Example – Exchange Server 2007

All aspects of management are driven by PowerShell!

Setup EMS EMC

Presented by

Understanding Windows PowerShell

Presented by

The PowerShell cmdlets

- Pronounced "command-let“
- The smallest unit of PowerShell functionality (aka like a command)
- Are always named in a verb-noun format
- PowerShell comes with 130 cmdlets (more can be “snapped in”)
- Are instances of .NET classes, not stand-alone executables
- Are just compiled DLL files that are loaded into the PowerShell process at startup

Presented by

Demos

One:

- Using the PowerShell console show the get-process cmdlet.
- Then show the get-command cmdlet. ***Note how many cmdlets there are.***

Two:

- Load the ESM, and again use the get-command cmdlet. Note how there are now more cmdlets.
- Run the get-pssnapin cmdlet and show the Exchange Management Shell snap-in.

Presented by

Other cmdlet Tidbits

- Typically, written using C# (but VB can be used)
- Parsing, error presentation, and output formatting are typically handled by the Windows PowerShell runtime (not a cmdlet)
- cmdlets process a single record at a time

Presented by

Executing cmdlets

- Verb-Noun
  - Verb-Noun –FirstP Val –SecondP Val, Val –ThirdP: Val
- You can alias
  - Set-Alias abc Verb-Noun
- Parameters/arguments can be positional
  - Verb-Noun Val
- Many arguments can be wild carded
  - Verb-Noun V*
- Partial parameter names allowed
  - Verb-Noun –P Val

Presented by

Demo

One:

get-process -name powerpnt

get-process -name powerpnt,powershell

get-process -name:powerpnt

Two:

gps

Three:

get-process powerpnt

Four:

get-process p*

Five:

get-process -n powerpnt

Presented by

What about existing interfaces?

- **You can still use them!**
- PowerShell is fully backward compatible
- Other command lines tools still work
- All other interfaces also still work
  - Windows Management Instrumentation (WMI)
  - Active Directory Services Interface (ADSI)
  - Microsoft .NET Framework and COM

Presented by

Demo

One:

```
ipconfig
```

Two:

```
get-wmiobject
    Win32_NetworkAdapterConfiguration
```

Three:

```
$User =
    [ADSI]"LDAP://CN=user1,OU=Accounts,DC=companyabc,DC=com"
```

Four

```
$IE = new-object -com
    InternetExplorer.Application
$IE.visible = $True
```

Five

```
$Ping = new-object
    Net.NetworkInformation.Ping
$Ping.send("host1")
```

Presented by

Getting Help

- Man-page style help on language and commands
  - Help
  - Help <command>
  - <command> -?
  - Help About_While
  - Get-Help * | Where {$_.Synopsis -match "process"}
- Finding Commands
  - Get-Command get*
  - Get-Command –Noun Process
  - Get-Command –Type {Alias | Function | Filter | Cmdlet | ExternalScript | Application | Script | All }

Presented by

Demo

One:

help get-process

help get-process -full

Two:

Help about_Escape_character

These are located at…

C:\Windows\System32\WindowsPowerShell\v1.0\en-US

Presented by

PowerShell Tab Completion

- Works with files and folders
- **But wait there is more!**
  - cmdlets
  - parameters
  - variable attributes

Presented by

Demo

Demo:

C:\progra[tab]

get-p[tab]

get-process –n[tab]

$ie.v[tab]

Note: The IE object should still present from past demo (do not close open console).

Presented by

Variables

- Names begin with $
- Can contain objects not just values
- Can contain a collection of objects
  - $users[0], $users[1], etc.
- Can be used directly from the command line
- Can be declared as an object type
  - [int]myinteger
- Keep variable names simple without special characters and spaces (- or _ is ok)
- ***Current pipeline object is the $_ variable***

Presented by

Demo

One (string):
$MyVar = "mystring"

Two (array):
$MyVar = "a","b","c","d"

Three (object collection):
$MyVar = get-process

Four:
get-service | where {$_.status -eq "Running" }

Aliases

- Used to simulate existing shell commands
  - Set-Location = cd
  - Get-Childitem = dir
- Use them to reduce typing
- You can create your own
  - Set-Alias
- **But watch out!**
  - Don't use cryptic aliases (you may forget)
  - The same may not be used by others

Demo

One:
get-alias

Two:
set-alias myalia get-item
myalia

PowerShell is Object Based!

- A PowerShell cmdlet returns an object collection
- Objects in these collections have:
  - Methods
  - Properties
- **Hurray!** - Traditional text parsing is now replaced with direct object manipulation
- Objects can be looked at, piped, modified, manipulated, and so on
- **Get-Member** - "Reflect" on an object to learn more about it

Demo

One:
get-service
Notice the default view!

Two:
get-service | gm
Notice the different attributes of the resulting object collection…

The Pipeline

- Just like any other shell pipeline
- **Except!** - it works with objects (not just text)
- All PowerShell commands execute within a pipeline
- Use the | to pipe information (text + objects) between commands
- All pipelines end with the Out-Default cmdlet
  - It selects a set of properties and their values
  - Then displays those values in a list or table
- **PowerShell also gives you a nice default view…**

Demo

One:

ipconfig | select-string “IPv4 Address”

Two:

get-process | where-object {$_.Company -match ".*Microsoft*"}

Presented by

Demo

Three:

get-process | where-object {$_.Company -match ".*Microsoft*"} | select Name, ID, Path

Four:

get-process | where-object {$_.Company -match ".*Microsoft*"} | select Name, ID, Path | convertto-html > report.html

Presented by

Objects in the Pipeline

What is so cool about objects in the pipeline?

You can create extremely powerful “one-liners”!

import-csv .\users.csv | foreach {Get-ADObject -filter "(&(samAccountName=$($_.employeeID)))"} | select @{name='Name';Expression={$_.cn} },@{ name='Accountname'; Expression={$_.sAMAccountName} },@{ name='Mail'; Expression={$_.mail}} | export-csv results.csv

Presented by

Demo (EMS)

Create Mailbox:

new-mailbox -alias tyson -name tyson -database "Mailbox Database" -org Users -UserPrincipalName tyson@companyabc.com

Move Mailbox:

move-mailbox tyson -targetdatabase “db2"

Move a whole bunch of mailboxes:

get-mailbox -server dc01 | move-mailbox -targetdatabase “db3“

Create a mailbox report:

get-mailbox | export-csv report.csv

Presented by

Providers

- Provides access/interface to a hierarchical stored data
- Treated as “Drives” in PowerShell
  - Filesystem, Registry, Alias, Certs, Env, Functions, Variables, etc.
- Drives and their items are navigated accessed just like the hard disk

Get-ChildItem	dir	ls
Get-Location	cd	pwd
Get-Content	type	cat
New-Item –type Directory	mkdir	mkdir
Set-Location	cd	cd

Presented by

Demo

get-psdrive
cd function:
dir
get-content elevate
cd c:\
dir
cd hkcu:
dir
get-itemproperty “Console”

Presented by

Extended Type System (ETS)

- **PowerShell is almost typeless!**
- Common interfaces for operating on pipeline objects independent of type (for example .NET, WMI, XML, ADO, ADSI etc)
  - Every object is a [psobject]
- Allows you to transparently extend objects
  - Aliases, Notes, Properties, Methods, PropertySets, etc..
- Type extensions can be defined via Types.ps1xml configuration files
  - **DO NOT MODIFY DEFAULT Types.ps1xml!**
- Or objects can be manipulated interactively using the Add-Member cmdlet

Presented by

Demo

One:

```
$ProcList = get-process "Power*"
$ProcList | add-member -Type scriptProperty "RunTime"
    {return ((date) - ($this.starttime))}
$Proclist | select Name, @{name='RunTime';
    Expression={"{0:n0}" -f $_.RunTime.TotalMinutes}}
```

Two:

```
$mystring = [string]”This is a string”
$mystring
$mystring.gettype()
```

Presented by

Demo

Three:

```
$myarray = [array]”This is an array”
$myarray
$myarray.gettype()
```

Presented by

Scopes

- **Global**
  - Created when the Windows PowerShell starts.
  - Visible from all child scopes.
- **Local**
  - Always the Current Scope.
  - A new local scope is created whenever you run a function, script, or start a new instance of the Windows PowerShell.
- **Script**
  - Created when script is started and removed when script is finished.
  - Only visible why script is running.
- **Private**
  - Is similar to a local scope.
  - With one key difference, definitions in the private scope aren’t inherited by any child scopes

Presented by

Demo

One:

```
function scope {
  $mymessage = “Hi!”
  write-host $mymessage
  write-host $mystring
  write-host “Done”
}

scope
$mymessage
```

Two:

```
function scope {
  $Global:mymessage = “Hi!”
  write-host $mymessage
  write-host $mystring
  write-host “Done”
}

scope
$mymessage
```

Presented by

Scripting Security

Shell Defaults

- Scripts do not run
- PowerShell files are associated with Notepad
- Must provide a path to a script file or executable in order to run it

You Can Configure

- ExecutionPolicy
  - Restricted
  - AllSigned
  - RemoteSigned
  - Unrestricted (No!!!)
- Centrally configurable via Group Policy

Presented by

Other Notables

- You can have your own startup profile:
  - <My Docs>\WindowsPowerShell\profile.ps1
- You can create command aliases
- “Pimp your Prompt”
- Many preferences are controllable via variables

Presented by convergent

Demo

One:

```
function prompt {
  return “Its PowerShell:"
 }
```

Variable Preferences:

```
dir Variable:*preference*
dir Variable:Maximum*
dir Variable:Report*
```

Three:

Show PSCX profile.

Presented by convergent

Scripting with Windows PowerShell

Presented by convergent

Script Files

- Scripts are just plain .ps1 text files (can’t be executed)
- Scripts are just a collection of commands
- Scripts are good for repeatable automation tasks
- Scripting tasks and console tasks are interchangeable
- Scripts are also great as “Dot Sourced” libraries
  - . .\cool-functions.ps1

Presented by convergent

```
[illegible] companyabc.com "OU=Accounts,DC=com [illegible]
[illegible]
Checking OU Name          [OK]
Checking Import File      [OK]
Please enter password: ********
```

ProvisionExchangeUsers.ps1
Uses a csv based import file to provision mail-enabled users into an Exchange Server 2007 environment.

Presented by convergent

```
[illegible] .ps1 "OU=Servers,OU=Managed Objects,DC= [illegible]
[illegible] - Administrator
[illegible]
Checking OU Name          [OK]
Do you want me to generate a random password?
[Y] Yes  [N] No  [?] Help (default is "Y"): N
Then please enter a password: ****************
Getting Server Info       [DONE]
Getting Status Info       [DONE]
Changing Passwords
```

ChangeLocalAdminPassword.ps1
Allows an Administrator to change the local Administrator password for all machines in an OU.

Presented by convergent

ServiceNow-Replication.ps1
Allows an administrator to check if a list of users are members of a specified Active Directory group.

Presented by convergent

Scripting Best Practices

- Always plan your scripts out
- Try to always sign your scripts
- Try not to use Aliases
- Write scripts using PowerShell long hand
- Always include comments in your scripts
- For large complex scripts provide documentation
- Test, test, test…

Presented by convergent

Seminar Wrap-up

Presented by convergent

PowerShell Resources

Microsoft
- http://www.microsoft.com/powershell

Blogs
- http://blogs.msdn.com/PowerShell/
- http://thepowershellguy.com/blogs/posh
- http://www.leeholmes.com/blog/
- http://www.networkworld.com/community/?q=kopczynski -or-
- http://www.taosage.net

Others
- http://www.microsoft.com/technet/scriptcenter/hubs/msh.mspx
- http://www.codeplex.com/PowerShellCX/

Presented by convergent

Convergent Computing Services

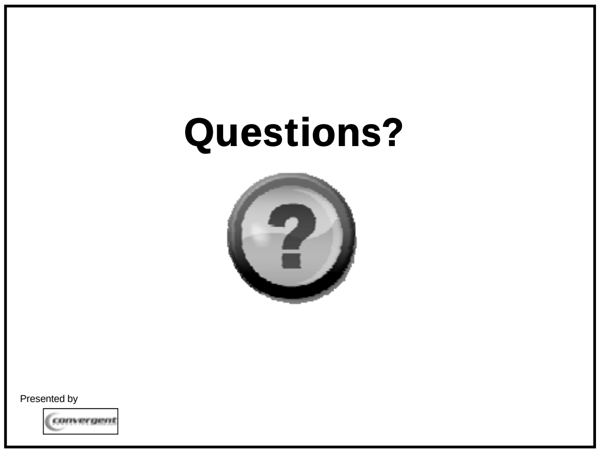
- “Automation (Admin/Ops Tasks) Health Check” – Review the level of automation within an organization on routine administrative and operational tasks, and provide a 5-7 page write-up on ways the organization can simplify these tasks through scripts
- CCO can come in and build-out Scripts to Meet Specific Requirements
- CCO can cross-train and knowledge transfer automation scripting knowledge by showing you how to Create Scripts, Understand Best Practices, and CCO can Provide Sample Code
- Some combination inbetween that meets your needs

Presented by convergent

Next Seminar

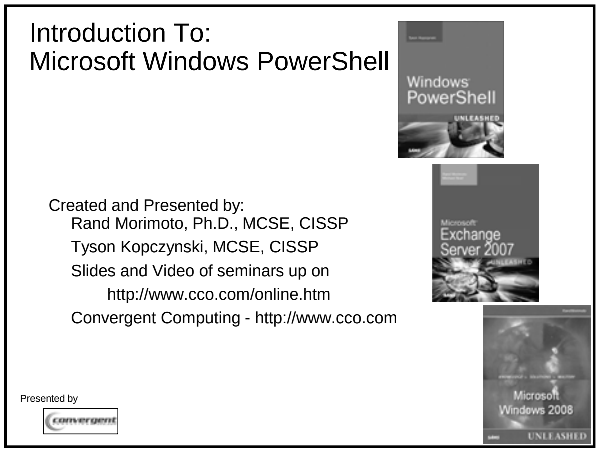
- “High Availability and Disaster Recovery of a Windows Networking Environment” (Windows, Exchange, SQL)
  - Dec 4th (Tues), 9am-11:30am (San Francisco Hyatt)
  - Dec 6th (Thurs), 9am-11:30am (Santa Clara Marriott)
  - Dec 11th (Tues), 9am-11:30am (Sacramento Hyatt)
- Details:
  - Free
  - To Register for this Seminar:
    - Email - seminar@cco.com
    - --or--
    - Call – (510) 444-5700 x179 (Seminar Registrations)
    - --or--
    - Write – “HA and DR Seminar {location}” on today's seminar eval form

Presented by convergent



Questions?

Presented by



Introduction To: Microsoft Windows PowerShell

Created and Presented by:

- Rand Morimoto, Ph.D., MCSE, CISSP
- Tyson Kopczynski, MCSE, CISSP
- Slides and Video of seminars up on
  - http://www.cco.com/online.htm
- Convergent Computing - http://www.cco.com

Presented by