PowerShell for Exchange Admins

Get-Speaker | FL

Name: Kamal Abburi
Title: Premier Field Engineer
Expertise: Exchange
Email: Kamal.Abburi@Microsoft.com
Blog: mrproactive.com

Note: Inspired by my fellow PFEs
What We Do

Proactive Services
Workshops
Health Checks
Risk Assessments
Supportability Reviews
Chalk & Talks
Knowledge Transfers
Overview

- PowerShell
- Exchange and PowerShell
- Tips and Tricks
Architecture

PowerShell Engine

Exchange Cmdlets

- AD
- Registry
- MAPI
- Metabase
What is PowerShell

- Command Driven Environment
- A scripting language
- Based on .NET Framework
- Unit of operation is a cmdlet
- Cmdlets are .NET classes
- All Exchange management operations are implemented as PowerShell cmdlets
Exchange Management Shell

- Built upon PowerShell technology
- Windows Management Framework
- PowerShell SDK available
- Role Based Access Control – Exchange 2010
PowerShell Commands in the EMC

Logs every Shell command in the EMC

<table>
<thead>
<tr>
<th>Start Execution Time</th>
<th>End Execution Time</th>
<th>Execution Status</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/11/2012 4:46:57 PM</td>
<td>9/11/2012 4:46:57 PM</td>
<td>Completed</td>
<td>Get-Recipient-PropertySet...</td>
</tr>
<tr>
<td>9/11/2012 5:51:34 PM</td>
<td>9/11/2012 5:51:34 PM</td>
<td>Completed</td>
<td>Get-MailboxStatistics -Ident...</td>
</tr>
<tr>
<td>9/11/2012 5:51:42 PM</td>
<td>9/11/2012 5:51:43 PM</td>
<td>Completed</td>
<td>Get-User-ContactPhone 111...</td>
</tr>
<tr>
<td>9/11/2012 5:51:46 PM</td>
<td>9/11/2012 5:51:46 PM</td>
<td>Completed</td>
<td>Get-Recipient-PropertySet...</td>
</tr>
</tbody>
</table>

Command:

Output message of command execution:

1 command(s) selected

Command logging started at 9/11/2012 4:46:43 PM
Basics

- **Local Shell**
  - Exchange 2007
  - Windows PowerShell host
  - Windows PowerShell snap-in, contains exchange cmdlets
  - Custom Scripts
  - Cmdlet is always run on the local Exchange server

- **Remote Shell**
  - Exchange 2010
  - `Powershell.exe` remote connection/session, No Snapin loaded
  - Uses Windows Remote Management 2.0
  - Gives access to cmdlets that are assigned based on management role
  - No need for Exchange Management Tools, but recommended

*Note: When you open shell on exchange 2010 server two sessions are created local and remote*
Edge Role

- What is that?
- Uses the local Shell only on the Edge Transport server role
- Administered individually
Remote PowerShell

Connect to an Exchange 2010 from a client with WMF installed and no 2010 management tools

$userCred = Get-Credential

$session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://casservernamefqdn/powershell -Credential $userCred

Import-PSSession $session

Remove-PSSession $session

Set-User kamal-RemotePowerShellEnabled $True
Verbs and Nouns

- New
- Get
- Set
- Add
- Update
- Remove
- Enable
- Disable
- Mount
- Dismount
- Test
- Stop
- Start
- Resume
- Retry

- MailboxDatabase
- SendConnector
- Mailbox
- TransportAgent
- ActiveSyncDevice
- SystemHealth
- ServiceHealth
- JournalRule
- MapiConnectivity
- DistributionGroup
- MailboxDatabaseCopyStatus
cmdlets

- Get-Command
- Get-ExCommand
- Get-Command *mailbox*
- Get-Command –Noun Mailbox
- Get-Command –Verb Restore

Tip: Start-Transcript
Objects

<table>
<thead>
<tr>
<th>Name</th>
<th>What_I_Wanted</th>
<th>What_I_Have</th>
<th>What_I_Should</th>
<th>What_I_Deserve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Classic</td>
<td>Family</td>
<td>Green</td>
<td>Not Affordable</td>
</tr>
<tr>
<td>Color</td>
<td>Red</td>
<td>Blue</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Speed</td>
<td>100</td>
<td>60</td>
<td>20</td>
<td>250</td>
</tr>
<tr>
<td>Mileage</td>
<td>18</td>
<td>35</td>
<td>55</td>
<td>8</td>
</tr>
</tbody>
</table>

Methods

- Start
- Drive
- Stop

Change Property

Set-Car “What_I_Have” -Color: Sparkling Bronze Metallic
Let's look at an example

- Get-Service
The Power of TAB

PowerShell auto completion
- Auto complete.
  - Try typing `get-a <TAB>`

- Scroll through parameters or cmdlets.
  - Try typing `Get-Mailbox -<TAB>`
Get-Help

Use help to find cmdlets and categories
- Get-help *User*
- Get-help -role *UM*
- Get-help –component *recipient*

Use help pages to drill into more detail
- Get-mailbox -? | more
- Help set-mailbox
- Get-help get-mailbox –Detailed
- Get-help set-mailbox –Examples
- Get-help set-mailbox –Online
- Get-Help <cmdlet> -Parameter <parameter name>
Alias

PowerShell Shorthand Notation

- Aliases are used to shorten common commands in PowerShell.
- Use `get-alias` to see all aliases
- Create your own alias using `new-alias`

<table>
<thead>
<tr>
<th>Alias</th>
<th>Cmdlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>get-childitem</td>
</tr>
<tr>
<td>cd</td>
<td>set-location</td>
</tr>
<tr>
<td>rm</td>
<td>remove-item</td>
</tr>
<tr>
<td>rmdir</td>
<td>remove-item</td>
</tr>
<tr>
<td>copy</td>
<td>copy-item</td>
</tr>
<tr>
<td>echo</td>
<td>write-output</td>
</tr>
<tr>
<td>del</td>
<td>remove-item</td>
</tr>
<tr>
<td>move</td>
<td>move-item</td>
</tr>
</tbody>
</table>
Parameters

- Provide information to the cmdlet
- Control how the cmdlet performs its task

**Verb-Noun** - **ParameterName** <**ParameterValue**>

**Types**
- Positional (Identity)
- Named (Specify the Parameter)
- Boolean($true, $false)
- Switch(confirm)
- Common(Verbosity, Debug, ErrorAction)
Syntax

- Verb-Noun - ParameterName < ParameterValue>
- hyphen indicates a parameter
- Space in Parameter Value:Double quotation marks ( " )
- Single Quote vs Double Quote
  - "$Server Example"
  - ‘$Server Example’
- Escape Character
  - “Gold is `$1600”
Exploring Parameters

- Explore the properties of output objects using format-list
  - Get-Mailbox TestUser | format-list
  - Get-Mailbox TestUser | fl *quota*
  - Get-Mailbox TestUser | fl *
  - Get-ExchangeServer –Status | fl *
  - Get-Mailbox | FT Name,Database

- Explore the property types of output objects using get-member
  - Get-storagegroup TestUser | get-member

- Tab it Set-Mailbox- <tab>
Operators

= value on the right side of the equal sign is assigned to the variable on the left side

! logical NOT operator. How do I say “Not Equal To”

>, >> send the output of a command to a file

{ } – Expression

$ Variable

+, -, *, %

:eq, -ne, -Like, -and, -or, -gt, -lt


Tip: Tee-Object
Pipeline

What is pipe between cmdlets?
Use Cases for Pipeline

- Bulk management is possible using pipelining
  - Get-Mailbox | Set-Mailbox -param1 arg1 -param2 arg2
- Piping (cmd1 | cmd2) works within same noun
  - Get-Mailbox contoso\joe | remove-mailbox
- And certain different nouns
  - Get-Mailbox contoso\joe | Test-MapiConnectivity
What are we talking about - Demo
Process Data

• Get-Car Where Color is not Red
$_$

- **Get-Car** Where **Color** is not **Red**

**What I Wanted**

- **Type:** Classic
- **Color:** Red
- **Speed:** 100
- **Mileage:** 18

**What I Have**

- **Type:** Family
- **Color:** Blue
- **Speed:** 60
- **Mileage:** 35

**What I Should**

- **Type:** Green
- **Color:** Green
- **Speed:** 20
- **Mileage:** 55

**What I Deserve**

- **Type:** Not Affordable
- **Color:** Red
- **Speed:** 250
- **Mileage:** 8
Let's Confuse you

- Get-Mailbox | Where-Object {$_ .Name -like "*admin*"}


- Get-Mailbox | ? {$_ .Name -like "*admin*"} | Select-Object Name, ProhibitSendReceiveQuota | Export-Csv -Path c:\Export.csv
Pipelining to Pipe Data between Dissimilar Nouns

- Use the data from one cmdlet with another cmdlet
- Haven't been optimized to pass objects directly

Get-Mailbox | Set-Mailbox

VS

Get-Mailbox | New-InboxRule

New-InboxRule -Name "Mark as Read" -Mailbox TEST
So .. How do we do it?

We Know about $_

Get-Mailbox | **ForEach-Object** {Write-Host $_.Name}

Get-Mailbox | **ForEach** { New-InboxRule -Name "Mark as Read" -Mailbox $_.Name -From john@contoso.com -MarkAsRead $True}
WhatIf, Confirm, and ValidateOnly Switches

- **Whatif**
  - Objects that would be affected by running the command and what changes would be made to those objects

- **Confirm**
  - Stop processing before any changes are made

- **ValidateOnly**
  - Evaluate all the conditions and requirements that are needed to perform the operation before you apply any changes
Multi Valued

@{Add="chris@contoso.com"}
@{Remove="david@contoso.com"}
Variables

- $ $
- $CurrentDate = Get-Date$
- $CurrentDate | Get-Member$
Filtering

Built in –filter

- Get-mailbox -filter {alias-like “ka*”}

Wildcard

- Get-mailbox admin*
- Get-ExchangeServer *North*
- Get-SendConnector *.test.com

Where-object (alias where)

- Get-mailbox | where {$_._.Alias –like “*admin*”}
- Get-TransportServer | where {$_._.MessageTrackingLogEnabled –eq $false}
Working with Command Output

- **Format-list (FL)**
  - Get-Mailbox | FL
  - Get-Mailbox | FL *
  - Get-Mailbox | FL *Quota*

- **Format-table (FT)**
  - Get-Mailbox | FT
  - Get-Mailbox | FT *
  - Get-Mailbox | FT Name, Alias, Database

- **Sort-object (sort)**
  - Get-mailboxstatistics | sort –property itemcount –desc

- **Group-object (group)**
  - Get-mailbox | group –property Database
Import

- Get-Content
  - Un Structured data

- Import-Csv
  - Structured data
  - First row = Properties Names
  - All other rows are data
  - Import creates the objects to work with other PowerShell cmdlets
Examples

To find the number of mailboxes per database:
Get-mailbox | group –property database

To find all users who are nearing or over their quota limit:
Get-mailboxstatistics | where {$_._.storagelimitstatus –ne ‘BelowLimit’}

To find all folders with more than 5000 items:
Get-mailboxfolderstatistics | where {$_._.itemsinfolder –gt 5000}

To find queues in the retry state:
Get-queue | where {$_._.status –eg ‘Retry”’}

Get Mailbox Database Size
Get-MailboxDatabase -Status | select ServerName,Name,DatabaseSize
Useful Links

- http://gallery.technet.microsoft.com/ScriptCenter/