

Don't Fear the Exchange Management Shell

Jim McBee

<http://www.ithicos.com>

Microsoft®
Exchange
CONNECTIONS

Who is Jim McBee!!??

- Consultant, Writer, MCSE, MVP and MCT
- Author – Wiley Mastering Exchange Server series
- Contributor – Exchange and Outlook Administrator
- Blog and web site
 - <http://mostlyexchange.blogspot.com>
 - <http://www.ithicos.com>

Is this session right for me?

- 200-level session
- Introduce PowerShell concepts
- Intended for administrators with little to no experience

Introduction

- PowerShell 101
- PowerShell vs Old-School Scripting
- Cool PowerShell One-Liners

PowerShell 101

- Flexible, object-oriented command shell
- Supplements other scripting methods
 - CMD.exe
 - VBScript
 - WMI
 - CDO / CDOEX / CDOEXM
 - Extensions can be accessed via .NET classes
- Can be used for one-off commands or to write scripts and applications

PowerShell 101: Why?

- Why is Microsoft doing this?
 - Composition and pipelining
 - Richer functionality
 - Make the command line better than UNIX
 - Provide secure remote scripting
 - Provide better batching (one-to-many)
 - Automate *and repeat* anything you can do through the GUI

PowerShell 101: Vocabulary

- **Cmdlet:**
 - Base PowerShell object that takes input, does something to it, and produces output
 - Base set of cmdlets provided with PowerShell
 - Exchange 2007/2010 adds Exchange-specific cmdlet set
 - You can write your own!

PowerShell 101: Vocabulary

- Cmdlets follow a standard verb-object naming structure
 - *Get-XXX* fetches an object or its properties
 - *New-XXX* creates something
 - *Set-XXX* sets a property on an object
 - *Format-XXX* displays object properties in a given format
- 120+ built-in cmdlets in default PowerShell install
- Exchange adds its own rich set of Exchange-specific cmdlets

Demo: Introduction to the PowerShell

- **Verb-Noun combinations**
 - Verbs: Get, Set, New, Delete, Mount, Disable
 - Nouns: User, Mailbox, MailboxServer, TransportServer, Database, Contact, DistributionGroup, ActiveSyncPolicy
- **Not case sensitive**
- **Tab completion**
 - Complete cmdlets and parameters
- **Getting help**
 - Help and Get-ExCommand *mailbox*
 - Get-Mailbox -?
 - Get-Help Get-Mailbox –Full
 - Get-Help Get-Mailbox –Example
 - Using the EMC!
- **We can view the properties used by a cmdlet**
 - Get-Mailbox | Get-Member –MemberType Properties
 - Get-Mailbox “Lee Adama” | Format-List
- **Output**
 - Outputs to the screen “text”
 - Output to the shell is objects

PowerShell + Exchange 2003

- If you're running Exchange 2003, you can still use PowerShell
 - Install .NET Framework 2.0
 - Install PowerShell
- What can you do with it? Plenty!
 - Service management and control
 - WMI monitoring
 - WMI property setting
 - Basic Active Directory administration
 - Quest tools

<http://www.quest.com/powershell/activeroles-server.aspx>

PowerShell in Exchange 2007/2010

- PowerShell is the core of Exchange Management Console
 - All EMC actions really call PowerShell cmdlets
 - All those cmdlets are available from the command line
- Don't let the cmdlet idea fool you
 - Some Exchange cmdlets are extremely rich, e.g. move-mailbox
 - Many cmdlets compress big functionality into one line

Mailbox-Enabling a User

```
' get the default and config NC names
Set oIADS = GetObject("LDAP://RootDSE")
strDefaultNC = oIADS.Get("defaultnamingcontext")
strConfigNC = oIADS.Get("configurationNamingContext")
strContainer= "/CN=Users," & strDefaultNC
Set objContainer = GetObject("LDAP://" & strDCName & strContainer)

' find the target user and connect to it
Set oIADSUser = GetObject("LDAP://Joe User,CN=Users," & strDefaultNC)
Set oMailBox = oIADSUser
Set oConnection = CreateObject("ADODB.Connection")
set oCommand = CreateObject("ADODB.Command")
Set oRecordSet = CreateObject("ADODB.Recordset")
oConnection.Provider = "ADsDSOObject"
oConnection.Open "ADs Provider"
```

Mailbox-Enabling a User

' Build the query to find the private MDBs. Use the first one if any are found.

```
strQuery = "<LDAP://" & strConfigNC & _  
    ">;(objectCategory=msExchPrivateMDB);name,adspath;subtree"
```

```
oCommand.ActiveConnection = oConnection
```

```
oCommand.CommandText = strQuery
```

```
Set oRecordSet = oCommand.Execute
```

```
If Not oRecordSet.EOF Then
```

```
    oRecordSet.MoveFirst
```

```
    firstMDB = CStr(oRecordSet.Fields("ADsPath").Value)
```

```
Else
```

```
    firstMDB = ""
```

```
End If
```

' create the mailbox

```
oMailbox.CreateMailbox firstMDB
```

```
oIADsUser.SetInfo
```

EMS: Mailbox-Enabling a User

```
Enable-Mailbox "JoeUser" -Database  
MDB001
```

Note that you could also do this on many users at once, e.g.

```
Get-DistributionGroupMember "New Hires" | Enable-Mailbox -  
Database MDB001
```

Turning on OWA Attachment Blocking

```
' ----- SCRIPT CONFIGURATION -----  
strOWA = "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\  
strOwa = strOWA & "MSExchangeWeb\OWA\  
strBlockList = "pst, tmp, pl, exe, cmd, pif, bat, msh"  
' ----- END CONFIGURATION -----
```

```
Set objWSH = wscript.CreateObject("WScript.Shell")  
objWSH.RegWrite strOWA, "Level1FileTypes", "REG_SZ", strBlockList
```

Run this on **each** Exchange 2003 server

EMS: Turning on OWA Attachment Blocking

```
Get-OWAVirtualDirectory | Set-OWAVirtualDirectory -BlockedFileTypes ".PST"
```

Note that this sets the parameter on all the OWA instances in your organization, at once, with no extra steps!

Cool EMS One-Liners

- Get the number of user mailboxes for each database
 - *Get-MailboxStatistics | Group MailboxDatabase | Format-Table count,name*

Cool EMS One-Liners

- Which mailboxes are in a given database?
 - *Get-Mailbox | Group Database | Format-List*
 - Lists each MDB and shows you which users are in each database

Cool EMS One-Liners

- Mass-set properties on all members of a group
 - `Get-DistributionGroupMember "Executives" | Set-Mailbox -SendStorageQuota 500MB -UseDatabaseQuotaDefaults: $False`

Cool EMS One-Liners

- Find all files created on a certain day and move them

- ```
dir *.eml | where
 {$_.LastWriteTime -like "04/02 *"}
 | move-item -destination
 c:\oldSpam
```

- You can just as easily remove them, rename them, or do other things

# Cool EMS One-Liners

- **Get all the users in an OU**
  - `Get-User -OrganizationalUnit "cta.net/Pilots"`
- **Mailbox-enable every user in an OU**
  - `Get-User -OrganizationalUnit "gotham.ci.us/PublicSafety" | Where {$_.RecipientType -eq "user"} | Enable-Mailbox -database "Gotham PD"`

# Cool EMS One-Liners

- **Retry any message queue that has more than 50 pending messages**
  - `get-queue | where-object { $_.MessageCount -gt 50 } | retry-queue`

# More One-Liners

- Output (output) of some cmdlets can be piped to another cmdlet to be used as input
- This allows the creation of “one-liners”
  - `Get-MailboxStatistics | where {$_.DatabaseName -eq “Mailbox Database”} | Format-Table DisplayName, ItemCount, TotalItemSize,StorageLimitStatus`
  - `Get-Mailbox “Lee.Adama” | Set-Mailbox –ProhibitSendQuota:75000KB`
  - `Get-DistributionGroupMember “Raptor Pilots” | Move-Mailbox – TargetDatabase “Raptor Pilots”`
  - `Get-DistributionGroupMember “Raptor Pilots” | Set-Mailbox – IssueWarningQuota:100MB –ProhibitSendQuota:125MB – ProhibitSendReceiveQuota:150MB – UseDatabaseQuotaDefaults:$False`

# Demo: Creating Users and Mailboxes

- Creating a text file

Name,Database,OrganizationalUnit,UserPrincipalName

Saul Tigh,Mailbox Database,colonialfleet.local/Engineering,Saul.Tigh@colonialfleet.local

Sharon Agathon,Mailbox Database,colonialfleet.local/Engineering,Sharon.Agathon@colonialfleet.local

Tom Zarek,Mailbox Database,colonialfleet.local/Engineering,Tom.Zarek@colonialfleet.local

Laura Roslin,Mailbox Database,colonialfleet.local/Engineering,Laura.Roslin@colonialfleet.local

Samuel Anders,Mailbox Database,colonialfleet.local/Engineering,Samuel.Anders@colonialfleet.local

- Script to read this text file and create users

```
$Users = Import-Csv C:\Demo\newaccounts.txt
```

```
$Users
```

```
$Password = Read-Host "Please enter a password for the users" -AsSecureString
```

```
Foreach ($User in $Users) {
```

```
 New-Mailbox
```

```
 -Name $User.Name
```

```
 -Database $User.Database
```

```
 -OrganizationalUnit $User.OrganizationalUnit
```

```
 -UserPrincipalName $User.UserPrincipalName
```

```
 -Password $Password
```

```
}
```

# E2K7 SP1 and E2K10 Management Improvements

- Includes some major EMC improvements
  - Public folder management tools
  - POP / IMAP server management tools
  - Clustered mailbox server management
  - Remoting (in E2K10)
- There are EMS improvements as well
  - Import and export mailboxes to PST!
  - Improved tools for bulk mailbox manipulation
  - Some syntax improvements

# PowerShell Tricks to Know

- *Get-command -synopsis* will give you all flags for the specified cmdlet
  - *Get-command -name get-service -synopsis*
- *Get-excommand* can be used to search for Exchange-specific commands
  - *Get-ExCommand \*ailbox\* | where {\$\$.Name -eq "Cmdlet"}* tells you what command exist
  - *Follow up by piping the cmdlet name to get-member*
    - *Get-MailboxServer | Get-Member -MemberType property* to get a list of properties

# For more information

- Visit the Exchange home page
  - <http://www.microsoft.com/exchange/>
- Exchange Team blog
  - <http://msexchangeteam.com>

# Where to Learn More

- Exchange 2007/2010 online help: canonical source for what Exchange cmdlets do
- PowerShell team blog
  - <http://blogs.msdn.com/PowerShell>
- Vivek Sharma's blog
  - <http://viveksharma.com/techlog>
- Free EMS Quick Reference
  - <http://examples.oreilly.com/9780735627192/>

# Where to Learn More

- *Microsoft Exchange 2010 PowerShell Cookbook (Packt Publishing)*
  - *Newest book on the market*
- *PowerShell (Oakley; O'Reilly Media; ISBN 0-596-10009-4)*
  - Quick, light intro; no Exchange content
- *PowerShell: TFM (Jones & Hicks; SAPIEN Press; 0-977-65972-0)*
  - Written by Windows scripting gurus Don Jones & Jeffrey Hicks

Questions?

Thanks for attending!

Microsoft®  
**Exchange**  
CONNECTIONS

# Your Feedback is Important

Please fill out a session evaluation form  
drop it off at the conference registration  
desk.

Thank you!