nordic infrastructure conference

4th Edition

David O'Brien (@david_obrien)

PowerShell DSC – Resources

David O'Brien

MVP for SysCtr Cloud and Datacenter Management
Heavily focused on Automation

- Powershell
- SMA
- Orchestrator
- ConfigMgr

Blog: www.david-obrien.net
Twitter: @david_obrien
Principal Consultant at Dilignet

Agenda

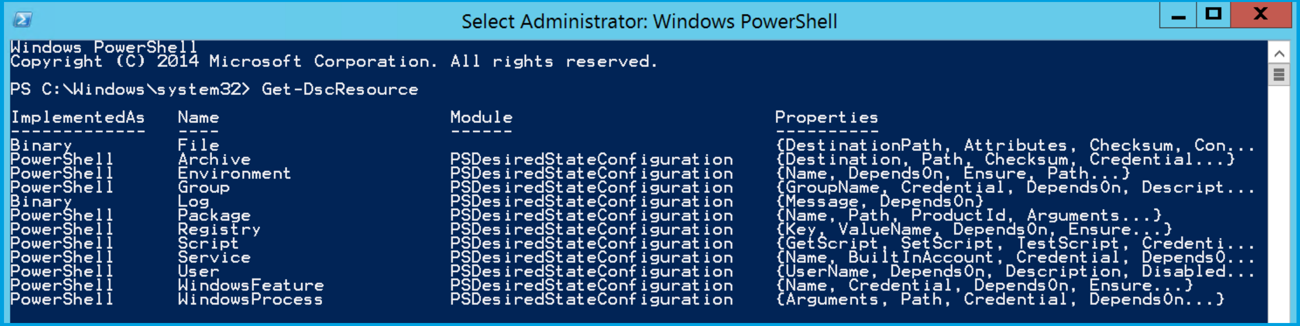
- What are resources?
- What are resources for?
- Some ground rules…
- Good practices
- Demos
- Q&A

The “Make it so.”

- The smarts behind the scene
- Very “dev” side of DevOps (at least for me)
- This is where the future work will be
- Declarative vs imperative syntax

What is a resource?

- Resource Module = PowerShell Module
- Inside of a Resource Module are resources
  - 1 Module can house multiple resources
- Get-DscResource
  - 12 on WMF4
  - 15 on WMF5 (and much faster enumeration)



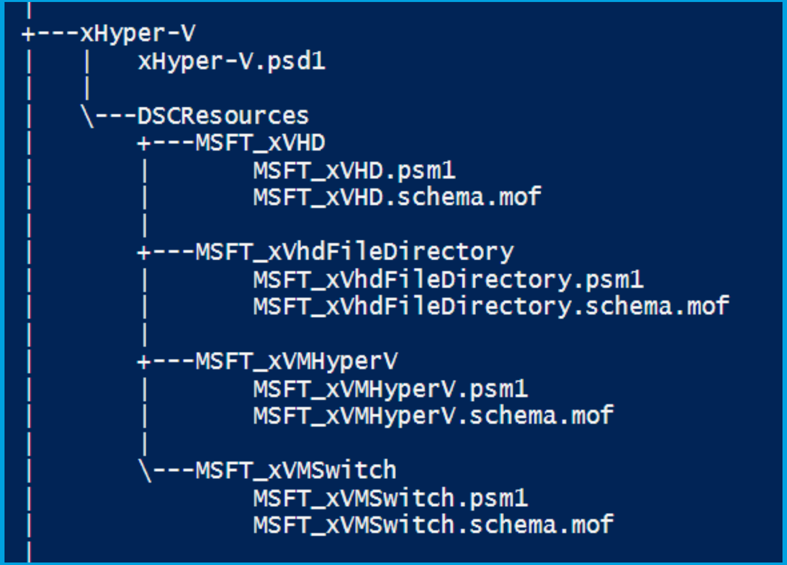
Select Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Get-DscResource

ImplementedAs   Name               Module                      Properties
-------------   ----               ------                      ----------
Binary          File                                           {DestinationPath, Attributes, Checksum, Con...
PowerShell      Archive            PSDesiredStateConfiguration {Destination, Path, Checksum, Credential...}
PowerShell      Environment        PSDesiredStateConfiguration {Name, DependsOn, Ensure, Path...}
PowerShell      Group              PSDesiredStateConfiguration {GroupName, Credential, DependsOn, Descript...
Binary          Log                PSDesiredStateConfiguration {Message, DependsOn}
PowerShell      Package            PSDesiredStateConfiguration {Name, Path, ProductId, Arguments...}
PowerShell      Registry           PSDesiredStateConfiguration {Key, ValueName, DependsOn, Ensure...}
PowerShell      Script             PSDesiredStateConfiguration {GetScript, SetScript, TestScript, Credenti...
PowerShell      Service            PSDesiredStateConfiguration {Name, BuiltInAccount, Credential, DependsO...
PowerShell      User               PSDesiredStateConfiguration {UserName, DependsOn, Description, Disabled...
PowerShell      WindowsFeature     PSDesiredStateConfiguration {Name, Credential, DependsOn, Ensure...}
PowerShell      WindowsProcess     PSDesiredStateConfiguration {Arguments, Path, Credential, DependsOn...}
```

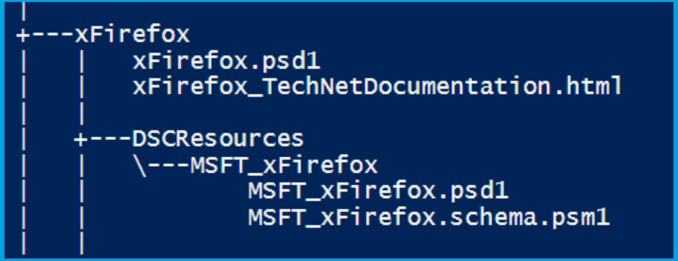
What are resources for?

- Like PowerShell cmdlets they implement “Commands” with “Parameters” the DSC engine knows how to execute
- Copy your modules to C:\Program Files\WindowsPowerShell\Modules
  - **NOT** to C:\Windows\System32\WindowsPowerShell\v1.0\Modules
  - If Get-DscResource can’t find your custom modules here, check for KB2883200
- Code reuse!

Folder Structure



```
|
+---xHyper-V
|   |   xHyper-V.psd1
|   |
|   \---DSCResources
|       +---MSFT_xVHD
|       |       MSFT_xVHD.psm1
|       |       MSFT_xVHD.schema.mof
|       |
|       +---MSFT_xVhdFileDirectory
|       |       MSFT_xVhdFileDirectory.psm1
|       |       MSFT_xVhdFileDirectory.schema.mof
|       |
|       +---MSFT_xVMHyperV
|       |       MSFT_xVMHyperV.psm1
|       |       MSFT_xVMHyperV.schema.mof
|       |
|       \---MSFT_xVMSwitch
|               MSFT_xVMSwitch.psm1
|               MSFT_xVMSwitch.schema.mof
|
```

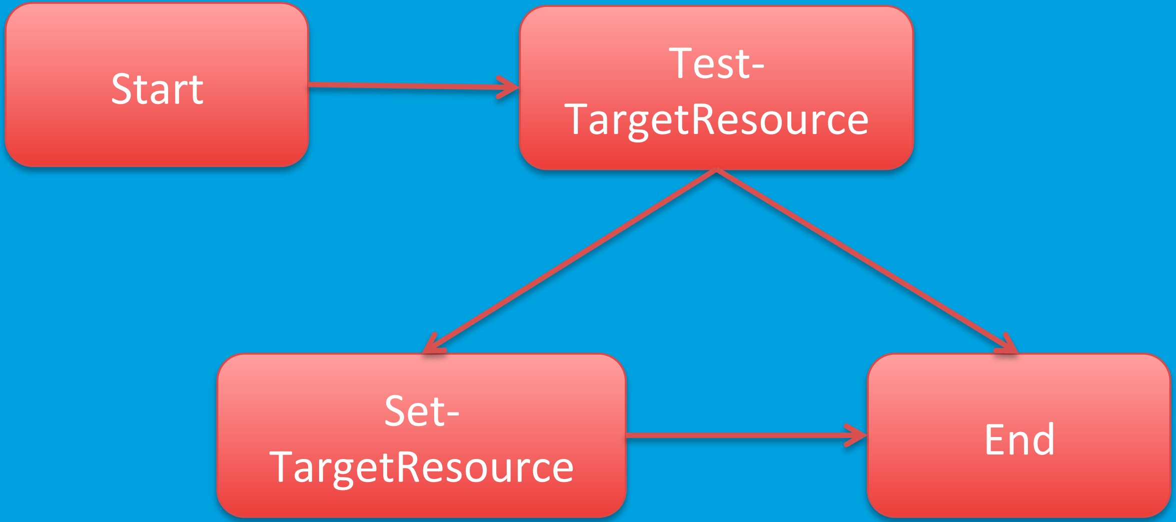


```
|
+---xFirefox
|   |   xFirefox.psd1
|   |   xFirefox_TechNetDocumentation.html
|   |
|   +---DSCResources
|   |   \---MSFT_xFirefox
|   |           MSFT_xFirefox.psd1
|   |           MSFT_xFirefox.schema.psm1
|   |
```

Some quick ground rules…

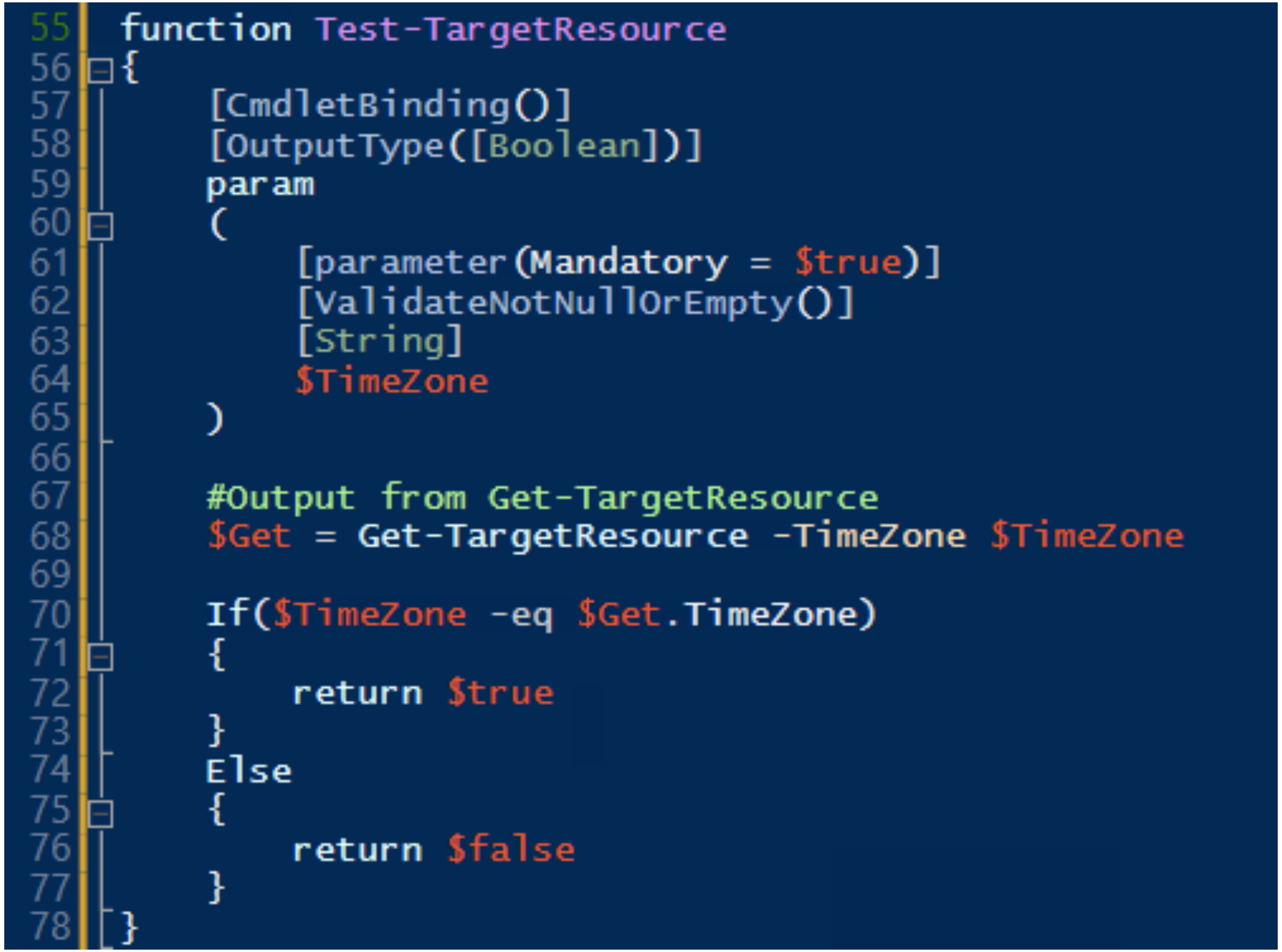
- Get-, Set- and Test-TargetResource
- As a minimum these three Functions need to be present in a resource
    - Output:
        - Test-TargetResource = Boolean
        - Set-TargetResource = none
        - Get-TargetResource = Hashtable of current values
    - Require same set of parameters

Execution Phase



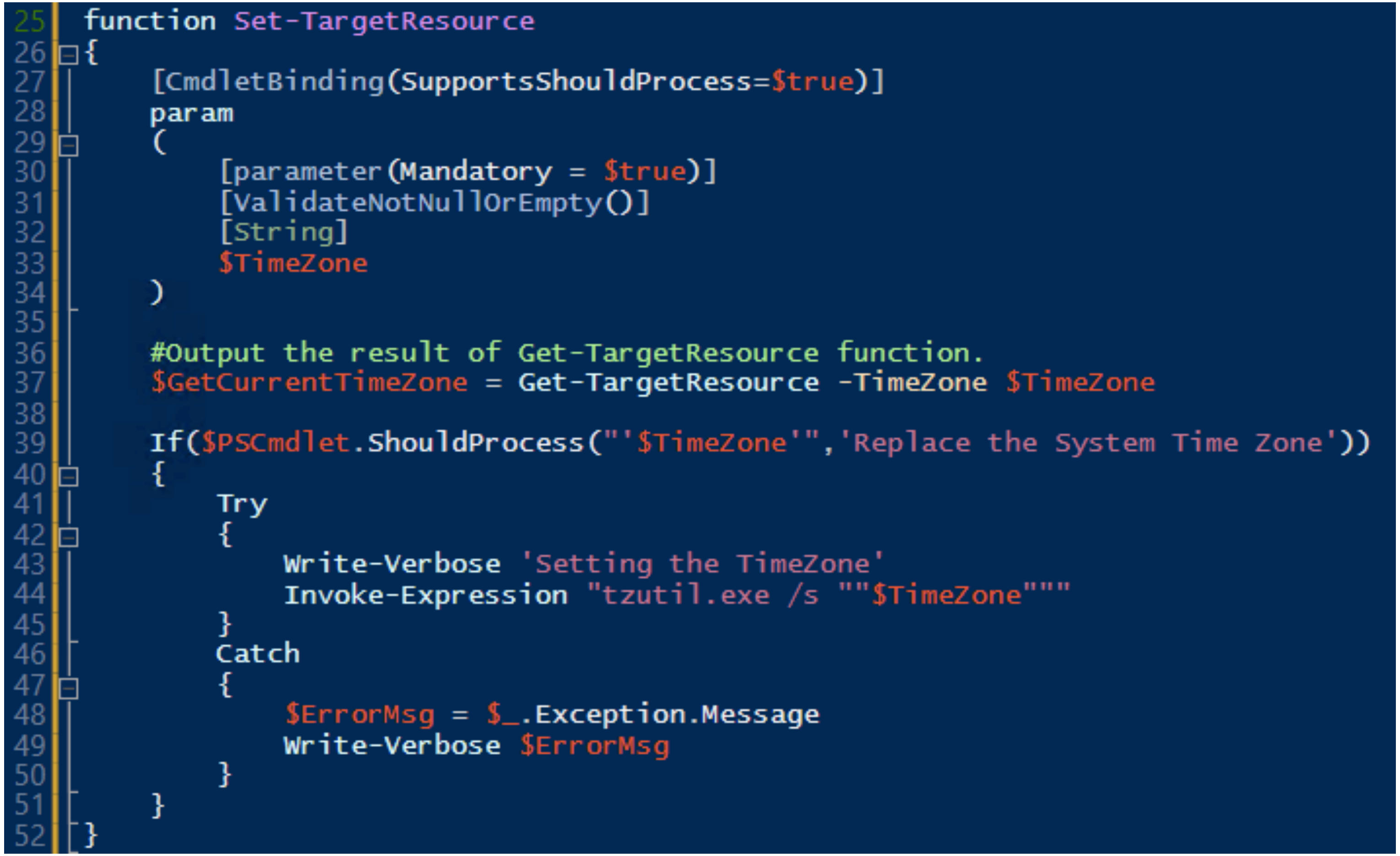
Test-TargetResource

- Checks current state of the system
- Output (return) is either Boolean $true or $false
    - If $true
        - Desired State already in place
    - If $false
        - Not in Desired State -> Set-TargetResource
- Always remember the **PRINCIPLE OF IDEMPOTENCE**


```
function Test-TargetResource
{
    [CmdletBinding()]
    [OutputType([Boolean])]
    param
    (
        [parameter(Mandatory = $true)]
        [ValidateNotNullOrEmpty()]
        [String]
        $TimeZone
    )

    #Output from Get-TargetResource
    $Get = Get-TargetResource -TimeZone $TimeZone

    If($TimeZone -eq $Get.TimeZone)
    {
        return $true
    }
    Else
    {
        return $false
    }
}
```

Set-TargetResource

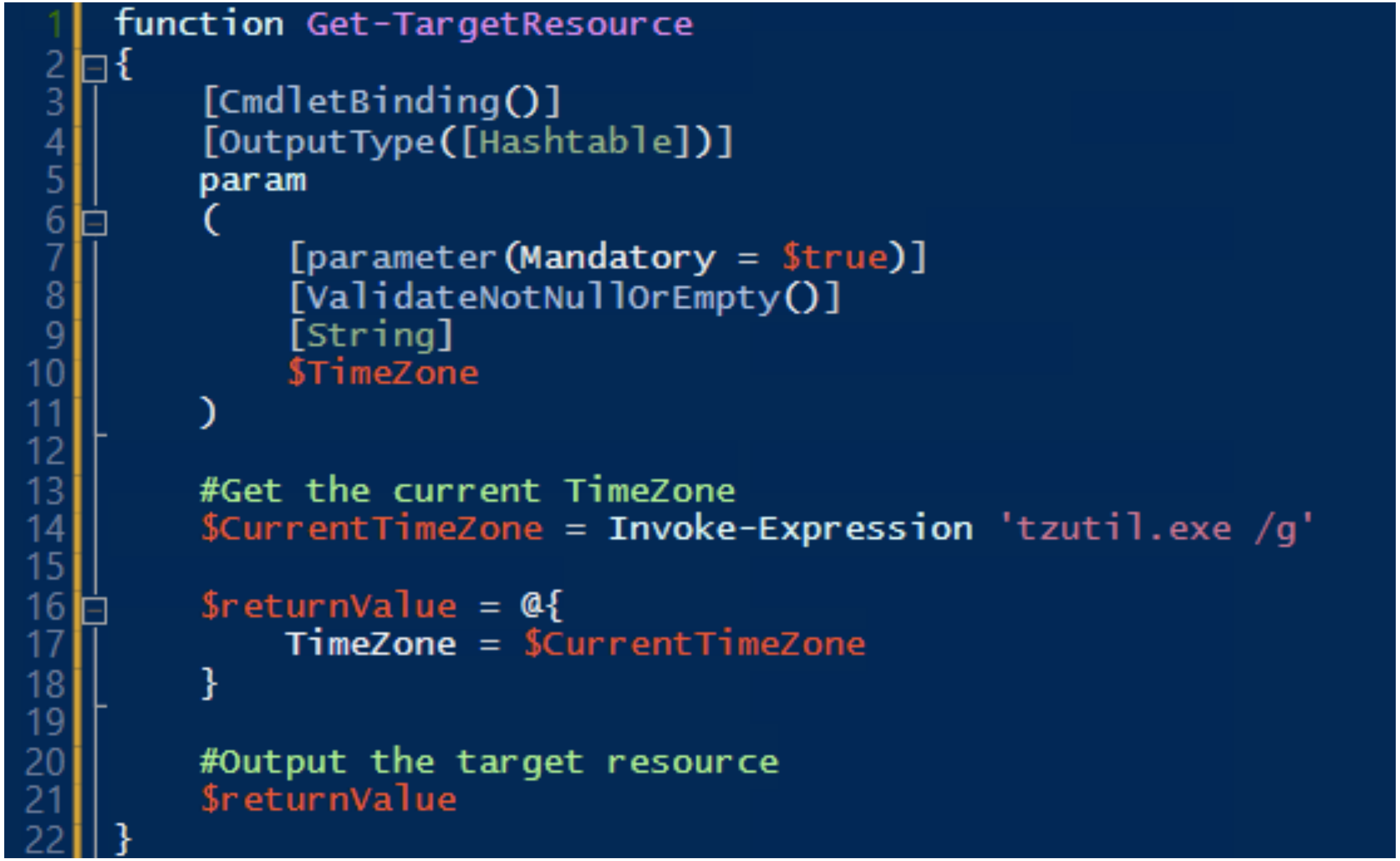
- Must only run if Test-TargetResource returns $false
    - Brings the System (back) to the Desired State
- no output



```
function Set-TargetResource
{
    [CmdletBinding(SupportsShouldProcess=$true)]
    param
    (
        [parameter(Mandatory = $true)]
        [ValidateNotNullOrEmpty()]
        [String]
        $TimeZone
    )

    #Output the result of Get-TargetResource function.
    $GetCurrentTimeZone = Get-TargetResource -TimeZone $TimeZone

    If($PSCmdlet.ShouldProcess("'$TimeZone'",'Replace the System Time Zone'))
    {
        Try
        {
            Write-Verbose 'Setting the TimeZone'
            Invoke-Expression "tzutil.exe /s ""$TimeZone"""
        }
        Catch
        {
            $ErrorMsg = $_.Exception.Message
            Write-Verbose $ErrorMsg
        }
    }
}
```

Get-TargetResource

- Does not take part in execution process
- Most implementations seem to
    - either ignore it or
    - Use it to be called from Test-TargetResource



```
function Get-TargetResource
{
    [CmdletBinding()]
    [OutputType([Hashtable])]
    param
    (
        [parameter(Mandatory = $true)]
        [ValidateNotNullOrEmpty()]
        [String]
        $TimeZone
    )

    #Get the current TimeZone
    $CurrentTimeZone = Invoke-Expression 'tzutil.exe /g'

    $returnValue = @{
        TimeZone = $CurrentTimeZone
    }

    #Output the target resource
    $returnValue
}
```

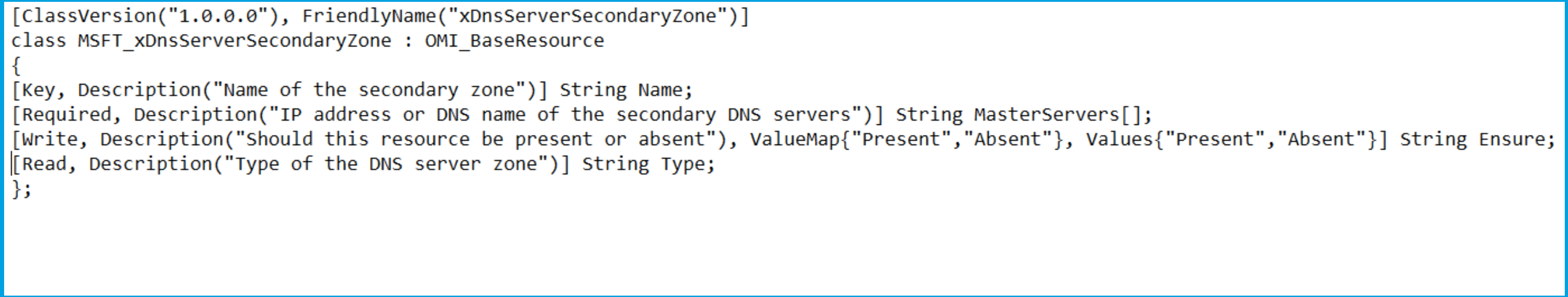
Finish it off

- Reboot the machine
    - Set-TargetResource requires a reboot?
    - $global:DscMachineStatus = 1

    - **Make sure Test-TargetResource works properly!**

MOF Files

- “Describes” the resource



```
[ClassVersion("1.0.0.0"), FriendlyName("xDnsServerSecondaryZone")]
class MSFT_xDnsServerSecondaryZone : OMI_BaseResource
{
[Key, Description("Name of the secondary zone")] String Name;
[Required, Description("IP address or DNS name of the secondary DNS servers")] String MasterServers[];
[Write, Description("Should this resource be present or absent"), ValueMap{"Present","Absent"}, Values{"Present","Absent"}] String Ensure;
[Read, Description("Type of the DNS server zone")] String Type;
};
```

- Classname.schema.mof

Module Manifest

- New-ModuleManifest
- Required
  - Especially for the Moduleversion (Pull Server)

Some “good” practices

- Export-ModuleMember *-TargetResource to only export the 3 main functions
- If applies, add “Ensure” as a key property to your resource
- Test-TargetResource should be **fast**
    - Called with every consistency check
- Validate input parameters
- Write-Verbose
- If a dependency is missing, bomb out, don’t install it
    - i.e. WindowsFeature should install a Feature, not your custom resource
- Test-xDscResource

Take aways

- Not much different to PowerShell modules
- Re-use your code!
- You don’t need to be a developer to write your own modules
- More than one way to skin the cat
    - DSC is not the answer to everything!

PowerShell v5

- Implement resources as classes
    - No need for schema.mof anymore
- Faster enumeration of resources

Q & A

David O’Brien
OBrien.David@Outlook.com
@david_obrien
http://www.david-obrien.net