



DSC in PowerShell 4.0

Martin Schwartzman

Senior Premier Field Engineer
maschvar@microsoft.com

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

© 2014 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
<http://www.microsoft.com/about/legal/permissions/>

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

WMF 4.0

- PowerShell 4.0
- PowerShell Integrated Scripting Environment (ISE)
- PowerShell Web Services (OData IIS Extension)
- Remote Management (WinRM)
- Management Infrastructure (WMI)
- PowerShell Desired State Configuration (DSC)

Windows Management Framework 4.0

<http://www.microsoft.com/en-us/download/details.aspx?id=40855>

Requires .NET Framework 4.5

<http://www.microsoft.com/en-us/download/details.aspx?id=30653>

The good and the bad

Scale	
😊	Business is growing
😞	More servers = More Failures

Rapid changes	
😊	Adjust and respond fast
😞	More changes = More Failures

Scale + Rapid changes = Constant failures?

Desired State Configuration

Enables you to **ensure** that the components of your data center have the **correct configuration**

Allows **"continuous deployment"** and prevents **"configuration drift"**

Uses language extensions and providers to enable declarative, autonomous and idempotent (repeatable) Deployment, Configuration and Conformance of **standards-based managed elements**

Imperative vs. Declarative

- **Imperative** - **How** a task should be performed
- **Declarative** – **What** needs to be done

Meaning, you'll write "configuration documents", not scripts

DSC Practical Applications

- Install or remove server roles and features
- Manage registry settings
- Manage files and directories
- Start, stop, and manage processes and services
- Manage local groups and user accounts
- Install and manage packages such as .msi and .exe
- Manage environment variables
- Run Windows PowerShell scripts
- and many more...

DSC Components

- PowerShell Language Extensions
- MOF (Managed Object Format) Instance doc
- WMI Service
- Local Configuration Store
- Local Configuration Manager (a WMI Provider)
- Configuration Agent (CA)
- Resource Provider

DSC Resources

**Archive
Resource**

**Environment
Resource**

File Resource

**Group
Resource**

Log Resource

**Package
Resource**

**Process
Resource**

**Registry
Resource**

**Role
Resource**

**Script
Resource**

**Service
Resource**

**User
Resource**

DSC Resource Kit Waves (3)

Active Directory	Computer Management	Database
Failover Cluster	Hyper-V	Networking
Remote Desktop	SmbShare	SQL
System Security	Web Admin- istration	...

And you can build your own custom resources

Archive Resource

- Provides a mechanism to unpack archive (.zip) files at a specific path on a target node

```
Archive myArchiveExample
{
    Ensure = "Present" # You can also set Ensure to "Absent"
    Path = "\\RepositoryServer01\Share\SysinternalsSuite.zip"
    Destination = "C:\Tools\SysinternalsSuite"
}
```

Demo

Archive Resource

Environment Resource

- Provides a mechanism to manage system environment variables on a target node

```
Environment myEnvironmentExample
{
    Ensure = "Present" # You can also set Ensure to "Absent"
    Name = "_NT_SYMBOL_PATH"
    Value = "SRV*C:\symbols*http://msdl.microsoft.com/download/symbols"
}
```

File Resource

- Provides a mechanism to manage files and folders on a target node

```
File myDirectoryCopy
{
    Ensure = "Present" # You can also set Ensure to "Absent"
    Type = "Directory" # Default is "File".
    Recurse = $true # Ensure presence of subdirectories, too
    SourcePath = "\\DC01\DSC\FourthCoffeeWebSite"
    DestinationPath = "C:\inetpub\wwwroot"
}
```

Group Resource

- Provides a mechanism to manage local groups on a target node

```
Group myGroupExample
{
    # This will remove the myTestGroup ,if present
    # To create a new group, set Ensure to "Present"
    Ensure = "Absent"
    GroupName = "myTestGroup"
}
```

Log Resource

- Provides a mechanism to write messages to the Microsoft-Windows-Desired State Configuration/Analytic event log

```
Log myLogExample
{
    Message = "This is a test message."
}
```


Package Resource

- Provides a mechanism to install or uninstall packages (setup.exe or *.msi), on a target node

```
Package myPackageExample
{
    Ensure = "Present" # You can also set Ensure to "Absent"
    Path = "\\DC01\DSC\WPTx64-x86_en-us.msi"
    Name = "WPTx64"
    ProductId = "{986EABFC-92F6-CECD-9E5A-B13CAC40BB1D}"
    Arguments = "/qn"
    LogPath = "C:\myMsiInstall.log"
}
```

Demo

Package Resource and Configuration Parameters

Process Resource

- Provides a mechanism to configure processes on a target node

```
WindowsProcess myProcess
{
    Arguments = "/a /b /c"
    Path = "C:\myPath\myApp.exe"
    Ensure = "Absent"
    WorkingDirectory = "C:\Windows"
}
```

Registry Resource

- Provides a mechanism to manage registry keys and values on a target node

```
Registry myRegistryExample
{
    Ensure = "Present" # You can also set Ensure to "Absent"
    Key = "HKEY_LOCAL_MACHINE\SOFTWARE\myApplication"
    ValueName = "DatabaseServerName"
    ValueData = "SQLServer01.contoso.com"
}
```

Role Resource

- Provides a mechanism to ensure that roles and features are added or removed on a target node

```
WindowsFeature myIIS
{
    Ensure = "Present"
    Name = "Web-Server"
}
```

Script Resource

- Provides a mechanism to run Windows PowerShell script blocks on target nodes
- The TestScript block runs first
- If it returns False, the SetScript block will run

```
Script myScriptExample
{
    SetScript = { Set-Content -Path "C:\Temp\TestFile.txt" -Value "Test" }
    TestScript = { Test-Path "C:\Temp\TestFile.txt" -PathType Leaf}
    GetScript = { <# This must return a hash table #> }
}
```

Service Resource

- Provides a mechanism to manage services on the target node

```
Service myServiceExample
{
    Name = "TermService"
    StartupType = "Manual"
    State = "Running"
}
```

User Resource

- Enables local user accounts management on a target node

```
User myUserExample
{
    Ensure = "Present" # To delete a user account, set Ensure to "Absent"
    Username = "Martin"
    Password = $passwordCred # This needs to be a credential object
    DependsOn = "[Group]myGroupExample" # Configures the group first
}
```


Deploying Configuration

Configuration MyConfig

```
{  
    # A Configuration block can have zero or more Node blocks  
    Node "Server01"  
    {  
        # Next, specify one or more resource blocks  
  
        # Ensure the Web Server (IIS) role is installed  
        WindowsFeature MyRoleIIS  
        {  
            Ensure = "Present" # To uninstall the role, set to "Absent"  
            Name = "Web-Server"  
        }  
    }  
}
```

PS C:\> MyConfig

PS C:\> Start-DscConfiguration -Wait -Verbose -Path .\MyConfig

Demo

Web Server installation, configuration and continuous deployment

*-DscConfiguration

- Start-DscConfiguration
 - Applies configuration to nodes
- Get-DscConfiguration
 - Gets the node's current configuration
- Test-DscConfiguration
 - Tests whether the actual configuration on the nodes matches the desired configuration
- Restore-DscConfiguration
 - Restores the previous configuration for the node

Demo

Get, Test and Restore Configuration

Demo

Multi-node configuration and parameters

Push Model

Authoring Phase
(May include imperative as well as declarative code)

PS V1, V2, V3

PS V4*

3rd party languages and tools

Staging Phase

- Fully declarative configuration representation using DMTF standard MOF instances
- Configuration is calculated for all nodes

Configuration Staging Area
(Contains DSC data)

Set Phase

(Declarative configuration is reified through imperative providers.)

Local Configuration Store

Parser and Dispatcher

Imperative Providers

*When authoring in PowerShell, on top of PSV3 imperative features, PSV4 adds:

- Declarative syntax extensions
- Schema-driven Intellisense
- Schema validation (early-binding)

Providers implement changes:

- Monotonic
- Imperative
- Idempotent

Pull Model

Authoring Phase
(May include imperative as well as declarative code)

PS V1, V2, V3

PS V4*

3rd party languages and tools

Staging Phase

- Fully declarative configuration representation using DMTF standard MOF instances
- Configuration is calculated for all nodes

Pull Server
(Contains DSC data and Modules)

Set Phase

(Declarative configuration is reified through imperative providers.)

Local Configuration Store

Parser and Dispatcher

Imperative Providers

*When authoring in PowerShell, on top of PSV3 imperative features, PSV4 adds:

- Declarative syntax extensions
- Schema-driven Intellisense
- Schema validation (early-binding)

Providers implement changes:

- Monotonic
- Imperative
- Idempotent

Push vs. Pull

- Push
 - Simple
 - Control
 - Immediately applies the configuration
- Pull
 - Complete Automation
 - Scalability
 - Dedicated Pull Server

Local Configuration Manager

- Runs on all nodes (as a WMI provider)
- Responsible for calling the DSC resources
- LCM Properties can be set using DSC

```
PS> Get-DscLocalConfigurationManager
```

```
AllowModuleOverwrite           : False
CertificateID                   :
ConfigurationID                 :
ConfigurationMode               : ApplyAndMonitor
ConfigurationModeFrequencyMins : 30
Credential                     :
DownloadManagerCustomData      :
DownloadManagerName            :
RebootNodeIfNeeded             : False
RefreshFrequencyMins           : 15
RefreshMode                     : PUSH
PSComputerName                 :
```

ConfigurationMode

- Apply

Will apply the configuration once and after a successful run is logged, it will stop attempting to apply configuration or checking the configuration

- ApplyAndMonitor

Will apply a configuration as in Apply, but will continue to validate that a node is configured as described. No corrective action will take place if there is configuration drift

- ApplyAndAutoCorrect

Applies a configuration and checks it regularly. If configuration drift is detected, the configuration manager will attempt to return the machine to the *desired state*

*FrequencyMins

- ConfigurationModeFrequencyMins

Determines how frequently the configured method (RefreshMode) is run. In the case of a pull server, this is how frequently the pull server will be checked for updated configurations. The minimum value for this is 30.

- RefreshFrequencyMins

Determines how often DSC runs an integrity check against the cached configuration value. The minimum value for this setting is 15 minutes

Configuring LCM

```
Configuration mySetLcmConfiguration
{
    Node "localhost"
    {
        LocalConfigurationManager
        {
            ConfigurationMode = "ApplyAndAutoCorrect"
            RefreshFrequencyMins = 30
            ConfigurationModeFrequencyMins = 60
            RefreshMode = "PUSH"
            RebootNodeIfNeeded = $true
        }
    }
}
```

```
PS> mySetLcmConfiguration
```

```
PS> Set-DscLocalConfigurationManager -Path .\mySetLcmConfiguration -Verbose
```

Demo

LCM Configuration

Desired State Configuration...

- Declarative syntax
- Simplify configuration
- Prevent configuration drift
- Enable continuous deployment

*"Hey, you're a Web server
Here's what you should look like. Get to it, and stay
that way!"*

Q & A





Microsoft