Desired State Configuration Governance

Contents

[Overview 1](#_Toc417476773)

[Resources 1](#_Toc417476774)

[Quick links (also available from the Overview page): 1](#_Toc417476775)

[Governance 2](#_Toc417476776)

[Basic Tenets 2](#_Toc417476777)

[Design Process 2](#_Toc417476778)

[MOF file creation 2](#_Toc417476779)

[External References 3](#_Toc417476780)

[Why DSC? 3](#_Toc417476781)

[How to DSC 3](#_Toc417476782)

Document reviewed: 7/09/2015 tpluciennik

# Overview

DSC is a new management platform in Windows PowerShell (4.x or greater) that enables deploying and managing configuration data for software services and managing the environment in which these services run.

DSC provides a set of Windows PowerShell language extensions, new Windows PowerShell cmdlets, and resources that you can use to declaratively specify how you want your software environment to be configured. It also provides a means to maintain and manage existing configurations.

## Resources

Microsoft DSC Overview: <https://technet.microsoft.com/en-us/library/dn249912.aspx>

### Quick links (also available from the Overview page):

* Getting started: <https://technet.microsoft.com/en-us/library/dn249918.aspx>
* Built-In DSC Resources: <https://technet.microsoft.com/en-us/library/dn249921.aspx>
* Powershell Cmdlets: <https://technet.microsoft.com/en-us/library/dn407385(v=wps.630).aspx>
* Creating DSC Custom Resources: <https://technet.microsoft.com/en-us/library/dn249927.aspx>

# Governance

* Data driven; use PS1 files to create MOF (Managed Object Format) files
* Configure in conjunction with STIG-ServerStandup
* All DSC \*.ps1 files committed to Version Control (TFS): *$/Enterprise/Enterprise/Systems/Scripts/Powershell/DSC*
* Promote changes same as SDLC
* Test changes prior to commit
* Import validated changes into uDeploy for DML distribution

### Basic Tenets

* Use DSC wherever possible
* Leverage *STIG-ServerStandup.*ps1 “<*DSC*>” XML element and path to MOF file
* Sample XML file: *$/Enterprise/Enterprise/Systems/Scripts/Powershell/sample\_xml/StandupTemplate.xml*
  + (Combines ServerStandup XML with DSC)
* Sample DSC XML file:*$/Enterprise/Enterprise/Systems/Scripts/Powershell/sample\_xml/DSCStandupTemplate.xml*
  + (For use with DSC only driving Standup XML)

#### Existing Server Standup XML

If updating an *existing* XML file:

* 1. Unit test with a single XML file that only uses DSC (example: *DSCStandupTemplate.xml)*
  2. Update the existing XML file with that same DSC element

### Design Process

1. Create scripts and MOF Files on a “toolbox” server; e.g. toolscmapp01
   1. D:\DSC is a directory that’s available for executing the DSC .ps1 files to create the MOF
2. Test .MOF on sandbox server(s); e.g. dsbxscm.\*
   1. Copy the directory/MOF file and execute
3. Create or update the appropriate XML file for *STIG-ServerStandup.ps1*
   1. Use the <DSC> XML element with the path to the MOF file
   2. Example DML MOF path: *\\10.13.0.206\scratch\DML\Microsoft\Powershell\MOF\<MOF\_directory>\<MOF\_file>.mof*
4. Test/Validate the XML file
   1. e.g. *.\STIG-ServerStandup.ps1 -XMLFile <path to XML*> (or) *.\Get-ServerStandupFiles.ps1 -XMLFile <path to XML>*
   2. For additional usage, see [The Bridgepoint Scripting Library](http://insite.bridgepoint.local/collaborate/EASE/_layouts/WordViewer.aspx?id=/collaborate/EASE/Shared%20Documents/WIP%20Documents/Documents/The%20Bridgepoint%20Scripting%20Library.docx&Source=http%3a//insite.bridgepoint.local/collaborate/EASE/Shared%2520Documents/Forms/AllItems.aspx?RootFolder%3D%252Fcollaborate%252FEASE%252FShared%2520Documents%252FWIP%2520Documents%252FDocuments&DefaultItemOpen=1&DefaultItemOpen=1): “Obtaining the library files”
5. Add/update/commit created .PS1 scripts, XML files and MOF directories/files to TFS
   1. Scripts: *$/Enterprise/Enterprise/Systems/Scripts/Powershell/DSC/<script>.ps1*
   2. MOF directories/files: *$/SCM/Scripts/PowerShell/DSC/MOF/<directory>/<MOF\_file>.mof*
   3. XML (Typically) *$/Enterprise/Enterprise/Systems/Scripts/Powershell/WebServices*
6. Sync to DML (via uDeploy)
   1. This will be performed automatically upon commit

Application: [\_udeploy sync\_](https://udapp01:8443/#application/15f48a4b-47e3-4e7e-bea8-6c19b2bf9d1b)

Component responsible: [Rsync\_DesiredStateConfiguration-MOF](https://udapp01:8443/#component/419cd44c-c159-41c9-91e9-e999f1ca7024)

#### Template (Example) DSC Script

A template/example exists with some pre-described examples for DSC

*$/Enterprise/Enterprise/Systems/Scripts/Powershell/DSC/DSC-Template.ps1*

Usage and notes:

Get-Help DSC-Template.ps1

Example:

Get-Help DSC-Template.ps1 -example

# MOF file creation

To create the MOF directory/file:

“dot-source” the .ps1 script, this will load the Configuration into memory as a function

Execute the Configuration with any arguments needed, this will create the MOF

Example usage: DSC-Template.ps1

Note that, in this example, we’re executing two Configurations to create 2 separate MOFs:

*HelloWorldConfig*

*MakeItSo*

PS D:\DSC> . .\DSC-Template.ps1

PS D:\DSC> HelloWorldConfig

Directory: C:\\_TFS\TFS2013\SCM\Scripts\PowerShell\DSC\HelloWorldConfig

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a--- 4/29/2015 1:07 PM 2208 localhost.mof

PS D:\DSC> MakeItSo -ConfigurationData $ConfigurationData -Credential (Get-Credential)

cmdlet Get-Credential at command pipeline position 1

Supply values for the following parameters:

Directory: D:\DSC\MakeItSo

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a--- 4/29/2015 1:08 PM 20014 localhost.mof

PS D:\DSC>

In this case, there’s a ConfigurationData variable that’s defined within the script as $ConfigurationData ; additionally the -Credential is needed for any “Credential = $Credential” properties. Use the “*bridgepoint\svc\_dsc\_ro*” account (password is in [Passwordstate](https://pwds01t/)) for any remote access (e.g. when referencing external DML)

# External References

## Why DSC?

TechEd with the “Father of Powershell” Jeffery Snover:

<http://channel9.msdn.com/Events/TechEd/NorthAmerica/2013/MDC-B302#fbid=?hashlink=fbid>

Pluralsight Blog:

<http://blog.pluralsight.com/desired-state-configuration-powershell-4>

Introduction:

<http://blogs.technet.com/b/privatecloud/archive/2013/08/30/introducing-powershell-desired-state-configuration-dsc.aspx>

## How to DSC

Getting started:

<http://technet.microsoft.com/en-us/library/dn249918.aspx>

Working with:

<http://www.windowsnetworking.com/articles-tutorials/windows-server-2012/working-desired-state-configuration-feature-part1.html>

Building Configurations:

<http://powershell.org/wp/2013/10/08/building-a-desired-state-configuration-configuration/>

### Package resource

Reference: https://technet.microsoft.com/en-us/library/dn282132.aspx

The **Package** resource in Windows PowerShell Desired State Configuration (DSC) provides a mechanism to install or uninstall packages, such as Windows Installer and setup.exe packages, on a target node.

The following required properties ***must*** match the installer’s information:

Name = [string]

ProductId = [string]

The ProductId is the *ProductCode* of the MSI, which you can get by opening the MSI in Orca (part of the Windows SDK) or you can install the get-msitable module from [\\10.13.0.206\scratch\DML\Microsoft\Powershell\PSSnapIn\MSI\psmsi.msi](file:///\\10.13.0.206\scratch\DML\Microsoft\Powershell\PSSnapIn\MSI\psmsi.msi)

To obtain properties::

PS> get-msitable <yourmsi.msi> -table Property | where { $\_.Property -eq "ProductCode" }

get-msitable <yourmsi.msi> -table Property | where { $\_.Property -eq "ProductName" }

Example: *MS Webdeploy MSI in D:\temp\WebDeploy\_amd64\_en-US.msi*

PS H:\> get-msitable -Path C:\temp\WebDeploy\_amd64\_en-US.msi -table Property | where { $\_.Property -eq "ProductCode" }

Property Value

-------- -----

ProductCode {AA72C306-30BE-4BB1-9E42-59552BAD2CDF}

PS H:\> get-msitable -Path C:\temp\WebDeploy\_amd64\_en-US.msi -table Property | where { $\_.Property -eq "ProductName" }

Property Value

-------- -----

ProductName Microsoft Web Deploy 3.0

So in this case, the corresponding DSC would be:

Package WebDeploy

{

Ensure = "Present"

Path = "C:\temp\WebDeploy\_amd64\_en-US.msi"

Name = "Microsoft Web Deploy 3.0"

ProductId = "AA72C306-30BE-4BB1-9E42-59552BAD2CDF"

Arguments = "/q log c:\temp\WebDeploy.log ADDLOCAL=MSDeployFeature,MSDeployAgentFeature,MSDeployUIFeature,DelegationUIFeature,MSDeployWMSVCHandlerFeature"

}

MOF TFS and DML locations