

Configuration Planning and Installation Guide

Windows PowerShell Desired State Configuration Pull Server

**Author**: Michael Greene

**Reviewers:** Ben Gelens, Ravikanth Chaganti, Aleksandar Nikolic

**Published**: April, 2015

**Summary**: This document is intended to include process and extensibility to assist engineers who are preparing for the solution. Details should provide best practices as identified by customers and then validated by the product team to ensure recommendations are future facing and considered stable.

© 2015 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here.

Contents

[Abstract 4](#_Toc417310995)

[Versions of the Windows Management Framework 4](#_Toc417310996)

[Windows PowerShell Desired State Configuration 4](#_Toc417310997)

[Pull server role 5](#_Toc417310998)

[Configuration planning 5](#_Toc417310999)

[Software requirements 5](#_Toc417311000)

[Software downloads 6](#_Toc417311001)

[WMF 6](#_Toc417311002)

[DSC Resource 6](#_Toc417311003)

[Software process planning 6](#_Toc417311004)

[Hardware requirements 7](#_Toc417311005)

[Hardware process planning 7](#_Toc417311006)

[Accounts 7](#_Toc417311007)

[DNS records 7](#_Toc417311008)

[DNS process planning 8](#_Toc417311009)

[Public Key Infrastructure 8](#_Toc417311010)

[PKI process planning 8](#_Toc417311011)

[Choosing an architecture 9](#_Toc417311012)

[Designing for high availability 9](#_Toc417311013)

[Load balancing 9](#_Toc417311014)

[Shared storage 10](#_Toc417311015)

[Shared Storage Process Planning 10](#_Toc417311016)

[Staging configurations and modules on the pull server 10](#_Toc417311017)

[DSC modules 10](#_Toc417311018)

[DSC modules process planning 11](#_Toc417311019)

[DSC configurations 11](#_Toc417311020)

[GUIDs 11](#_Toc417311021)

[Process planning for configurations 12](#_Toc417311022)

[Installation Guide 12](#_Toc417311023)

[Prerequisites 12](#_Toc417311024)

[Installation and configuration scripts 13](#_Toc417311025)

[Pull server deployment example scripts 13](#_Toc417311026)

[Pull server client example scripts 16](#_Toc417311027)

[Additional references, snippets, and examples 17](#_Toc417311028)

[Revision History 18](#_Toc417311029)

# Abstract

This document is designed to provide official guidance for anyone planningfor a Windows PowerShell Desired State Configuration pull server implementation. A pull server is a simple service that should take only minutes to deploy. Although this document will offer technical how-toguidance that can be used in a deployment, the value of this document is as a reference for best practices and *what to think about before deploying*.

Readers should have basic familiarity with DSC, and the terms used to describe the components that are included in a DSC deployment. For more information, see the [Windows PowerShell Desired State Configuration Overview](https://technet.microsoft.com/en-us/library/dn249912.aspx)[[1]](#footnote-2) topic.

As DSC is expected to evolve at cloud cadence, the underlying technology including pull server is also expected to evolve and to introduce new capabilities. This document includes a version table in the appendix that provides references to previous releases and references to future looking solutions to encourage forward-looking designs.

The two major sections of this document:

* Configuration Planning
* Installation Guide

### Versions of the Windows Management Framework

The information in this document is intended to apply to Windows Management Framework 5.0. While WMF 5.0 is not required for deploying and operating a pull server, version 5.0 is the focus of this document.

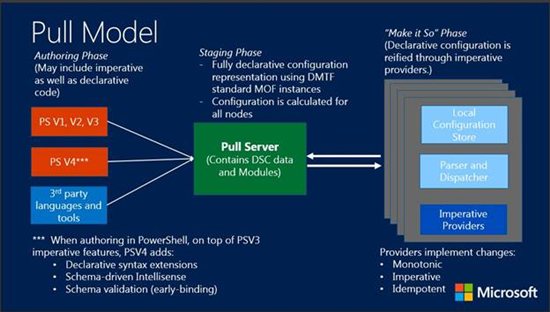
# Windows PowerShell Desired State Configuration

Desired State Configuration (DSC) is a management platform that enables deploying and managing configuration data by using an industry syntax named the Managed Object Format (MOF) to describe the Common Information Model (CIM). An open source project, Open Management Infrastructure (OMI), exists to further development of these standards across platforms including Linux and network hardware operating systems. For more information, see the [DMTF page linking to MOF specifications](http://dmtf.org/standards/cim)[[2]](#footnote-3), and [OMI Documents and Source](https://collaboration.opengroup.org/omi/documents.php)[[3]](#footnote-4).

Windows PowerShell provides a set of language extensions for Desired State Configuration that you can use to create and manage declarative configurations.

## Pull server role

A pull server provides a centralized service to store configurations that will be accessible to target nodes.



The pull server role can be deployed as either a Web Server instance or an SMB file share. The web server capability includes an OData interface and can optionally include capabilities for target nodes to report back confirmation of success or failure as configurations are applied. This functionality is useful in environments where there are a large number of target nodes. After configuring a target node (also referred to as a client) to point to the pull server the latest configuration data and any required scripts are downloaded and applied. This can happen as a one-time deployment or as a re-occurring job which also makes the pull server an important asset for managing change at scale. For more information, see [Windows PowerShell Desired State Configuration Pull Servers](https://technet.microsoft.com/en-us/library/dn249913.aspx)[[4]](#footnote-5) and [Push and Pull Configuration Modes](https://technet.microsoft.com/en-us/library/dn249913.aspx)[[5]](#footnote-6).

# Configuration planning

For any enterprise software deployment there is information that can be collected in advance to help plan for the correct architecture and to be prepared for the steps required to complete the deployment. The following sections provide information regarding how to prepare and the organizational connections that will likely need to happen in advance.

## Software requirements

Deployment of a pull server requires the DSC Service feature of Windows Server. This feature was introduced in Windows Server 2012, and has been updated through ongoing releases of Windows Management Framework (WMF).

### Software downloads

In addition to installing the latest content from Windows Update, there are two downloads considered best practice to deploy a DSC pull server: The latest version of Windows Management Framework, and a DSC module to automate pull server provisioning.

### WMF

Windows Server 2012 R2 includes a feature named the DSC Service. The DSC Service feature provides the pull server functionality, including the binaries that support the OData endpoint. WMF is included in Windows Server and is updated on an agile cadence between Windows Server releases. [New versions of WMF 5.0](http://aka.ms/wmf5latest)[[6]](#footnote-7) can include updates to the DSC Service feature. For this reason, it is a best practice to download the latest release of WMF and to review the release notes to determine if the release includes an update to the DSC service feature. You should also review the section of the release notes that indicates whether the design status for an update or scenario is listed as stable or experimental[[7]](#footnote-8). To allow for an agile release cycle, individual features can be declared stable, which indicates the feature is ready to be used in a production environment even while WMF is released in preview.

Other features that have historically been updated by WMF releases (see the WMF Release Notes for further detail):

* Windows PowerShell
* Windows PowerShell Integrated Scripting Environment (ISE)
* Windows PowerShell Web Services (Management OData IIS Extension)
* Windows PowerShell Desired State Configuration (DSC)
* Windows Remote Management (WinRM)
* Windows Management Instrumentation (WMI)

### DSC resource

A pull server deployment can be simplified by provisioning the service using a DSC configuration script. This document includes configuration scripts that can be used to deploy a production ready server node. To use the configuration scripts, a DSC module is required that is not included in Windows Server. The required module name is **xPSDesiredStateConfiguration**, which includes the DSC resource **xDscWebService**. The xPSDesiredStateConfiguration module can be downloaded [here](https://gallery.technet.microsoft.com/xPSDesiredStateConfiguratio-417dc71d)[[8]](#footnote-9).

Use the **Install-Module** cmdlet from the **PowerShellGet** module.

|  |
| --- |
| Install-Module xPSDesiredStateConfiguration |

The **PowerShellGet** module will download the module to C:\Program Files\Windows PowerShell\Modules.

### Software process planning

|  |
| --- |
| Planning task |
| Do you have access to the installation files for Windows Server 2012 R2? |
| Will the deployment environment have Internet access to download WMF and the module from the online gallery? |
| How will you install the latest security updates after installing the operating system? |
| Will the environment have Internet access to obtain updates, or will it have a local Windows Server Update Services (WSUS) server? |
| Do you have access to Windows Server installation files that already include updates through offline injection? |

## Hardware requirements

Pull server deployments are supported on both physical and virtual servers. The sizing requirements for pull server align with the requirements for Windows Server 2012 R2.

|  |
| --- |
| CPU: 1.4 GHz 64-bit processor  Memory: 512 MB  Disk Space: 32 GB  Network: Gigabit Ethernet Adapter |

### Hardware process planning

|  |
| --- |
| Planning task |
| Will you deploy on physical hardware or on a virtualization platform? |
| What is the process to request a new server for your target environment? |
| What is the average turnaround time for a server to become available? |
| What size server will you request? |

## Accounts

There are no service account requirements to deploy a pull server instance. However there are scenarios where the website could run in the context of a local user account. As an example, if there is a need to access a storage share for website content and either the Windows Server or the device hosting the storage share are not domain joined.

## DNS records

You will need a server name to use when configuring clients to work with a pull server environment. In test environments, typically the server hostname is used, or the IP address for the server can be used if DNS name resolution is not available. In production environments or in a lab environment that is intended to represent a production deployment, the best practice is to create a DNS CNAME record.

A DNS CNAME allows you to create an alias to refer to your host (A) record. The intent of the additional name record is to increase flexibility should a change be required in the future. A CNAME can help to isolate the client configuration so that changes to the server environment, such as replacing a pull server or adding additional pull servers, will not require a corresponding change to the client configuration.

When choosing a name for the DNS record, keep the solution architecture in mind. If using load balancing, the certificate used to secure traffic over HTTPS will need to share the same name as the DNS record. Similarly, if using a highly available file share the virtual name for the cluster would be used.

|  |  |
| --- | --- |
| Scenario | Best Practice |
| Test Environment | Reproduce the planned production environment, if possible. A server hostname is suitable for simple configurations. If DNS is not available, an IP address may be used in lieu of a hostname. |
| Single Node Deployment | Create a DNS CNAME record that points to the server hostname. |
| Highly Available Deployment | If clients will connect through a load balancing solution, create a hostname for the virtual IP and a CNAME record that references that hostname. If DNS round robin will be used to distribute client requests across pull servers, you must configure the name records to include the host names of all deployed pull server instances. |

For more information, see [Configuring DNS Round Robin in Windows Server](https://technet.microsoft.com/en-us/library/cc787484(v=ws.10).aspx)[[9]](#footnote-10).

### DNS process planning

|  |
| --- |
| Planning task |
| Do you know who to contact to have DNS records created and changed? |
| What is the average turnaround for a request for a DNS record? |
| Do you need to request static Hostname (A) records for servers? |
| What will you request as a CNAME? |
| If needed, what type of Load Balancing solution will you utilize? (see section titled Load Balancing for details) |

## Public Key Infrastructure

Most organizations today require that network traffic, especially traffic that includes such sensitive data as how servers are configured, must be validated and/or encrypted during transit. While it is possible to deploy a pull server using HTTP which facilitates client requests in clear text, it is a best practice to secure traffic using HTTPS. The service can be configured to use HTTPS using a set of parameters in the DSC resource **xPSDesiredStateConfiguration**.

The certificate requirements to secure HTTPS traffic for pull server are not different than securing any other HTTPS web site. The **Web Server** template in a Windows Server Certificate Services satisfies the required capabilities.

### PKI process planning

|  |
| --- |
| Planning task |
| If certificate requests are not automated, who will you need to contact to requests a certificate? |
| What is the average turnaround for the request? |
| How will the certificate file be transferred to you? |
| How will the certificate private key be transferred to you? |
| How long is the default expiration time? |
| Have you settled on a DNS name for the pull server environment, that you can use for the certificate name? |

# Choosing an architecture

A pull server can be deployed using either a web service hosted on IIS, or an SMB file share. In most situations, the web service option will provide greater flexibility. It is not uncommon for HTTPS traffic to traverse network boundaries, whereas SMB traffic is often filtered or blocked between networks. The web service also offers the option to include a Conformance Server or Web Reporting Manager (both topics to be addressed in a future version of this document) that provide a mechanism for clients to report status back to a server for centralized visibility. SMB provides an option for environments where policy dictates that a web server should not be utilized, and for other environmental requirements that make a web server role undesirable. In either case, remember to evaluate the requirements for signing and encrypting traffic. HTTPS, SMB signing, and IPSEC policies are all options worth considering.

### Designing for high availability

The pull server role can be deployed in a highly available architecture. The web service role can be load balanced and the files and folders that include DSC modules and DSC configurations can be located on highly available storage.

Bear in mind that once configurations and modules are delivered to a target node all data required to perform tests and to set configurations is stored locally on each node. Only changes are delivered from the pull server. A service outage for a pull server would not be an interruption unless deployments are active. Typically, high availability is only warranted for the largest of environments.

Configuring a highly available pull server environment requires decisions about how to distribute client requests across multiple server nodes, and how to share the required server files across those nodes.

## Load balancing

Clients interacting with the web service make a request for information that is returned in a single response. No sequential requests are required, so it is not necessary for the load balancing platform to ensure sessions are maintained on a single server at any point in time.  
Load balancing process planning

|  |
| --- |
| Planning task |
| What solution will be used for load balancing traffic across servers? |
| If using a hardware load balancer, who will take a request to add a new configuration to the device? |
| What is the average turnaround for a request to configure a new load balanced web service? |
| What information will be required for the request? |
| Will you need to request an additional IP or will the team responsible for load balancing handle that? |
| Do you have the DNS records needed, and will this be required by the team responsible for configuring the load balancing solution? |
| Does the load balancing solution require that PKI be handled by the device or can it load balance HTTPS traffic as long as there are no session requirements? |

## Shared storage

In a highly available scenario where multiple servers are configured as pull servers and connections are load balanced across them, it is critical that the resources and configurations available from those servers be identical. The best way to accomplish this is to store this content on a highly available location such as a clustered file share. The location of the share can be specified in the configuration for each server.For more information about shared storage options, see [Scale-Out File Server for Application Data Overview](https://technet.microsoft.com/en-us/library/hh831349.aspx)[[10]](#footnote-11).

### Shared Storage Process Planning

|  |
| --- |
| Planning task |
| What solution will be used to host the highly available share? |
| Who will handle the request for a new highly available share? |
| What is the average turnaround time for a highly available share to be available? |
| What information will the teams responsible for storage and/or clustering need? |

# Staging configurations and modules on the pull server

As part of configuration planning, you will need to think about which DSC modules and configurations will be hosted by the pull server. For the purpose of configuration planning it is important to have a basic understanding of how to prepare and deploy content to a pull server.

In the future, this section will be expanded and included in an Operations Guide for DSC Pull Server. The guide will discuss the day to day process for managing modules and configurations over time with automation.

## DSC modules

Clients that request a configuration will need the required DSC modules. A functionality of the pull server is to automate distribution on demand of DSC modules to clients. If you are deploying a pull server for the first time, perhaps as a lab or proof of concept, you are likely going to depend on DSC modules that are available from public repositories such as the PowerShell Gallery or the PowerShell.org GitHub repositories for DSC modules.

It is critical to remember that even for trusted online sources such as the PowerShell Gallery, any module that is downloaded from a public repository should be reviewed by someone with PowerShell experience and knowledge of the environment where the modules will be used prior to being used in production. While completing this task it is a good time to check for any additional payload in the module that can be removed such as documentation and example scripts. This will reduce the network bandwidth per client in their first request, when modules will be downloaded over the network from server to client.

Each module must be packaged in a specific format, a ZIP file named ModuleName\_Version.zip that contains the module payload. After the file is copied to the server a checksum file must be created. When clients connect to the server, the checksum is used to verify the content of the DSC module has not changed since it was published.

|  |
| --- |
| New-DscCheckSum -ConfigurationPath .\ -OutPath .\ |

### DSC modules process planning

|  |
| --- |
| Planning task |
| If you are planning a test or lab environment which scenarios are key to validate? |
| Are there publicly available modules that contain resources to cover everything you need or will you need to author your own resources? |
| Will your environment have Internet access to retrieve public modules? |
| Who will be responsible for reviewing DSC modules? |
| If you are planning a production environment what will you use as a local repository for storing DSC modules? |
| Will a central team accept DSC modules from application teams? What will the process be? |
| Will you automate packaging, copying, and creating a checksum for production-ready DSC modules to the server, from your source repo? |
| Will your team be responsible for managing the automation platform as well? |

## DSC configurations

The purpose of a pull server is to provide a centralized mechanism for distributing DSC configurations to client nodes. The configurations are stored on the server as MOF documents. Each document will be named with a unique GUID. When clients are configured to connect with a pull server, they are also given the GUID for the configuration they should request. This system of referencing configurations by GUID guarantees global uniqueness and is flexible such that a configuration can be applied with granularity per node, or as a role configuration that spans many servers that should have identical configurations.

### GUIDs

Planning for configuration GUIDs is worth additional attention when thinking through a pull server deployment. There is no specific requirement for how to handle GUIDs and the process is likely to be unique for each environment. The process can range from simple to complex: a centrally stored CSV file, a simple SQL table, a CMDB, or a complex solution requiring integration with another tool or software solution. There are two general approaches:

**Assigning GUIDs per server —** Provides a measure of assurance that every server configuration is controlled individually. This provides a level of precision around updates and can work well in environments with few servers.

**Assigning GUIDs per server role —** All servers that perform the same function, such as web servers, use the same GUID to reference the required configuration data. Be aware that if many servers share the same GUID, all of them would be updated simultaneously when the configuration changes.

The GUID is something that should be considered sensitive data because it could be leveraged by someone with malicious intent to gain intelligence about how servers are deployed and configured in your environment. For more information, see [Securely allocating GUIDs in PowerShell Desired State Configuration Pull Mode](http://blogs.msdn.com/b/powershell/archive/2014/12/31/securely-allocating-guids-in-powershell-desired-state-configuration-pull-mode.aspx)[[11]](#footnote-12).

### Process planning for configurations

|  |
| --- |
| Planning task |
| Who will be responsible for copying configurations in to the pull server folder when they are ready? |
| If Configurations are authored by an application team, what will the process be to hand them off? |
| Will you leverage a repository to store configurations as they are being authored, across teams? |
| Will you automate the process of copying configurations to the server and creating a checksum when they are ready? |
| How will you map GUIDs to servers or roles, and where will this be stored? |
| What will you use as a process to configure client machines, and how will it integrate with your process for creating and storing Configuration GUIDs? |

# Installation Guide

**Important** Scripts given in this document are stable examples. Always review scripts carefully before executing them in a production environment.

## Prerequisites

To verify the version of PowerShell on your server use the following command.

|  |
| --- |
| $PSVersionTable.PSVersion |

If possible, upgrade to the latest version of Windows Management Framework.

Next, download the xPsDesiredStateConfiguration module using the following command.

|  |
| --- |
| Install-Module xPSDesiredStateConfiguration |

The command will ask for your approval before downloading the module.

## Installation and configuration scripts

The best method to deploy a DSC pull server is to use a DSC configuration script. This document will present scripts including both basic settings that would configure only the DSC web service and advanced settings that would configure a Windows Server end-to-end including DSC web service.

### Pull server deployment example scripts

Note: Currently the xPSDesiredStateConfiguation DSC module requires the server to be EN-US locale.

#### Basic configuration for Windows Server 2012

|  |
| --- |
| # This is a very basic Configuration to deploy a pull server instance in a lab environment on Windows Server 2012.  Configuration PullServer {  Import-DscResource -ModuleName xPSDesiredStateConfiguration    # Load the Windows Server DSC Service feature  WindowsFeature DSCServiceFeature  {  Ensure = 'Present'  Name = 'DSC-Service'  }  # Use the DSC Resource to simplify deployment of the web service  xDSCWebService PSDSCPullServer  {  Ensure = 'Present'  EndpointName = 'PSDSCPullServer'  Port = 8080  PhysicalPath = "$env:SYSTEMDRIVE\inetpub\wwwroot\PSDSCPullServer"  CertificateThumbPrint = 'AllowUnencryptedTraffic'  ModulePath = "$env:PROGRAMFILES\WindowsPowerShell\DscService\Modules"  ConfigurationPath = "$env:PROGRAMFILES\WindowsPowerShell\DscService\Configuration"  State = 'Started'  DependsOn = '[WindowsFeature]DSCServiceFeature'  }  }  PullServer -OutputPath 'C:\PullServerConfig\'  Start-DscConfiguration -Wait -Force -Verbose -Path 'C:\PullServerConfig\' |

#### Advanced configuration for Windows Server 2012 R2 (script contents)

|  |
| --- |
| # This is an advanced Configuration example for Pull Server production deployments on Windows Server 2012 R2.  # Many of the features demonstrated are optional and provided to demonstrate how to adapt the Configuration for multiple scenarios  # Select the needed resources based on the requirements for each environment.  # Optional scenarios include:  # \* Reduce footprint to Server Core  # \* Rename server and join domain  # \* Switch from SSL to TLS for HTTPS  # \* Automatically load certificate from Certificate Authority  # \* Locate Modules and Configuration data on remote SMB share  # \* Manage state of default websites in IIS  param (  [Parameter(Mandatory=$true)]  [ValidateNotNullorEmpty()]  [System.String] $ServerName,  [System.String] $DomainName,  [System.String] $CARootName,  [System.String] $CAServerFQDN,  [System.String] $CertSubject,  [System.String] $SMBShare,  [Parameter(Mandatory=$true)]  [ValidateNotNullorEmpty()]  [PsCredential] $Credential  )  Configuration PullServer {  Import-DscResource -ModuleName xPSDesiredStateConfiguration, xWebAdministration, xCertificate, xComputerManagement  Node localhost  {    # Configure the server to automatically corret configuration drift including reboots if needed.  LocalConfigurationManager  {  ConfigurationMode = 'ApplyAndAutoCorrect'  RebootNodeifNeeded = $node.RebootNodeifNeeded  CertificateId = $node.Thumbprint  }  # Remove all GUI interfaces so the server has minimum running footprint.  WindowsFeature ServerCore  {  Ensure = 'Absent'  Name = 'User-Interfaces-Infra'  }  # Set the server name and if needed, join a domain. If not joining a domain, remove the DomainName parameter.  xComputer DomainJoin  {  Name = $Node.ServerName  DomainName = $Node.DomainName  Credential = $Node.Credential  }  # The next series of settings disable IIS and enable TLS, for environments where that is required by policy.  Registry TLS1\_2ServerEnabled  {  Ensure = 'Present'  Key = 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server'  ValueName = 'Enabled'  ValueData = 1  ValueType = 'Dword'  }  Registry TLS1\_2ServerDisabledByDefault  {  Ensure = 'Present'  Key = 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server'  ValueName = 'DisabledByDefault'  ValueData = 0  ValueType = 'Dword'  }  Registry TLS1\_2ClientEnabled  {  Ensure = 'Present'  Key = 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client'  ValueName = 'Enabled'  ValueData = 1  ValueType = 'Dword'  }  Registry TLS1\_2ClientDisabledByDefault  {  Ensure = 'Present'  Key = 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client'  ValueName = 'DisabledByDefault'  ValueData = 0  ValueType = 'Dword'  }  Registry SSL2ServerDisabled  {  Ensure = 'Present'  Key = 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Server'  ValueName = 'Enabled'  ValueData = 0  ValueType = 'Dword'  }  # Install the Windows Server DSC Service feature  WindowsFeature DSCServiceFeature  {  Ensure = 'Present'  Name = 'DSC-Service'  }  # If using a certificate from a local Active Directory Enterprise Root Certificate Authority, complete a request and install the certificate  xCertReq SSLCert  {  CARootName = $Node.CARootName  CAServerFQDN = $Node.CAServerFQDN  Subject = $Node.CertSubject  AutoRenew = $Node.AutoRenew  Credential = $Node.Credential  }  # Use the DSC resource to simplify deployment of the web service. You might also consider modifying the default port, possibly leveraging port 443 in environments where that is enforced as a standard.  xDSCWebService PSDSCPullServer  {  Ensure = 'Present'  EndpointName = 'PSDSCPullServer'  Port = 8080  PhysicalPath = "$env:SYSTEMDRIVE\inetpub\wwwroot\PSDSCPullServer"  CertificateThumbPrint = 'CertificateSubject'  CertificateSubject = $Node.CertSubject  ModulePath = "$($Node.SMBShare)\DscService\Modules"  ConfigurationPath = "$($Node.SMBShare)\DscService\Configuration"  State = 'Started'  DependsOn = '[WindowsFeature]DSCServiceFeature'  }  # Validate web config file contains current DB settings  xWebConfigKeyValue CorrectDBProvider  {  ConfigSection = 'AppSettings'  Key = 'dbprovider'  Value = 'System.Data.OleDb'  WebsitePath = 'IIS:\sites\PSDSCPullServer'  DependsOn = '[xDSCWebService]PSDSCPullServer'  }  xWebConfigKeyValue CorrectDBConnectionStr  {  ConfigSection = 'AppSettings'  Key = 'dbconnectionstr'  Value = 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program Files\WindowsPowerShell\DscService\Devices.mdb;'  WebsitePath = 'IIS:\sites\PSDSCPullServer'  DependsOn = '[xDSCWebService]PSDSCPullServer'  }  # Stop the default website  xWebsite StopDefaultSite  {  Ensure = 'Present'  Name = 'Default Web Site'  State = 'Stopped'  PhysicalPath = 'C:\inetpub\wwwroot'  DependsOn = '[WindowsFeature]DSCServiceFeature'  }  }  }  $configData = @{  AllNodes = @(  @{  NodeName = 'localhost'  ServerName = $ServerName  DomainName = $DomainName  CARootName = $CARootName  CAServerFQDN = $CAServerFQDN  CertSubject = $CertSubject  AutoRenew = $true  SMBShare = $SMBShare  Credential = $Credential  RebootNodeifNeeded = $true  CertificateFile = 'c:\PullServerConfig\Cert.cer'  Thumbprint = 'B9A39921918B466EB1ADF2509E00F5DECB2EFDA9'  }  )  }  PullServer -ConfigurationData $configData -OutputPath 'C:\PullServerConfig\'  Set-DscLocalConfigurationManager -ComputerName localhost -Path 'C:\PullServerConfig\'  Start-DscConfiguration -Wait -Force -Verbose -Path 'C:\PullServerConfig\'  # .\Script.ps1 -ServerName web1 -domainname 'test.pha' -carootname 'test-dc01-ca' -caserverfqdn 'dc01.test.pha' -certsubject 'CN=service.test.pha' -smbshare '\\sofs1.test.pha\share' |

#### Verify pull server functionality

|  |
| --- |
| # This function is meant to simplify a check against a DSC pull server. If you do not use the default service URL, you will need to adjust accordingly.  function Verify-DSCPullServer ($fqdn) {  ([xml](invoke-webrequest "https://$($fqdn):8080/psdscpullserver.svc" | % Content)).service.workspace.collection.href  }  Verify-DSCPullServer 'INSERT SERVER FQDN'  Expected Result:  Action  Module  StatusReport  Node |

### Pull server client example scripts

#### Configure clients

|  |
| --- |
| Configuration PullClient {  param(  $ID,  $Server  )  LocalConfigurationManager  {  ConfigurationID = $ID;  RefreshMode = 'PULL';  DownloadManagerName = 'WebDownloadManager';  RebootNodeIfNeeded = $true;  RefreshFrequencyMins = 30;  ConfigurationModeFrequencyMins = 15;  ConfigurationMode = 'ApplyAndAutoCorrect';  DownloadManagerCustomData = @{ServerUrl = "http://"+$Server+":8080/PSDSCPullServer.svc"; AllowUnsecureConnection = $true}  }  }  PullClient -ID 'INSERTGUID' -Server 'INSERTSERVER' -Output 'C:\DSCConfig\'  Set-DscLocalConfigurationManager -ComputerName 'Localhost' -Path 'C:\DSCConfig\' -Verbose |

# Additional references, snippets, and examples

This example shows how to manually initiate a client connection (requires WMF5) for testing.

|  |
| --- |
| Update-DSCConfiguration –Wait -Verbose |

The [Add-DnsServerResourceRecordName](https://technet.microsoft.com/en-us/library/jj649894.aspx) (http://bit.ly/1G1H31L) cmdlet is used to add a type CNAME record to a DNS zone.

The [PowerShell Function to Create a Checksum and Publish DSC MOF to SMB Pull Server](https://gallery.technet.microsoft.com/scriptcenter/PowerShell-Function-to-3bc4b7f0) (http://bit.ly/1E46BhI) automatically generates the required checksum, and then copies both the MOF configuration and checksum files to the SMB pull server.

# Revision history

|  |  |  |
| --- | --- | --- |
| Publication Date | Version | Comments |
| 4/10/2015 | 0.1 | Initial Draft Review |
| 4/14/2015 | 0.13 | Draft including first round of feedback from Ben and Ravi |
| 4/15/205 | 0.14 | Draft including feedback from Keith Bankston |
| 4/17/2015 | 0.15 | Draft including customer feedback |
| 4/20/2015 | 0.16 | Final draft include feedback from Aleksander |
| 4/20/2015 | 1.0 | Release 1.0 |

1. http://bit.ly/1HoS4yp [↑](#footnote-ref-2)
2. http://bit.ly/1F8XrS1 [↑](#footnote-ref-3)
3. http://bit.ly/1aLTMwQ [↑](#footnote-ref-4)
4. http://bit.ly/1beafuk [↑](#footnote-ref-5)
5. http://bit.ly/19nrDLN [↑](#footnote-ref-6)
6. http://aka.ms/wmf5latest [↑](#footnote-ref-7)
7. [Defining Stable and Experimental](http://blogs.msdn.com/b/powershell/archive/2014/12/10/wmf-5-0-preview-defining-quot-experimental-designs-quot-and-quot-stable-designs-quot.aspx) (http://bit.ly/1IyPO6b) [↑](#footnote-ref-8)
8. http://bit.ly/1EsplW9 [↑](#footnote-ref-9)
9. http://bit.ly/15D7SPp [↑](#footnote-ref-10)
10. http://bit.ly/1DFtNkV [↑](#footnote-ref-11)
11. http://bit.ly/1HaAMWZ [↑](#footnote-ref-12)