**xWebAdministration Module – Windows PowerShell Desired State Configuration Resource Kit**

# **Introduction**

The **xWebAdministration** module is a part of Windows PowerShell Desired State Configuration (DSC) Resource Kit, which is a collection of DSC Resources produced by the PowerShell Team. This module contains the **xWebsite and xIisModule** resource. These DSC Resources allow configuration of IIS Website.

**All of the resources in the DSC Resource Kit are provided AS IS, and are not supported through any Microsoft standard support program or service. The “x” in xWebAdministration stands for experimental**, which means that these resources will be **fix forward** and monitored by the module owner(s).

Please leave comments, feature requests, and bug reports in the Q & A tab for this module.

If you would like to modify **xWebAdministraion** module**,** feel free. When modifying, please update the module name, resource friendly name, and MOF class name (instructions below). As specified in the license, you may copy or modify this resource as long as they are used on the Windows Platform.

For more information about Windows PowerShell Desired State Configuration, check out the blog posts on the [PowerShell Blog](http://blogs.msdn.com/b/powershell/) ([this](http://blogs.msdn.com/b/powershell/archive/2013/11/01/configuration-in-a-devops-world-windows-powershell-desired-state-configuration.aspx) is a good starting point). There are also great community resources, such as [PowerShell.org](http://powershell.org/wp/tag/dsc/), or [PowerShell Magazine](http://www.powershellmagazine.com/tag/dsc/). For more information on the DSC Resource Kit, check out [this blog post](http://go.microsoft.com/fwlink/?LinkID=389546).

# Installation

To install **xWebAdministration** module

* Unzip the content under $env:ProgramFiles\WindowsPowerShell\Modules folder

**To confirm installation:**

* **Run Get-DSCResource to see that xWebsite is among the DSC Resources listed**

# **Requirements**

This module requires the latest version of PowerShell (v4.0, which ships in Windows 8.1 or Windows Server 2012R2). It also requires IIS features. To easily use PowerShell 4.0 on older operating systems, [install WMF 4.0](http://www.microsoft.com/en-us/download/details.aspx?id=40855). Please read the installation instructions that are present on both the download page and the release notes for WMF 4.0.

# **Description**

# The **xWebsiteAdministration** module contains the **xWebsite** DSC Resource. This DSC Resource allows for simple configuration of IIS Websites. Using this resource, you can define your website’s state, application pool, binding info, and other characteristics. When used with the Windows Feature and File Resources (that ships in Windows), this resource allows you to set up a web server entirely through DSC.

# The **xWebsiteAdministration** module contains the **xIisModule** DSC Resource. This DSC Resource allows for registration of modules, such as FastCgiModules, with IIS. Using this resource, you can define your where you module is, the paths and verbs allowed.

# Details

**xWebsite** resource has following properties:

* **Name:** The desired name of the website
* **PhysicalPath:** The path of the files that compose the website
* **State:**  State of the website – started or stopped
* **Protocol:**  Web protocol (currently only “http” is supported)
* **BindingInfo**: Binding information to match the above protocol
* **ApplicationPool:** The website’s application pool
* **Ensure:** Should website be present or absent
* **DefaultPage:** On array of default page(s) for the site.

# Details

**xIisModule** resource has following properties:

* **Path:** The path to the module to be registered.
* **Name:** The logical name to register the module as in IIS.
* **RequestPath:** The allowed request paths, such as \*.php.
* **Verb:**  An array of allowed verbs, such as get and post.
* **SiteName**: The name of the Site to register the module for. If empty, the resource will register the module with all of IIS.
* **ModuleType:** The type of the module. Currently, only FastCgiModule is supported.
* **Ensure:** Should module be present or absent

# **Example: Stopping the default website**

When configuring a new IIS Server, several references recommend removing or stopping the default website for security purposes. This example sets up your IIS webserver by installing IIS Windows Feature. Following that, it will stop the default website by setting “State = Stopped ”.

configuration Sample\_xWebsite\_StopDefault

{

param

(

# Target nodes to apply the configuration

[string[]]$NodeName = 'localhost'

)

# Import the module that defines custom resources

Import-DscResource -Module xWebAdministration

Node $NodeName

{

# Install the IIS role

WindowsFeature IIS

{

Ensure = "Present"

Name = "Web-Server"

}

# Stop the default website

xWebsite DefaultSite

{

Ensure = "Present"

Name = "Default Web Site"

State = "Stopped"

PhysicalPath = "C:\inetpub\wwwroot"

DependsOn = "[WindowsFeature]IIS"

}

}

}

# **Example: Create a new website**

While setting up IIS and stopping the default website is interesting, it isn’t quite useful yet. After all, typically people use IIS to set up websites of your own. Fortunately, using DSC, adding another website is as simple as using the File and xWebsite resources to copy the website content and configure the website with a default page.

configuration Sample\_xWebsite\_NewWebsite

{

param

(

# Target nodes to apply the configuration

[string[]]$NodeName = 'localhost',

# Name of the website to create

[Parameter(Mandatory)]

[ValidateNotNullOrEmpty()]

[String]$WebSiteName,

# Source Path for Website content

[Parameter(Mandatory)]

[ValidateNotNullOrEmpty()]

[String]$SourcePath,

# Destination path for Website content

[Parameter(Mandatory)]

[ValidateNotNullOrEmpty()]

[String]$DestinationPath

)

# Import the module that defines custom resources

Import-DscResource -Module xWebAdministration

Node $NodeName

{

# Install the IIS role

WindowsFeature IIS

{

Ensure = "Present"

Name = "Web-Server"

}

# Install the ASP .NET 4.5 role

WindowsFeature AspNet45

{

Ensure = "Present"

Name = "Web-Asp-Net45"

}

# Stop the default website

xWebsite DefaultSite

{

Ensure = "Present"

Name = "Default Web Site"

State = "Stopped"

PhysicalPath = "C:\inetpub\wwwroot"

DependsOn = "[WindowsFeature]IIS"

}

# Copy the website content

File WebContent

{

Ensure = "Present"

SourcePath = $SourcePath

DestinationPath = $DestinationPath

Recurse = $true

Type = "Directory"

DependsOn = "[WindowsFeature]AspNet45"

}

# Create the new Website

xWebsite NewWebsite

{

Ensure = "Present"

Name = $WebSiteName

State = "Started"

PhysicalPath = $DestinationPath

DefaultPage = "Default.aspx"

DependsOn = "[File]WebContent"

}

}

}

# **Example: Removing the default website**

In this example, we’ve moved the parameters used to generate the website into a configuration data file – all of the variant portions of the configuration are stored in a separate file. This can be a powerful tool when using DSC to configure a project that will be deployed to multiple environments. For example, users managing larger environments may want to test their configuration on a small number of machines before deploying it across many more machines in their production environment.

Configuration files are made with this in mind. This is an example configuration data file (saved as a .psd1).

configuration Sample\_xWebsite\_FromConfigurationData

{

# Import the module that defines custom resources

Import-DscResource -Module xWebAdministration

# Dynamically find the applicable nodes from configuration data

Node $AllNodes.where{$\_.Role -eq "Web"}.NodeName

{

# Install the IIS role

WindowsFeature IIS

{

Ensure = "Present"

Name = "Web-Server"

}

# Install the ASP .NET 4.5 role

WindowsFeature AspNet45

{

Ensure = "Present"

Name = "Web-Asp-Net45"

}

# Stop an existing website (set up in Sample\_xWebsite\_Default)

xWebsite DefaultSite

{

Ensure = "Present"

Name = "Default Web Site"

State = "Stopped"

PhysicalPath = $Node.DefaultWebSitePath

DependsOn = "[WindowsFeature]IIS"

}

# Copy the website content

File WebContent

{

Ensure = "Present"

SourcePath = $Node.SourcePath

DestinationPath = $Node.DestinationPath

Recurse = $true

Type = "Directory"

DependsOn = "[WindowsFeature]AspNet45"

}

# Create a new website

xWebsite BakeryWebSite

{

Ensure = "Present"

Name = $Node.WebsiteName

State = "Started"

PhysicalPath = $Node.DestinationPath

DependsOn = "[File]WebContent"

}

}

}

Content of configuration data file (e.g. ConfigurationData.psd1) could be:

# Hashtable to define the environmental data

@{

# Node specific data

AllNodes = @(

# All the WebServer has following identical information

@{

NodeName = "\*"

WebsiteName = "FourthCoffee"

SourcePath = "C:\BakeryWebsite\"

DestinationPath = "C:\inetpub\FourthCoffee"

DefaultWebSitePath = "C:\inetpub\wwwroot"

},

@{

NodeName = "WebServer1.fourthcoffee.com"

Role = "Web"

},

@{

NodeName = "WebServer2.fourthcoffee.com"

Role = "Web"

}

);

}

Pass the configuration data to configuration as follows:

Sample\_xWebsite\_FromConfigurationData -ConfigurationData ConfigurationData.psd1

# **Example: Registering Php**

When configuring an IIS Application that uses PHP, you first need to register the PHP CGI module with IIS. The following xPhp configuration downloads and installs the prerequisites for PHP, downloads PHP, registers the PHP CGI module with IIS and sets the system environment variable that PHP needs to run.

Note: this sample is intended to be used as a composite resource, so it does not use Configuration Data. Please see the [Composite Configuration Blog](http://blogs.msdn.com/b/powershell/archive/2014/02/25/reusing-existing-configuration-scripts-in-powershell-desired-state-configuration.aspx) on how to use this in configuration in another configuration.

<sample from //depot/fbl\_srv2\_ci\_mgmt/admintestdata/REDIST/monad/PSArtifactSharing/Modules/DSCPack/xPhp/DscResources/xPhp/xPhp.Schema>

# Composite configuration to install the IIS pre-requisites for php

Configuration IisPreReqs\_php

{

param

(

[Parameter(Mandatory = $true)]

[Validateset("Present","Absent")]

[String]

$Ensure

)

foreach ($Feature in @("Web-Server","Web-Mgmt-Tools","web-Default-Doc", `

"Web-Dir-Browsing","Web-Http-Errors","Web-Static-Content",`

"Web-Http-Logging","web-Stat-Compression","web-Filtering",`

"web-CGI","web-ISAPI-Ext","web-ISAPI-Filter"))

{

WindowsFeature "$Feature$Number"

{

Ensure = $Ensure

Name = $Feature

}

}

}

# Composite configuration to install PHP on IIS

configuration xPhp

{

param(

[Parameter(Mandatory = $true)]

[switch] $installMySqlExt,

[Parameter(Mandatory = $true)]

[string] $PackageFolder,

[Parameter(Mandatory = $true)]

[string] $DownloadUri,

[Parameter(Mandatory = $true)]

[string] $Vc2012RedistDownloadUri,

[Parameter(Mandatory = $true)]

[String] $DestinationPath,

[Parameter(Mandatory = $true)]

[string] $ConfigurationPath

)

# Make sure the IIS Prerequisites for PHP are present

IisPreReqs\_php Iis

{

Ensure = "Present"

# Removed because this dependency does not work in

# Windows Server 2012 R2 and below

# This should work in WMF v5 and above

# DependsOn = "[File]PackagesFolder"

}

# Download and install Visual C Redist2012 from chocolatey.org

Package vcRedist

{

Path = $Vc2012RedistDownloadUri

ProductId = "{CF2BEA3C-26EA-32F8-AA9B-331F7E34BA97}"

Name = "Microsoft Visual C++ 2012 x64 Minimum Runtime - 11.0.61030"

Arguments = "/install /passive /norestart"

}

$phpZip = Join-Path $PackageFolder "php.zip"

# Make sure the PHP archine is in the package folder

xRemoteFile phpArchive

{

uri = $DownloadURI

DestinationPath = $phpZip

}

# Make sure the content of the PHP archine are in the PHP path

Archive php

{

Path = $phpZip

Destination = $DestinationPath

}

if ($installMySqlExt )

{

# Make sure the MySql extention for PHP is in the main PHP path

File phpMySqlExt

{

SourcePath = "$($DestinationPath)\ext\php\_mysql.dll"

DestinationPath = "$($DestinationPath)\php\_mysql.dll"

Ensure = "Present"

DependsOn = @("[Archive]PHP")

MatchSource = $true

}

}

# Make sure the php.ini is in the Php folder

File PhpIni

{

SourcePath = $ConfigurationPath

DestinationPath = "$($DestinationPath)\php.ini"

DependsOn = @("[Archive]PHP")

MatchSource = $true

}

# Make sure the php cgi module is registered with IIS

xIisModule phpHandler

{

Name = "phpFastCgi"

Path = "$($DestinationPath)\php-cgi.exe"

RequestPath = "\*.php"

Verb = "\*"

Ensure = "Present"

DependsOn = @("[Package]vcRedist","[File]PhpIni")

# Removed because this dependency does not work in

# Windows Server 2012 R2 and below

# This should work in WMF v5 and above

# "[IisPreReqs\_php]Iis"

}

# Make sure the php binary folder is in the path

Environment PathPhp

{

Name = "Path"

Value = ";$($DestinationPath)"

Ensure = "Present"

Path = $true

DependsOn = "[Archive]PHP"

}

}

xPhp -PackageFolder "C:\packages" `

-DownloadUri -DownloadUri "http://windows.php.net/downloads/releases/php-5.5.13-Win32-VC11-x64.zip" `

-Vc2012RedistDownloadUri "http://download.microsoft.com/download/1/6/B/16B06F60-3B20-4FF2-B699-5E9B7962F9AE/VSU\_4/vcredist\_x64.exe" `

-DestinationPath "C:\php" `

-ConfigurationPath "C:\MyPhp.ini" `

-installMySqlExt $false

# Renaming Requirements

1. Update the following names by replacing MSFT with your company/community name and replace the “x” with your own prefix (e.g. the resource name should change from MSFT\_xWebsite to Contoso\_myWebsite):

* **Module name**
* **Resource Name**
* **Resource Friendly Name**
* **MOF class name**
* **Filename for the <resource>.schema.mof**

1. Update module and metadata information in the module manifest
2. Update any configuration that use these resources

# Versions

1.0.0.0

* Initial Release with the following resources
  + xWebSite

1.1.0.0

* Second release adding and updating the following resources
  + xIisModule, added
  + xWebSite, updated with new property, DefaultPage