New features in Windows PowerShell 5.0

New features in Windows PowerShell

- Starting in Windows PowerShell 5.0, you can develop by using classes, by using formal syntax and semantics that are similar to other object-oriented programming languages. **Class**, **Enum**, and other keywords have been added to the Windows PowerShell language to support the new feature. For more information about working with classes, see about_Classes.
- In collaboration with Microsoft Research, a new cmdlet, ConvertFrom-String, has been added. ConvertFrom-String lets you extract and parse structured objects from the content of text strings. For more information, see ConvertFrom-String.
- A new module, Microsoft.PowerShell.Archive, includes cmdlets that let you compress files and folders into archive (also known as ZIP) files, extract files from existing ZIP files, and update ZIP files with newer versions of files compressed within them.
- A new module, OneGet, lets you discover and install software packages on the Internet. The OneGet module is a manager or multiplexer of existing package managers (also called package providers) to unify Windows package management with a single Windows PowerShell interface.
- A new module, PowerShellGet, lets you find, install, publish, and update modules and DSC resources on the PowerShell Resource Gallery, or on an internal module repository that you can set up by running the Register-PSRepository cmdlet.
- New-Item, Remove-Item, and Get-ChildItem have been enhanced to support creating and managing symbolic

links. The **ItemType** parameter for New-Item accepts a new value, **SymbolicLink**. Now you can create symbolic links in a single line by running the New-Item cmdlet.

- Windows PowerShell transcription has been improved to apply to all hosting applications (such as Windows PowerShell ISE) in addition to the console host (**powershell.exe**). Transcription options (including enabling a system-wide transcript) can be configured by enabling the **Turn on PowerShell Transcription** Group Policy setting, found in Administrative Templates/Windows Components/Windows PowerShell.
- A new detailed script tracing feature lets you enable detailed tracking and analysis of Windows PowerShell scripting use on a system. After you enable detailed script tracing, Windows PowerShell logs all script blocks to the Event Tracing for Windows (ETW) event log, **Microsoft-Windows-PowerShell/Operational**.
- Starting in Windows PowerShell 5.0, new Cryptographic Message Syntax cmdlets support encryption and decryption of content by using the IETF standard format for cryptographically protecting messages as documented by RFC5652. The Get-CmsMessage, Protect-CmsMessage, and Unprotect-CmsMessage cmdlets have been added to the Microsoft.PowerShell.Security module.
- New cmdlets in the Microsoft.PowerShell.Utility module, Get-Runspace, Debug-Runspace, Get-RunspaceDebug, Enable-RunspaceDebug, and Disable-RunspaceDebug, let you set debug options on a runspace, and start and stop debugging on a runspace. For debugging arbitrary runspaces—that is, runspaces that are not the default runspace for a Windows PowerShell console or Windows PowerShell ISE session—Windows PowerShell lets you set breakpoints in a script, and have added breakpoints stop the script from running until you can attach a debugger to debug the runspace script. Nested debugging support for arbitrary runspaces has been added to the Windows PowerShell script debugger for runspaces.

- New cmdlets Enter-PSHostProcess and Exit-PSHostProcess let you debug Windows PowerShell scripts in processes separate from the current process that is running in the Windows PowerShell console. Run Enter-PSHostProcess to enter, or attach to, a specific process ID, and then run Get-Runspace to return the active runspaces within the process. Run Exit-PSHostProcess to detach from the process when you are finished debugging the script within the process.
- A new Wait-Debugger cmdlet has been added to the Microsoft.PowerShell.Utility module. You can run Wait-Debugger to stop a script in the debugger before running the next statement in the script.
- The Windows PowerShell Workflow debugger now supports command or tab completion, and you can debug nested workflow functions. You can now press **Ctrl+Break** to enter the debugger in a running script, in both local and remote sessions, and in a workflow script.
- A Debug-Job cmdlet has been added to the Microsoft.PowerShell.Core module to debug running job scripts for Windows PowerShell Workflow, background, and jobs running in remote sessions.
- A new state, AtBreakpoint, has been added for Windows PowerShell jobs. The AtBreakpoint state applies when a job is running a script that includes set breakpoints, and the script has hit a breakpoint. When a job is stopped at a debug breakpoint, you must debug the job by running the Debug-Job cmdlet.
- Windows PowerShell 5.0 implements support for multiple versions of a single Windows PowerShell module in the same folder in $PSModulePath. A RequiredVersion property has been added to the ModuleSpecification class to help you get the desired version of a module; this property is mutually-exclusive with the ModuleVersion property. RequiredVersion is now supported as part of the value of the FullyQualifiedName parameter of the Get-Module, Import-Module, and Remove-Module cmdlets.

- You can now perform module version validation by running the Test-ModuleManifest cmdlet.
- Results of the Get-Command cmdlet now display a Version column; a new Version property has been added to the CommandInfo class. Get-Command shows commands from multiple versions of the same module. The Version property is also part of derived classes of CmdletInfo: CmdletInfo and ApplicationInfo.
- A new Get-ItemPropertyValue cmdlet lets you get the value of a property without using dot notation. For example, in older releases of Windows PowerShell, you can run the following command to get the value of the Application Base property of the PowerShellEngine registry key: **(Get-ItemProperty -Path HKLM:\SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine -Name ApplicationBase).ApplicationBase**. Starting in Windows PowerShell 5.0, you can run **Get-ItemPropertyValue -Path HKLM:\SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine -Name ApplicationBase**.
- A new NetworkSwitch module contains cmdlets that enable you to apply switch, virtual LAN (VLAN), and basic Layer 2 network switch port configuration to Windows Server 2012 R2 logo-certified network switches.
- The FullyQualifiedName parameter has been added to Import-Module and Remove-Module cmdlets, to support storing multiple versions of a single module.
- Save-Help, Update-Help, Import-PSSession, Export-PSSession, and Get-Command have a new parameter, FullyQualifiedModule, of type ModuleSpecification. Add this parameter to specify a module by its fully qualified name.
- The value of **$PSVersionTable.PSVersion** has been updated to 5.0.

New features in Windows PowerShell Desired State Configuration

- Windows PowerShell language enhancements let you define Windows PowerShell Desired State Configuration (DSC) resources by using classes. Import-DscResource is now a true dynamic keyword; Windows PowerShell parses the specified module's root module, searching for classes that contain the DscResource attribute. You can now use classes to define DSC resources, in which neither a MOF file nor a DSCResource subfolder in the module folder is required. A Windows PowerShell module file can contain multiple DSC resource classes.
- A new parameter, ThrottleLimit, has been added to the following cmdlets in the PSDesiredStateConfiguration module. Add the ThrottleLimit parameter to specify the number of target computers or devices on which you want the command to work at the same time.
    - Get-DscConfiguration
    - Get-DscConfigurationStatus
    - Get-DscLocalConfigurationManager
    - Restore-DscConfiguration
    - Test-DscConfiguration
    - Compare-DscConfiguration
    - Publish-DscConfiguration
    - Set-DscLocalConfigurationManager
    - Start-DscConfiguration
    - Update-DscConfiguration
- With centralized DSC error reporting, rich error information is not only logged in the event log, but it can be sent to a central location for later analysis. You can use this central location to store DSC configuration errors that have occurred for any server in their environment. After the report server is defined in the meta-configuration, all errors are sent to the report server, and then stored in a database. You can

set up this functionality regardless of whether or not a target node is configured to pull configurations from a pull server.

- Improvements to Windows PowerShell ISE ease DSC resource authoring. You can now do the following.
    - List all DSC resources within a **configuration** or **node** block by entering **Ctrl+Space** on a blank line within the block.
    - Automatic completion on resource properties of the **enumeration** type.
    - Automatic completion on the **DependsOn** property of DSC resources, based on other resource instances in the configuration.
    - Improved tab completion of resource property values.
- A new **DscLocalConfigurationManager** attribute designates a configuration block as a meta-configuration, which is used to configure the DSC Local Configuration Manager. This attribute restricts a configuration to containing only items which configure the DSC Local Configuration Manager. During processing, this configuration generates a *.meta.mof file that is then sent to the appropriate target nodes by running the Set-DscLocalConfigurationManager cmdlet.
- Partial configurations are now allowed in Windows PowerShell 5.0. You can deliver configuration documents to a node in fragments. For a node to receive multiple fragments of a configuration document, the node’s Local Configuration Manager must be first set to specify the expected fragments
- Cross-computer synchronization is new in DSC in Windows PowerShell 5.0. By using the built-in WaitFor* resources (**WaitForAll**, **WaitForAny**, and **WaitForSome**), you can now specify dependencies across computers during configuration runs, without external orchestrations. These resources provide node-to-node synchronization by using CIM connections over the WS-Man protocol. A

configuration can wait for another computer's specific resource state to change.

- Just Enough Administration (JEA), a new delegation security feature, leverages DSC and Windows PowerShell constrained runspaces to help secure enterprises from data loss or compromise by employees, whether intentional or unintentional. For more information about JEA, including where you can download the xJEA DSC resource, seeJust Enough Administration, Step by Step.
- The following new cmdlets have been added to the PSDesiredStateConfiguration module.
    - A new Get-DscConfigurationStatus cmdlet gets high-level information about configuration status from a target node. You can obtain the status of the last, or of all configurations.
    - A new Compare-DscConfiguration cmdlet compares a specified configuration with the actual state of one or more target nodes.
    - A new Publish-DscConfiguration cmdlet copies a configuration MOF file to a target node, but does not apply the configuration. The configuration is applied during the next consistency pass, or when you run the Update-DscConfiguration cmdlet.
    - A new Test-DscConfiguration cmdlet lets you verify that a resulting configuration matches the desired configuration, returning either True if the configuration matches the desired configuration, or False if the actual configuration does not match the desired configuration.
    - A new Update-DscConfiguration cmdlet forces a configuration to be processed. If the Local Configuration Manager is in pull mode, the cmdlet gets the configuration from the pull server before applying it.

New features in Windows PowerShell ISE

- You can now edit remote Windows PowerShell scripts and files in a local copy of Windows PowerShell ISE, by running Enter-PSSession to start a remote session on the computer that's storing the files you want to edit, and then running **PSEdit** . This feature eases editing Windows PowerShell files that are stored on the Server Core installation option of Windows Server, where Windows PowerShell ISE cannot run.
- The Start-Transcript cmdlet is now supported in Windows PowerShell ISE.
- You can now debug remote scripts in Windows PowerShell ISE.
- A new menu command, **Break All** (Ctrl+B), breaks into the debugger for both local and remotely-running scripts.

New features in Windows PowerShell Web Services (Management OData IIS Extension)

- Starting in Windows PowerShell 5.0, you can generate a set of Windows PowerShell cmdlets based on the functionality exposed by a given OData endpoint, by running the Export-ODataEndpointProxy cmdlet.

Notable bug fixes in Windows PowerShell 5.0

Windows PowerShell 5.0 includes a new COM implementation, which offers significant performance improvements when you are working with COM objects. For a video demonstration of the effect, see Com_Perf_Improvements.