New features in Windows PowerShell 4.0

Windows PowerShell 4.0 is backward-compatible. Cmdlets, providers, modules, snap-ins, scripts, functions, and profiles that were designed for Windows PowerShell 3.0 and Windows PowerShell 2.0 work in Windows PowerShell 4.0 without changes.

Windows PowerShell 4.0 is installed by default on Windows® 8.1 and Windows Server 2012 R2. To install Windows PowerShell 4.0 on Windows 7 with SP1, or Windows Server 2008 R2, download and install Windows Management Framework 4.0. Be sure to read the download details, and meet all system requirements, before you install Windows Management Framework 4.0.

Windows PowerShell 4.0 includes the following new features.

New features in Windows PowerShell

- **Windows PowerShell Desired State Configuration** (DSC) is a new management system in Windows PowerShell 4.0 that enables the deployment and management of configuration data for software services and the environment in which these services run. For more information about DSC, see Get Started with Windows PowerShell Desired State Configuration.
- **Save-Help** now lets you save help for modules that are installed on remote computers. You can use Save-Help to download module Help from an Internet-connected client (on which not all of the modules for which you want help are necessarily installed), and then copy the saved Help to a remote shared folder or a remote computer that does not have Internet access.
- The Windows PowerShell debugger has been enhanced to allow debugging of Windows PowerShell workflows, as well as scripts that are running on remote computers. Windows

PowerShell workflows can now be debugged at the script level from either the Windows PowerShell command line or Windows PowerShell ISE. Windows PowerShell scripts, including script workflows, can now be debugged over remote sessions. Remote debugging sessions are preserved over Windows PowerShell remote sessions that are disconnected and then later reconnected.

- A **RunNow** parameter for **Register-ScheduledJob** and **Set-ScheduledJob** eliminates the need to set an immediate start date and time for jobs by using the **Trigger** parameter.
- **Invoke-RestMethod** and **Invoke-WebRequest** now let you set all headers by using the Headers parameter. Although this parameter has always existed, it was one of several parameters for the web cmdlets that resulted in exceptions or errors.
- **Get-Module** has a new parameter, **FullyQualifiedName**, of the type **ModuleSpecification[]**. The **FullyQualifiedName** parameter of Get-Module now lets you specify a module by using the module's name, version, and optionally, its GUID.
- The default execution policy setting on Windows Server 2012 R2 is **RemoteSigned**. On Windows 8.1, there is no change in default setting.
- Starting in Windows PowerShell 4.0, method invocation by using dynamic method names is supported. You can use a variable to store a method name, and then dynamically invoke the method by calling the variable.
- Asynchronous workflow jobs are no longer deleted when the time-out period that is specified by the **PSElapsedTimeoutSec** workflow common parameter has elapsed.
- A new parameter, **RepeatIndefinitely**, has been added to the **New-JobTrigger** and **Set-JobTrigger** cmdlets. This eliminates the necessity of specifying a**TimeSpan.MaxValue** value for the **RepetitionDuration** parameter to run a scheduled job repeatedly for an

indefinite period.

- A **Passthru** parameter has been added to the **Enable-JobTrigger** and **Disable-JobTrigger** cmdlets. The Passthru parameter displays any objects that are created or modified by your command.
- The parameter names for specifying a workgroup in the **Add-Computer** and **Remove-Computer** cmdlets are now consistent. Both cmdlets now use the parameter**WorkgroupName**.
- A new common parameter, **PipelineVariable**, has been added. PipelineVariable lets you save the results of a piped command (or part of a piped command) as a variable that can be passed through the remainder of the pipeline.
- Collection filtering by using a method syntax is now supported. This means that you can now filter a collection of objects by using simplified syntax, similar to that for Where() or Where-Object, formatted as a method call. The following is an example: (Get-Process).where({$_.Name -match 'powershell'})
- The **Get-Process** cmdlet has a new switch parameter, **IncludeUserName**.
- A new cmdlet, **Get-FileHash**, that returns a file hash in one of several formats for a specified file, has been added.
- In Windows PowerShell 4.0, if a module uses the **DefaultCommandPrefix** key in its manifest, or if the user imports a module with the **Prefix** parameter, the**ExportedCommands** property of the module shows the commands in the module with the prefix. When you run the commands by using the module-qualified syntax, ModuleName\CommandName, the command names must include the prefix.
- The value of **$PSVersionTable.PSVersion** has been updated to 4.0.
- **Where()** operator behavior has changed. Collection.Where('property -match name') accepting a

string expression in the format "Property -CompareOperator Value" is no longer supported. However, the **Where()** operator accepts string expressions in the format of a scriptblock; this is still supported.

New features in Windows PowerShell Integrated Scripting Environment (ISE)

- Windows PowerShell ISE supports both Windows PowerShell Workflow debugging and remote script debugging.
- IntelliSense support has been added for Windows PowerShell Desired State Configuration providers and configurations.

New features in Windows PowerShell Workflow

- Support has been added for a new **PipelineVariable** common parameter in the context of iterative pipelines, such as those used by System Center Orchestrator; that is, pipelines that run commands simply left-to-right, as opposed to interspersed running by using streaming.
- Parameter binding has been significantly enhanced to work outside of tab completion scenarios, such as with commands that do not exist in the current runspace.
- Support for custom container activities has been added to Windows PowerShell Workflow. If an activity parameter is of the types **Activity**, **Activity[]**—or is a generic collection of activities—and the user has supplied a script block as an argument, then Windows PowerShell Workflow converts the script block to XAML, as with normal Windows PowerShell script-to-workflow compilation.
- After a crash, Windows PowerShell Workflow automatically

reconnects to managed nodes.

- You can now throttle **Foreach -Parallel** activity statements by using the **ThrottleLimit** property.
- The **ErrorAction** common parameter has a new valid value, **Suspend**, that is exclusively for workflows.
- A workflow endpoint now automatically closes if there are no active sessions, no in-progress jobs, and no pending jobs. This feature conserves resources on the computer that is acting as the workflow server, when the automatic closure conditions have been met.

New features in Windows PowerShell Web Services

- When an error occurs in Windows PowerShell Web Services (PSWS, also called Management OData IIS Extension), while a cmdlet is running, more detailed error messages are returned to the caller. In addition, error codes follow Windows Azure REST API error code guidelines.
- An endpoint can now define the API version, as well as enforce the usage of a specific API version. Whenever version mismatches occur between client and server, errors are displayed to both the client and the server.
- Management of the dispatch schema has been simplified by automatically generating values for any missing fields in the schema. Generation occurs, as a helpful starting point, even if the dispatch schema does not exist.
- Type handling in PSWS has been improved to support types that use a different constructor than the default constructor, by behaving similarly to the **PSTypeConverter** in Windows PowerShell. This lets you use complex types with PSWS.
- PSWS now allows expanding an associated instance while running a query. For larger binary contents (such as images, audio, or video), the transfer cost is significant, and it is better to transfer binary data

without encoding. PSWS uses named resource streams for transferring without encoding. The named resource stream is a property of an entity of the **Edm.Stream** type. Each named resource stream has a separate URI for GET or UPDATE operations.

- OData actions now provide a mechanism for invoking non-CRUD (Create, Read, Update, and Delete) methods on a resource. You can invoke an action by sending an HTTP POST request to the URI that is defined for the action. The parameters for the action are defined in the body of the POST request.
- To be consistent with Windows Azure guidelines, all URLs should be simplified. A change included in **Key As Segment** allows single keys to be represented as segments. Note that references that use multiple key values require comma-separated values in parenthetical notation, as before.
- Before this release of PSWS, the only way to perform Create, Update, or Delete operations was to invoke Post, Put, or Delete on a top-level resource. New in this release of PSWS, Contained Resource operations let users achieve the same results while reaching the same resource less directly, approaching as if these resources were contained.

New features in Windows PowerShell Web Access

- You can disconnect from and reconnect to existing sessions in the web-based Windows PowerShell Web Access console. A **Save** button in the web-based console lets you disconnect from a session without deleting it and reconnect to the session another time.
- Default parameters can be displayed on the sign-in page. To display default parameters, configure values for all of the settings displayed in the **Optional Connection**

Settings area of the sign-in page in a file named **web.config**. You can use the **web.config** file to configure all optional connection settings except for a second or alternate set of credentials.

- In Windows Server 2012 R2, you can remotely manage authorization rules for Windows PowerShell Web Access. The **Add-PswaAuthorizationRule** and **Test-PswaAuthorizationRule** cmdlets now include a Credential parameter that enables administrators to manage authorization rules from a remote computer or in a Windows PowerShell Web Access session.
- You can now have multiple Windows PowerShell Web Access sessions in a single browser session by using a new browser tab for each session. You no longer need to open a new browser session to connect to a new session in the web-based Windows PowerShell console.

Notable bug fixes in Windows PowerShell 4.0

- **Get-Counter** can now return counters that contain an apostrophe character in French editions of Windows.
- You can now view the **GetType** method on deserialized objects.
- **#Requires** statements now let users require Administrator access rights, if needed.
- The **Import-Csv** cmdlet now ignores blank lines.
- A problem where Windows PowerShell ISE uses too much memory when you are running an **Invoke-WebRequest** command has been fixed.
- **Get-Module** now displays module versions in a **Version** column.
- Remove-Item -Recurse now removes items from subfolders as expected.
- A **UserName** property has been added to **Get-Process** output objects.

- The **Invoke-RestMethod** cmdlet now returns all available results.
- **Add-Member** now takes effect on hashtables, even if the hashtables have not yet been accessed.
- **Select-Object -Expand** no longer fails or generates an exception if the value of the property is null or empty.
- **Get-Process** can now be used in a pipeline with other commands that get the **ComputerName** property from objects.
- **ConvertTo-Json** and **ConvertFrom-Json** can now accept terms within double quotes, and its error messages are now localizable.
- **Get-Job** now returns any completed scheduled jobs, even in new sessions.
- Issues with mounting and unmounting VHDs by using the **FileSystem** provider in Windows PowerShell 4.0 have been fixed. Windows PowerShell is now able to detect new drives when they are mounted in the same session.
- You no longer need to explicitly load **ScheduledJob** or **Workflow** modules to work with their job types.
- Performance improvements have been made to the process of importing workflows that define nested workflows; this process is now faster.