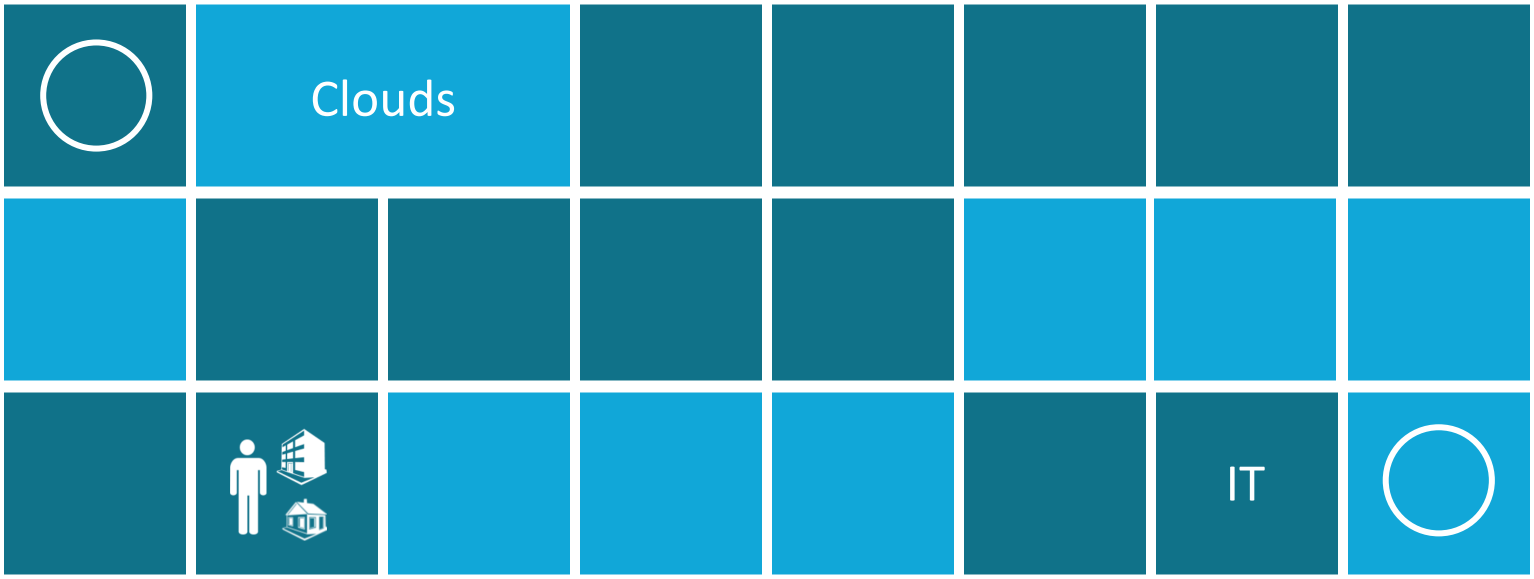
Microsoft®



Windows Server 2012 Release Candidate

Server Management

Contents

Introduction

Windows Server® 2012 brings the Microsoft® experience of building and operating public clouds to the server operating system, helping to make it a dynamic, highly available, and cost-effective platform for private clouds. It offers businesses and hosting providers a basis for a scalable, dynamic, and multitenant-aware cloud infrastructure that more securely connects across premises and allows IT to respond to business needs faster and more efficiently.

Windows Server 2012 Release Candidate (RC) offers excellent total cost of ownership as an integrated platform with comprehensive, multicomputer manageability. Three areas in which Windows Server 2012 RC improves multicomputer management are Server Manager, Windows PowerShell™ 3.0, and IP Address Management Services (IPAM). Server Manager in Windows Server 2012 RC helps you deploy and manage roles and features on the local server, on remote servers, and on both online and offline virtual hard disks. Windows PowerShell 3.0 provides comprehensive automation capabilities. Finally, IP Address Management Services provides a set of tools to help administrators manage the IP address infrastructure of their company network as a whole.

The following sections provide more detail about these three features of Windows Server 2012 RC.

Multi-server Management and Feature Deployment with Server Manager

In Windows Server 2012 RC, the capabilities of Server Manager have expanded considerably to facilitate multi-server tasks such as remote role and feature deployment to both physical and virtual servers, remote role and feature management, and custom server group creation.

By using Server Manager in Windows Server 2012 RC, IT pros can now provision servers and offline virtual hard disks from their desktops without requiring either physical access to the system or Remote Desktop Protocol (RDP) connections to each server. Server Manager also helps administrators manage groups of servers collectively from within a single, integrated console, allowing them respond to business-critical problems with greater speed and agility.

Technical description

Server Manager in Windows Server 2012 RC has evolved to include many server management features. The following sections describe some of the new capabilities provided by the new Server Manager.

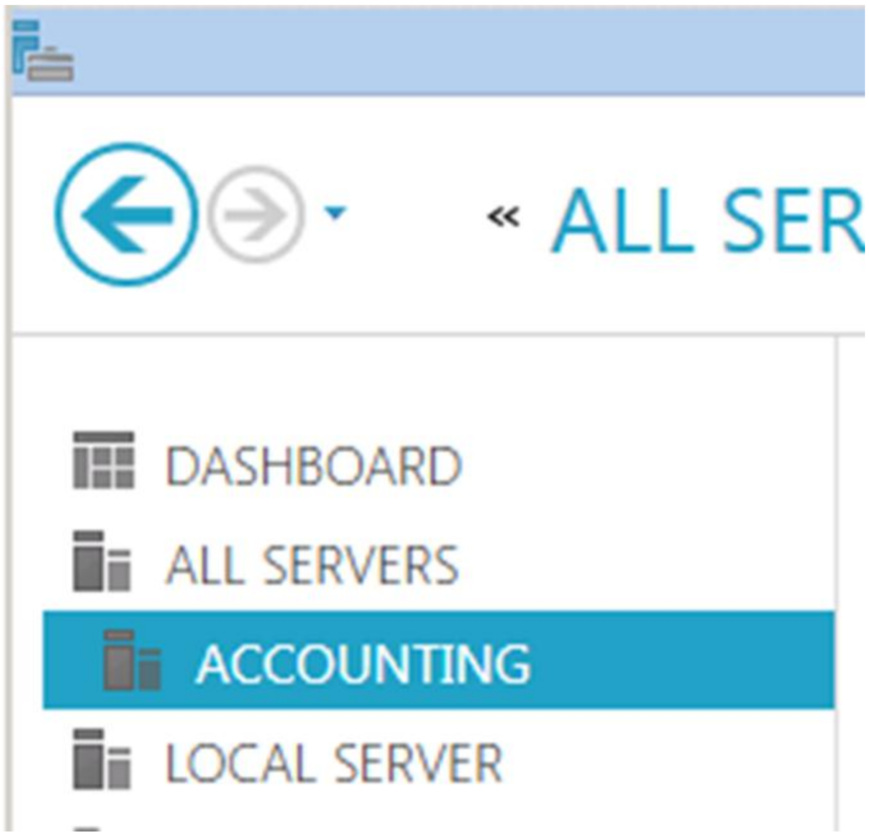
Multi-server experience

Server Manager can handle multiple servers in a server pool, and create server groups to organize them. The server grouping functionality resembles the grouping functionality in Windows Server Update Services (WSUS), or the cloud service Windows Intune™. Groups let you manage servers that are related by certain common characteristics (such as location, function, Windows operating system release, or hardware type) as if they were a single unit.

Improved management through high availability

Windows Server 2012 RC introduces new storage and networking features that improve management by preventing downtime by enduring various failures while maintaining service availability. For example, Windows Server 2012 RC introduces Server Message Block (SMB) 3.0, which improves the availability of server applications through features such as SMB Transparent Failover and SMB Multichannel which make effective, fault-tolerant use of multiple NICs. Another feature, NIC Teaming, supports multichannel traffic and failover for traffic that is not SMB-based.

For more information about availability improvements and SMB 3.0, see the white paper *Windows Server 2012 RC Storage*. For more information about NIC Teaming, see the white paper *Windows Server 2012 RC Networking*.



A server group in Server Manager

Efficient deployment of workloads to remote servers and offline virtual hard disks

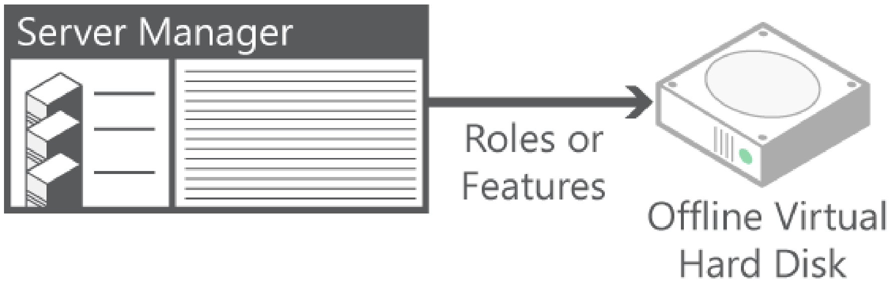
In Windows Server 2008 R2, roles and features are deployed by using the Add Roles Wizard or Add Features Wizard in Server Manager running on a local server. This requires either physical access to the server or Remote Desktop access by using Remote Desktop Protocol (RDP). Installing Remote Server Administration Tools lets you run Server Manager on a Windows-based client computer, but adding roles and features is disabled, because remote deployment is not supported.

In Windows Server 2012 RC, the deployment capabilities are extended to support robust remote deployment of roles and features. Using Server Manager in Windows Server 2012 RC, IT pros can provision servers from their desktops without requiring either physical access to the systems or the need to enable an RDP connection to each server.

Installing roles and features on a remote server/offline virtual hard disk

Windows Server 2012 RC with Server Manager can deploy both roles and features in a single session using the unified Add Roles and Features Wizard. The Add Roles and Features Wizard in Windows Server 2012 RC performs validation passes on a server that you select for deployment as part of the installation process; you don't need to pre-verify that a server in your Server Manager server pool is properly configured to support a role.

Administrators can deploy roles and features to remote servers and offline virtual hard disks from Server Manager on their local servers. In a single session in the Add Roles and Features Wizard, you can add your desired roles and features to an offline virtual hard disk, allowing for faster and simpler repetition and consistency of desired configurations.

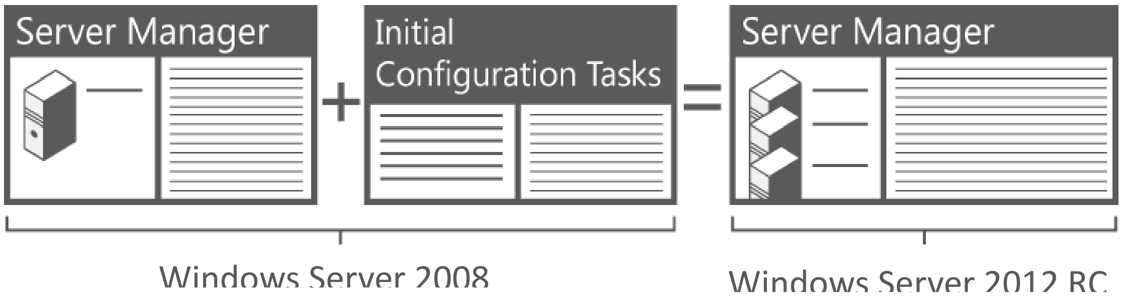


With Server Manager in Windows Server 2012 RC, you can deploy roles or features to an offline virtual hard disk.

Streamlined server configuration and deployment

In Windows Server 2012 RC, Server Manager includes configuration functionality previously provided by the Initial Configuration Tasks window. The result is a single surface for managing the configuration of Windows Server and its roles and features.

Deployment of both roles and features has been combined into a single Add Roles and Features Wizard. While the process of installing roles is familiar, and consistent with the Add Roles Wizard in earlier Windows Server releases, there are changes. To support remote deployment and installations on offline virtual hard disks, some roles have moved some initial configuration (tasks formerly performed in the Add Roles Wizard during an installation) into post-installation configuration wizards. For some offline virtual hard disk deployments, installation tasks are scheduled to run the first time the virtual machine is started.



Server Manager in Windows Server 2012 RC combines the functionality of two tools in Windows Server 2008 R2.

Batch deployment

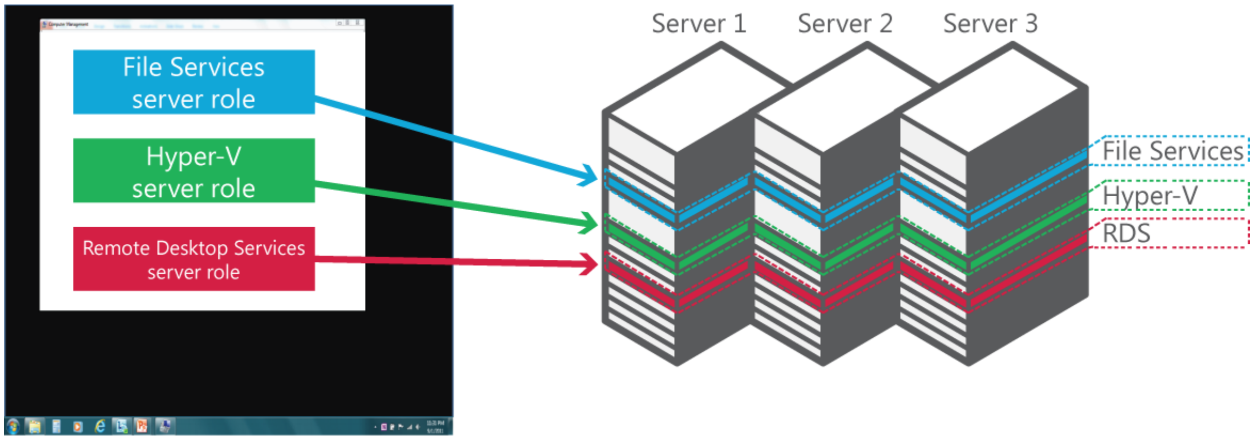
In Windows Server2012 RC, the Add Roles and Features Wizard lets you export configuration options to an XML file, for later use with Windows PowerShell deployment cmdlets. By using the fan-out capabilities of Windows PowerShell, you can perform batch deployment of roles and features on multiple remote servers, applying configuration settings that were saved during a previous wizard-based deployment.

Integration with other management tools

Server Manager remains the key access or launch point for server management tools. Where supported, Server Manager launches these tools in the context of the remote server that you are managing. New, modern, role-specific tools (such as File Storage Management, Remote Desktop Services, and IP Address Management) are integrated into the Server Manager console.

Server role management across multiple servers

Management of server roles is improved by shifting from a single-server, single-role model, to one in which multiple server roles can be managed remotely by using a single management application.



In Server Manager in Windows Server 2012 RC, you can manage server roles such as File Services across multiple servers.

Remote Desktop Services configuration

Remote Desktop Services provides session virtualization and virtual desktop infrastructure technologies that enable users to access session and virtual desktop collections. In Windows Server 2012 RC, new management features of Server Manager simplify how Remote Desktop Services is deployed and managed in a multi-server environment. Scenario-based deployment reduces the complexity of installing different Remote Desktop Services components across multiple servers based on how Remote

Desktop Services will be used. New multi-server management tools then simplify how administrators manage different servers that are running Remote Desktop Services role services and virtual desktop infrastructures.

Minimal performance impact

The Server Manager dashboard has a default 10-minute polling cycle that users can modify in the console. By using a relatively infrequent default polling cycle, and returning only incremental data with each poll, the performance-load impact on individual servers is minimized. Server Manager uses new Windows Management Instrumentation (WMI) providers and Windows PowerShell cmdlets to pull updated status information from servers.

Requirements

Server Manager in Windows Server 2012 RC requires the following:

- Windows Server 2012 RC (**Full installation** option)
- Remote deployment is only supported to computers that are running Windows Server 2012 RCj; remote deployment of roles and features to Windows Server 2008 R2 and earlier Windows releases is not supported.
- For remote management, the value of the **Remote Management** property must remain **Enabled** on the Local Server page in the Server Manager console on a server that you want to manage remotely. (This property is enabled by default in both Server Manager and Windows PowerShell.)

Summary

Windows Server 2012 RC Server Manager improves management of the data center by letting you do the following:

- Manage multiple servers easily with a clear and powerful role-centric dashboard.
- Simplify the processes of configuring new servers.
- Deploy roles and features even to remote servers and offline virtual hard disks.
- Consult a single tool for a clear summary of multiple server states.

Comprehensive, Resilient, and Simple Automation with Windows PowerShell 3.0

Windows PowerShell 3.0 provides a comprehensive platform to help you manage most server roles and aspects of the data center. In this newest version of Windows PowerShell, sessions to remote servers are resilient and can withstand various types of interruptions. In addition, learning Windows PowerShell is now easier than ever through improved cmdlet discovery and simplified, consistent syntax across all cmdlets.

Technical description

The following sections describe the major features of Windows PowerShell.

Robust Session Connectivity

Long running tasks, such as deploying a service pack or backing up a database, need to continue even if the client computer that initiated the requested operation goes down or disconnects.

With Robust Session Connectivity, remote sessions can remain in a connected state for up to four minutes, even if the client crashes or becomes inaccessible, and tasks on the managed nodes continue to run on their own making the end to end system more reliable. If connectivity cannot be restored within four minutes, execution on the managed nodes is suspended with no loss of data and remote sessions automatically transition to a disconnected state, allowing them to be reconnected after network connectivity is restored. Corruption of application and system state from premature termination of running tasks due to unexpected client disconnection is virtually eliminated.

Disconnected Sessions

Windows PowerShell 3.0 lets you disconnect from and then reconnect to any session without losing state. Disconnected Sessions allows you to create a session on a remote computer, start a command or job, disconnect from the session, shut down your computer, and then reconnect to the session from a different computer at a later time to check the job status or get the results. When administrators are disconnected from the session, persistent commands and jobs can continue to run.

The functionality of the following cmdlets demonstrates the Disconnected Sessions capability in Windows PowerShell 3.0:

- **Disconnect-PSSession.** Disconnects a session connection from a remote computer.
- **Connect-PSSession.** Reestablishes a session connection with a remote computer.

- **Receive-PSSession.** Resumes execution of a command on a remote session and retrieves the session output. Implicitly reconnects to session (without Connect-PSSession command).

Example:

```
# Start a remote session, disconnect from the session, and exit PowerShell.
PS C:\> $s = New-PSSession -ComputerName srv1 -Name LongSession
PS C:\> $job = Invoke-Command $s { 1..10| % {echo "Long running job - part $_";
sleep 5} } -AsJob
PS C:\> Disconnect-PSSession $s
exit

# Start Windows PowerShell on a different computer.
PS C:\> $s = Get-PSSession -ComputerName srv1 -Name LongSession
PS C:\> Receive-PSSession $s
```

Job scheduling

Windows PowerShell 3.0 allows administrators to schedule jobs to be run at a later time, or according to a particular schedule. To create a scheduled job, you first create a job definition, which names the job and specifies the commands that it runs, and then a job trigger, which specifies the job schedule. The Windows Task Scheduler is used to schedule and start the job and a per-user job repository is used to store job output so that it's available later in a Windows PowerShell session on the computer.

The following new cmdlets are available in the PSScheduledJob module to help you work with scheduled jobs:

- Add-JobTrigger
- Disable-JobTrigger
- Enable-JobTrigger
- Get-JobTrigger
- New-JobTrigger
- Remove-JobTrigger
- Set-JobTrigger
- Disable-ScheduledJob
- Enable-ScheduledJob
- Get-ScheduledJob
- Register-ScheduledJob
- Set-ScheduledJob
- Unregister-ScheduledJob
- Get-ScheduledJobOption
- New-ScheduledJobOption
- Set-ScheduledJobOption

Jobs can be scheduled to execute based on the following job triggers:

- Once
- Daily
- Weekly
- At Startup
- At Logon

Example:

```
$trigger = New-JobTrigger -Daily -At 4am
Register-ScheduledJob -Name MyScheduledJob -ScriptBlock { Get-Date } -Trigger
$trigger
Get-ScheduledJob
```

You can start a scheduled job manually.

Example:

```
Start-Job -DefinitionName MyScheduledJob
```

Once the trigger has fired and the job has run, you can work with it the same way you do regular background jobs.

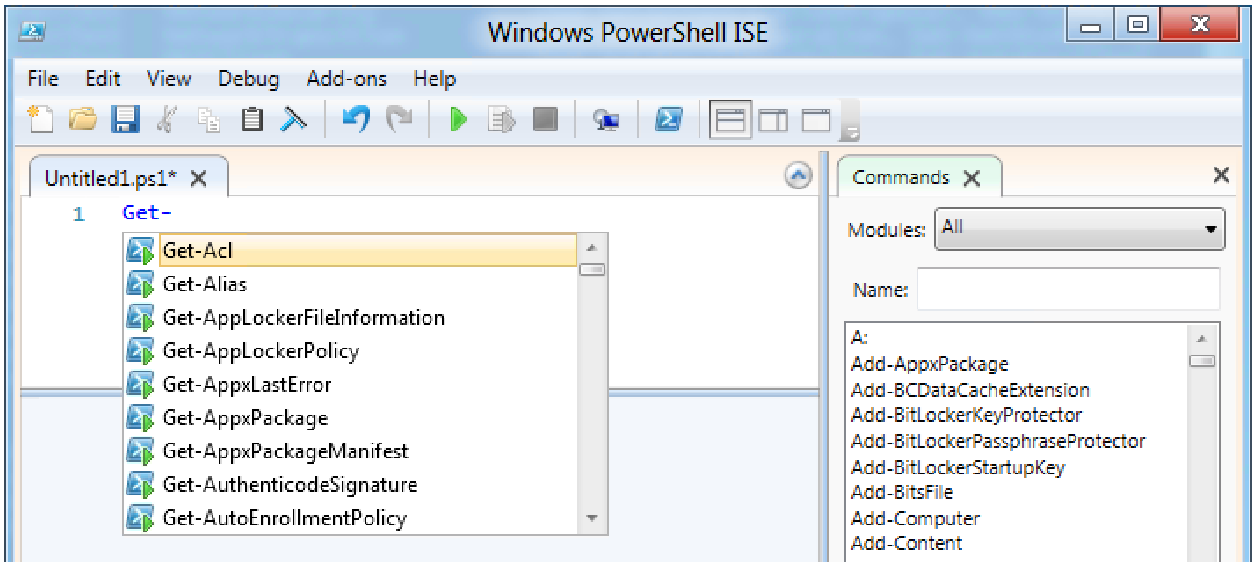
Example:

```
Import-Module PSScheduledJob
$j = Get-Job -Name MyScheduledJob
Receive-Job $j
```

New Windows PowerShell ISE features

The Windows PowerShell Integrated Scripting Environment (ISE) 3.0 includes many new features to ease beginning users into Windows PowerShell and provide advanced editing support for scripters. The following are some of the new features:

- Show-Command pane lets users find and run cmdlets in a dialog box.

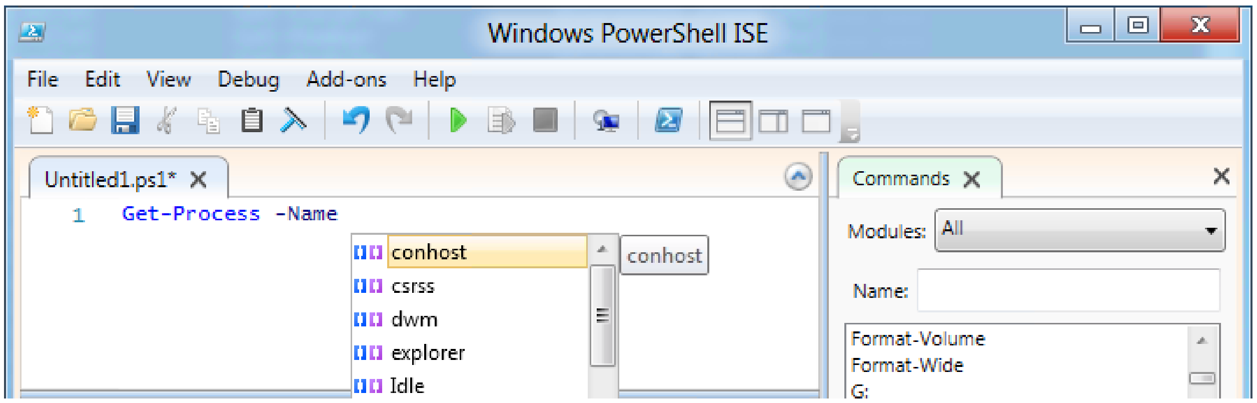


```
Get-
```

Get-Acl
Get-Alias
Get-AppLockerFileInformation
Get-AppLockerPolicy
Get-AppxLastError
Get-AppxPackage
Get-AppxPackageManifest
Get-AuthenticodeSignature
Get-AutoEnrollmentPolicy

A:
Add-AppxPackage
Add-BCDataCacheExtension
Add-BitLockerKeyProtector
Add-BitLockerPassphraseProtector
Add-BitLockerStartupKey
Add-BitsFile
Add-Computer
Add-Content

- IntelliSense provides context-sensitive command completion for cmdlet and script names, parameter names and enumerated values, and property and method names. IntelliSense also supports paths, types, and variables.



```
Get-Process -Name
```

- Code examples add reusable text to scripts and commands. The built-in code examples include templates for functions, parameters, and statements so that users don't have to remember the syntax.

- Collapsible regions in scripts and XML files make navigation in long scripts easier.

Windows PowerShell workflows

IT pros often automate the management of their multi-computer environments by running sequences of long-running tasks or workflows that can affect multiple managed computers or devices at the same time. Windows PowerShell workflows let IT pros and developers apply the benefits of workflows to the automation capabilities of Windows PowerShell.

A workflow is a sequence of automated steps or activities that execute tasks on or retrieve data from one or more managed nodes (computers or devices). These activities can include individual commands or scripts. Windows PowerShell workflows enable IT pros and developers alike to author sequences of multicomputer management activities — that are either long-running, repeatable, frequent, parallelizable, interruptible, stoppable, or restartable — as workflows. By design, workflows can be resumed from an intentional or accidental suspension or interruption, such as a network outage, a reboot or power loss. Workflows are also portable; they can be either exported as or imported from XAML files.

Custom session configurations (also called endpoints) allows workflows or activities within a workflow to be run by delegated or subordinate users. The security descriptors on the session configurations determine who can use them to create sessions, and thus manage nodes at different stages of a workflow.

Benefits of Windows PowerShell workflows

Windows PowerShell workflows manage the distribution, sequencing, and completion of multicomputer tasks, freeing users and administrators to focus on higher-level tasks. The following list describes many of the benefits of Windows PowerShell workflows.

- **Automation of sequenced, long-running tasks.** Remote monitoring of long-running tasks. Status and progress of activities are visible at any time.
- **Multicomputer management.** Simultaneously run tasks as workflows on up to hundreds of managed nodes. PowerShell workflows include a built-in library of common management parameters for workflows, which enables multi-computer management scenarios, such as PSComputerName.
- **Single task execution of complex processes.** You can combine related scripts that act on an entire end-to-end scenario into a single workflow.
- **Robustness: Automated failure recovery.** Workflow survives both planned and unplanned restarts. You can suspend workflow execution and then resume the workflow from the last persisted point, which is normally the point at which it was suspended.
- **Persistence:** a workflow is saved (or check-pointed) at specific points defined by its author so you can resume the workflow from the last persisted task (or checkpoint), instead of restarting the workflow from the very beginning.

  Additionally, workflow authors can designate specific activities to be re-run in case of failure on one or more managed nodes (for example, if one of the computers was down at the time the activity ran).
- **Ability to connect and disconnect (Disconnected Sessions).** Users can connect and disconnect from the workflow server, but the workflow remains running. For example, you can log off the client computer or restart the client computer, and monitor the workflow execution from another computer (such as a home computer) without interrupting the workflow. This is possible as long the client is running on a different computer than the workflow engine computer.

  Disconnect from a workflow and then reconnect; restart the managed nodes without disrupting workflows.
- **Ability to be scheduled.** Workflow tasks can be scheduled, just as any Windows PowerShell cmdlet or script.
- **Workflow and Connection Throttling.** Workflow execution and connections to nodes can be throttled, thus enabling Scalability and High Availability scenarios.

When to use a Windows PowerShell workflow instead of a cmdlet?

In general, you should consider using a workflow instead of a cmdlet when you need to meet any of the following requirements:

- You need to perform a long running task that combines multiple activities in a sequence.
- You need to perform a task that runs on multiple computers.
- You need to perform a task that requires bookmarking or checkpointing persistence.
- You need to perform a long-running task that is asynchronous, restartable, parallelizable and/or interruptible (like jobs, but more complex).
- You need the task to be logged in a way that combines data from all participating computers.
- You need the task to run at scale or in high availability environments, potentially requiring throttling and connection pooling.

Writing and running workflows in Windows PowerShell: Examples

Typically, workflows are started from a client computer and are ideal for executing long running tasks across multiple target computers. Workflows are just like any other PowerShell cmdlet, which means that you can use the Get-Command cmdlet to discover them, and the Get-Help cmdlet to learn how to use them. They can also survive machine and network interruptions such as reboots.

You can add a workflow to a Windows PowerShell session, either by typing it at the command line, including it in a script, or by using the Import-Module cmdlet to import a Windows PowerShell script workflow or a XAML-based workflow. Then the workflow behaves just like any other PowerShell cmdlet in that session.

Each step or command inside the workflow is called an *activity*. Each activity inherits the properties of the workflow, including the powerful Workflow common parameters mentioned above.

To write a workflow, you can either use the regular PowerShell console or Windows PowerShell ISE. For example, you can type the following workflow into the Windows PowerShell ISE Command pane:

```
Workflow MyWorkflow
{
   Write-Output -InputObject "Hello from Workflow!"
}
```

Notice the new "Workflow" keyword, which indicates that the command is a Windows PowerShell workflow. The keyword adds more than 20 new parameters to the workflow, allowing users to specify items such as:

- A list of target computers for the workflow (-PSComputerName)
- Credentials to use for running the workflow (-PSCredential)
- Quotas to manage the workflow as the work scales (e.g., -PSRunningTimeoutSec)
- Ability to retry the whole workflow or specific activities in case there are connection issues (for example, PSConnectionRetryCount)
- Ability to persist or checkpoint workflow activities, which will save the workflow metadata, output and errors to disk and enable you to resume workflow execution at given points during the execution (-PSPersist, -PSPersistInterval)

To run a workflow, type the workflow name, just as you would do to run any other Windows PowerShell command. For example, to run the new workflow we have just created, you can type **MyWorkflow** at the Windows PowerShell prompt.

The following is another example workflow, named LongWorkflow, that runs for approximately 30 seconds.

```
Workflow LongWorkflow
{
   Write-Output -InputObject "Loading some information..."
   Start-Sleep -Seconds 10
   Write-Output -InputObject "Performing some action..."
   Start-Sleep -Seconds 10
   Write-Output -InputObject "Cleaning up..."
   Start-Sleep -Seconds 10
}
```

Because this workflow defines a long task, you might want to run it as a background job. To do so, you can use the **AsJob** parameter, along with the **JobName** parameter to assign the "LongWF" name to the job.

```
LongWorkflow –AsJob –JobName LongWF
```

This third example that follows is a more complex workflow. This workflow, Install-Datacenter, simulates a basic datacenter deployment scenario. There are advanced workflow steps commented out at the end of the workflow in case you would like to practice and learn more.

```
<# This is a long running workflow that can be executed across multiple computers
running Windows Server to simulate a data center deployment.
 This workflow showcases the new PowerShell Workflow feature set of Windows Server
2012. In this particular example, the managed nodes must be running
a Windows Server 2012 SKU because of the Get-WindowsFeature activity below. #>


Workflow Install-Datacenter {

    Write-Output -InputObject "Let's get this Datacenter up and running!!!"
    InlineScript {"Installing...     " + (hostname); Start-Sleep -Seconds 2} -pscomputername
$pscomputername -pspersist:$TRUE
    Checkpoint-Workflow
    Write-Output -InputObject "=== Suspend NOW!!! ==="
    Start-sleep -seconds 8


    foreach -parallel ($computer in $pscomputername)
    {
       Sequence {
            InlineScript { cd WSMan:\localhost\Client }
            InlineScript { Set-Item AllowUnencrypted $TRUE -Force }
            InlineScript { Set-Item TrustedHosts *  -Force }
            Start-Sleep -Seconds 2
            Write-Output -InputObject "Done configuring WSMan"
       }

       # You could add an IF statement depending on the outcome of the following:
       Get-WindowsFeature

       # Let's simulate a long running task that can execute in parallel with WSMan
configuration
       Start-Sleep -Seconds 10
       Write-Output -InputObject "Time to get coffee, app setup about to start"

    }



    InlineScript { 1..30 | ForEach-Object `
       -Begin    { "`n=== Installing Datacenter Applications ===" }`
       -Process { "    Installing component $_..."; start-sleep -Seconds 2 }`
       -End      { "=== Datacenter Installation Complete ===`n" }`
     }` -PSComputerName $PSComputerName -PSPersist:$TRUE


# ADVANCED FEATURES YOU CAN TRY:

##################
# USE CASE 1:
##################

# Other things we could use workflows for: multiple, long running,
# pre-existing scripts that need to be executed across multiple machines
# and come together as a single end-to-end scenario. For example:

#   InlineScript { \\<REPLACEWITHYOURCN>\InstallIIS.ps1  ; Start-Sleep -Seconds 10} -
PSComputerName $computer
#   InlineScript { \\<REPLACEWITHYOURCN>\StartIIS.ps1    ; Start-Sleep -Seconds 10} -
PSComputerName $computer
#   Restart-computer -ComputerName $pscomputername -wait -for PowerShell
##################
# USE CASE 2:
##################

# Check for a condition (for example regkey set to value X) and if it's true, suspend the
workflow from within the workflow itself.
# This way, an admin can interact with the machine in question, resolve the problem and then
resume execution.

#   If (<REPLACE: ISSUE YOU WANT TO CHECK FOR, e.g., regkey value>) {
#      Suspend-Workflow
#      Write-Output -InputObject "ADMIN: Please fix the <REPLACE: ISSUE IN IF CONDITION> and
resume this workflow by running resume-job <job variable or -id <JOBID>"

#   }

#   Write-Output -InputObject "Registry Problem fixed. DataCenter deployed successfully. GO
HOME, HAVE A LIFE! :)"


}
Cmdlet discovery: Get-Command and module auto-loading.
```

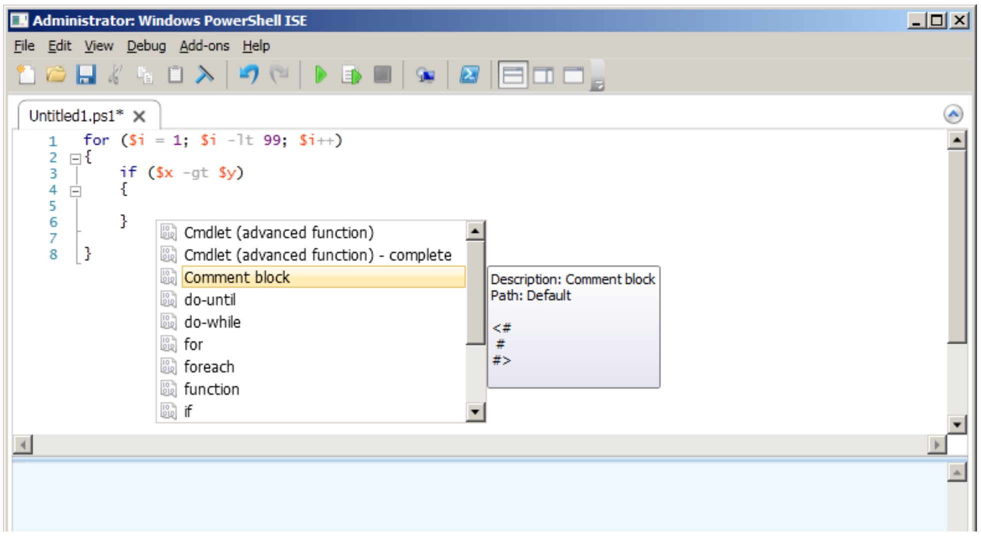
Windows Server 2012 RC includes more than 2,300 cmdlets, which you can learn and discover easily. Modules are easier than ever to find, explore, create, and use, and users no longer have to import modules manually to use cmdlets. Users can just run a cmdlet, and Windows PowerShell will automatically import the module. In addition, Get-Command has been updated to find all cmdlets installed on the system. For example, to find all networking cmdlets, you can run Get-Command *-Net*.

Syntax simplification

Windows PowerShell 3.0 includes simplified, consistent syntax across all cmdlets. The ForEach-Object and Where-Object cmdlets have been updated to support an intuitive command structure that more closely models natural language. Users are able to construct commands without script block, braces, the current object automatic variable ($_), or dot operators to get properties and methods. In short, the “punctuation” that plagued beginning users is no longer required.

Script sharing

Windows PowerShell 3.0 helps IT pros by providing access to a community-generated library of Windows PowerShell code snippets, called Integrated Script Snippets, within Windows PowerShell ISE. To access Integrated Script Snippets, the user presses the key combination (Ctrl+J). The user can then select from a list of script templates, select the appropriate template, and have the partially completed script inserted into the editor. By default, ISE ships with 12 script snippets to ease creating the commonly used programming syntax patterns.


Administrator: Windows PowerShell ISE
File Edit View Debug Add-ons Help
Untitled1.ps1*

```
for ($i = 1; $i -lt 99; $i++)
{
    if ($x -gt $y)
    {

    }

}
```

Cmdlet (advanced function)
Cmdlet (advanced function) - complete
Comment block
do-until
do-while
for
foreach
function
if

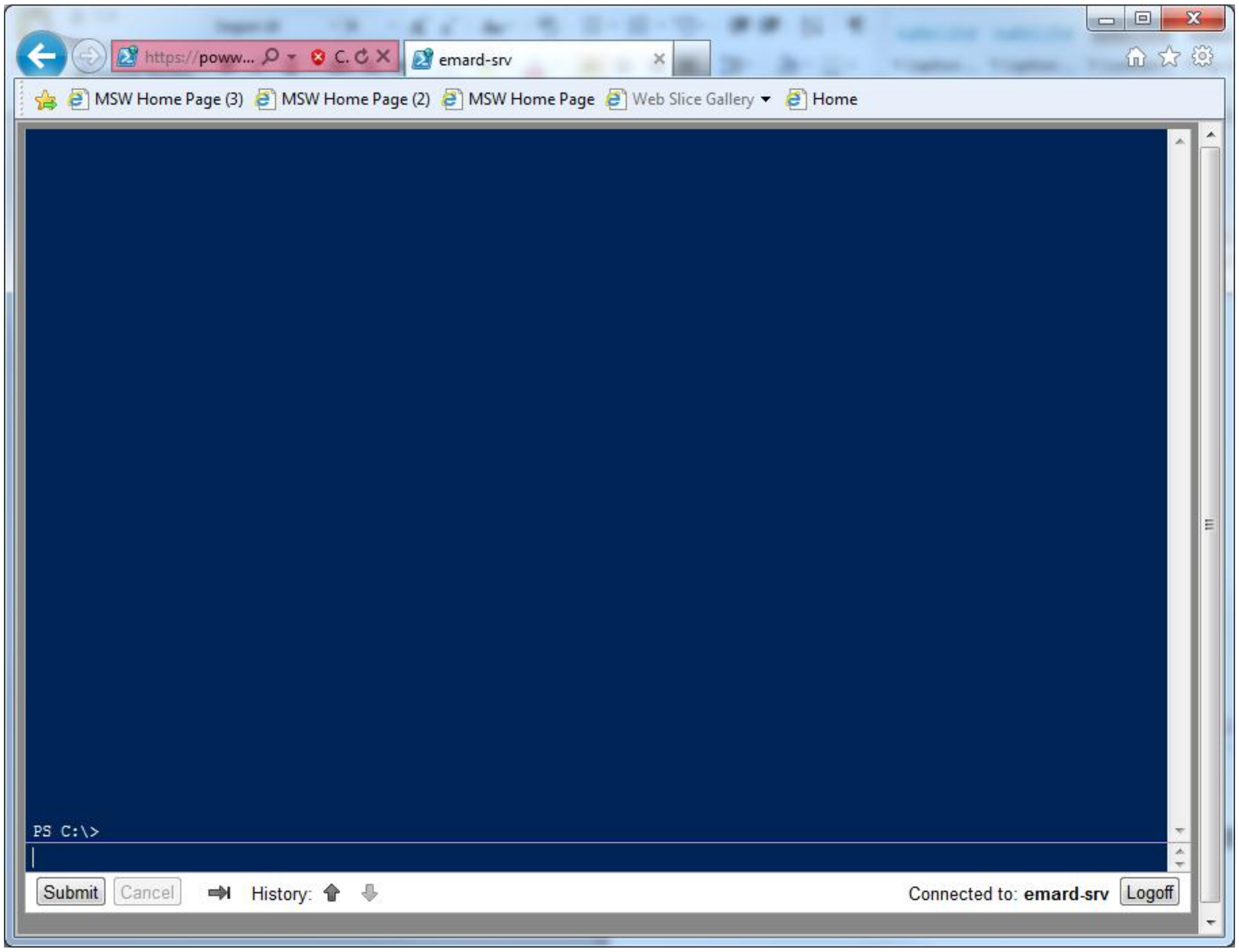
```
Description: Comment block
Path: Default

<#
 #
#>
```

Integrated Script Snippets in the ISE Editor

Windows PowerShell Web Access

Windows PowerShell Web Access is a new feature enabled by Windows Server 2012 RC that lets you manage Windows servers by using Windows PowerShell within a web browser. The target computers you want to manage can be running any version of Windows enabled for Windows PowerShell remoting.

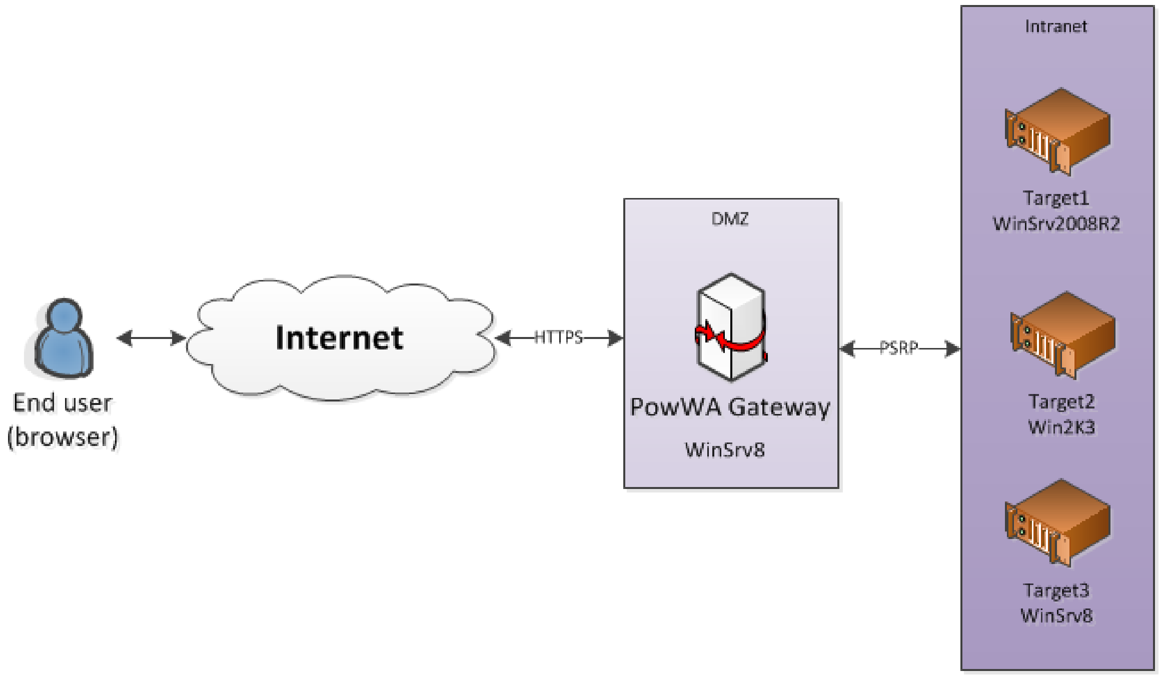


```
PS C:\>
```

Connected to: **emard-srv**

Windows PowerShell Web Access

To manage the remote server through Windows PowerShell in a web browser, you connect to a server running Windows Server 2012 RC on which the Windows PowerShell Web Access feature is installed. This server acts as a gateway that serves the web pages containing a Windows PowerShell interface to the remote clients. The following illustration shows the infrastructure.



PowWA infrastructure

Updatable Help

Windows PowerShell 2.0 included extensive Help topics that were frequently updated online. However, because the Help files were part of the Windows operating system, users couldn't update them, and the Help topics that were displayed at the command line could soon become outdated. Third-party products had to convert online Help to XML or display outdated Help topics.

In Windows PowerShell 3.0, new Update-Help and Save-Help cmdlets download and install the newest Help files for each module. The cmdlets find the Help files on the Internet, determining whether they are newer than local files, unpacking them, and installing them in the correct location. The updated files are ready for immediate use in Get-Help—you don't have to restart Windows PowerShell. Help files for Windows PowerShell 3.0 are guaranteed to be up to date on first use because they do not ship "in the box." Get-Help displays auto-generated Help for commands and then prompts you to use the Update-Help cmdlet to install or update the Help files for your modules.

For some environments, such as large enterprises behind Internet firewalls, it is preferable to be able to update Help files from a local share instead of from the Internet. In these cases, you can use Save-Help -DestinationPath <*share*> to create a local share that stores the latest Windows PowerShell Help files.

Users within the organization can then update their Help files by pointing to that share and running Update-Help –SourcePath *<share>.*

Updatable Help is available for all modules, including third-party modules, and includes support for multiple languages.

Session configuration files

Windows PowerShell simplifies the process of defining a new session configuration by allowing the administrator to specify the configuration in a declarative manner using name-value pairs in a PowerShell data file. For most settings, this is much simpler than writing a PowerShell script. It's also easier to understand how a session configuration is defined by inspecting the file.

RunAs capability

When remote administration is delegated, scenarios can result in which users lack the credentials required to perform needed tasks. With Windows PowerShell 3.0, administrators can configure sessions so that certain commands are run by default with the credentials of a different user. Credentials are stored securely in the WSMan provider.

For example, to change the credentials under which commands will be executed in the regular PowerShell endpoint you would execute the following:

```
cd WSMan:\localhost\Plugin\microsoft.powershell
$cred = Get-Credential
Set-Item .\RunAsUser $cred
```

You must restart the Windows Remote Management (WinRM) service for the changes to take effect.

Default parameter values

The new $**PSDefaultParameterValues** preference variable in Windows PowerShell 3.0 lets you specify default values for cmdlet parameters. You can set values for a parameter on a particular cmdlet or a set of cmdlets that match a wildcard expression.

The value of $**PSDefaultParameterValues** is a hash table that consists of a collection of key/value pairs. Each key consists of a command name and a parameter name separated by a colon. The command name and/or the parameter name can be enclosed in quotation marks ("CommandName":"ParameterName").

To override a default parameter value, add an explicit parameter value to the command. To disable the Default Parameter Value feature, enter the following key/value pair: "**Disable=$true**".

By default, the value of $**PSDefaultParameterValues** is session-specific. To set it for all Windows PowerShell sessions, add the $**PSDefaultParameterValues** variable to your Windows PowerShell profile.

Example:

```
$PSDefaultParameterValues=@{Invoke-
Command:ConfigurationName="AdminSession.PowerShell";*-Job:Verbose=$true}
```

New cmdlets

Windows PowerShell 3.0 includes many new cmdlets that expand its power and reach. The following is a partial list of new cmdlets included in Windows PowerShell 3.0.

Get-CimAssociatedInstance	Gets Common Information Model (CIM) instances connected to the given instance via an association.
Get-CimClass	Enables the user to enumerate the list of CIM Classes under a specific namespace.
Register-CimIndicationEvent	Subscribes to indications using the Filter Expression or Query Expression.
Get/New/Remove/Set-CimInstance	Gets, creates, removes, or edits a CIM instance on the server. For Get-CimInstance, the instance contains only the properties specified in the **Property** parameter, **KeyOnly** parameter, or the **Select** clause of the **Query** parameter.
Invoke-CimMethod	Invokes a method on a CIM object.
Get/New/Remove-CimSession	Gets, creates or removes a CIM session on the client representing a connection with a remote computer.
New-CimSessionOption	Creates an instance of a CimSessionOption, which can be used as an argument to the New-CimSession cmdlet.
Show-Command	Shows a graphical representation of a cmdlet as a Windows form.
Rename-Computer	Renames a computer.
Get/Show-ControlPanelItem	Gets a list of Control Panel applets installed on the local computer. Show-ControlPanelItem is used to launch the Control Panel applet.
Unblock-File	Removes the ZoneTransfer alternate NTFS stream; for example, the "Downloaded From Internet" stream.
Save/Update-Help	Save-Help exports the currently installed Help files to a location on the File System. Update-Help downloads Help files from the Internet or a file share and installs them on the local computer.
Resume/Suspend-Job	Suspends or resumes a job. These cmdlets currently only work with Workflow Jobs.

Add/Disable/Enable/Get/New/Remove/Set-JobTrigger	Manipulates job triggers that define when a scheduled job will execute.
ConvertFrom/ConvertTo-Json	Converts objects to/from a JSON-formatted string representation.
Connect/Disconnect/Receive-PSSession	Connects/disconnects from a remote session. Receive-PSSession resumes execution of a command in a disconnected session and gets the session output (implicitly reconnecting to the session).
New/Test-PSSessionConfigurationFile	Creates or validates a PSSession configuration file that can be used to create a constrained endpoint.
New-PSTransportOption	Creates a new PSTransportOption object.
New-PSWorkflowExecutionOption	Defines workflow endpoint configuration values to be used with Register-PSSessionConfiguration.
Invoke-RestMethod	Makes an HTTP or HTTPS request to a RESTful web service and returns the response.
Disable/Enable/Get/Register/Set/Unregister-ScheduledJob	Manipulates scheduled jobs on the computer.
Get/New/Set-ScheduledJobOption	Gets, creates, or sets an object that can be used to specify advanced configuration for Scheduled Jobs.
Get/Remove-TypeData	Gets or removes TypeData.
Invoke-WebRequest	Makes an HTTP or HTTPS request to a web service and returns the response.
New-WinEvent	Creates an event in the event log.

Summary

The following features of Windows PowerShell 3.0 offer comprehensive, resilient, and simple automation of your Windows Servers:

- More than two thousand new cmdlets that are easy to find and execute.
- Workflows that allow users to automate long running tasks across multiple computers in a resilient way.
- Disconnected sessions that allow you to start execution on a computer and return to it later (possibly from another computer).
- The ability to delegate a set of credentials that will be used when commands are run in certain sessions.

- Job scheduling that lets you run your scripts/workflows according to your defined schedule and stores results for later retrieval.

Requirements

The NAMS feature requires the following:

- Windows Server 2012 RC (on the server running IPAM).
- Windows Server 2008, Windows Server 2008 R2, or Windows Server 2012 RC (on the domain controllers, DHCP servers, DNS servers, and Network Policy Server [NPS] servers).
- A single Active Directory forest.
- A domain member computer to act as the NAMS server. You cannot install the IPAM feature on an Active Directory domain controller.

For more information about IP Address Management, see the white paper *Windows Server 2012 RC Networking*.

Conclusion

IT pros today face the challenge of managing and maintaining an increasing number of mission-critical servers and services, all with fewer resources. Windows Server 2012 RC addresses this problem by offering improvements to features, such as Server Manager and Windows PowerShell, and new features, such as IP Address Management Services, that help administrators manage a multi-server environment efficiently and cost effectively.