Install Nano Server

10/25/2021 • 4 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

Windows Server 2016 offers a new installation option: Nano Server. Nano Server is a remotely administered server operating system optimized for private clouds and datacenters. It is similar to Windows Server in Server Core mode, but significantly smaller, has no local logon capability, and only supports 64-bit applications, tools, and agents. It takes up far less disk space, sets up significantly faster, and requires far fewer updates and restarts than Windows Server. When it does restart, it restarts much faster. The Nano Server installation option is available for Standard and Datacenter editions of Windows Server 2016.

Nano Server is ideal for a number of scenarios:

- As a compute host for Hyper-V virtual machines, either in clusters or not
- As a storage host for Scale-Out File Server.
- As a DNS server
- As a web server running Internet Information Services (IIS)
- As a host for applications that are developed using cloud application patterns and run in a container or virtual machine guest operating system

Important differences in Nano Server

Because Nano Server is optimized as a lightweight operating system for running cloud-native applications based on containers and micro-services or as an agile and cost-effective datacenter host with a dramatically smaller footprint, there are important differences in Nano Server versus Server Core or Server with Desktop Experience installations:

- Nano Server is headless; there is no local logon capability or graphical user interface.
- Only 64-bit applications, tools, and agents are supported.
- Nano Server cannot serve as an Active Directory domain controller.
- Group Policy is not supported. However, you can use Desired State Configuration to apply settings at scale.
- Nano Server cannot be configured to use a proxy server to access the internet.
- NIC Teaming (specifically, load balancing and failover, or LBFO) is not supported. Switch-embedded teaming (SET) is supported instead.
- Microsoft Endpoint Configuration Manager and System Center Data Protection Manager are not supported.
- Best Practices Analyzer (BPA) cmdlets and BPA integration with Server Manager are not supported.
- Nano Server does not support virtual host bus adapters (HBAs).
- Nano Server does not need to be activated with a product key. When functioning as a Hyper-V host, Nano Server does not support Automatic Virtual Machine Activation (AVMA). Virtual machines running on a Nano Server host can be activated using Key Management Service (KMS) with a generic volume license key or

using Active Directory-based activation.

- The version of Windows PowerShell provided with Nano Server has important differences. For details, see PowerShell on Nano Server.
- Nano Server is supported only on the Current Branch for Business (CBB) model--there is no Long-Term Servicing Branch (LTSB) release for Nano Server at this time. See the following subsection for more information.

Current Branch for Business

Nano Server is serviced with a more active model, called Current Branch for Business (CBB), in order to support customers who are moving at a cloud cadence, using rapid development cycles. In this model, feature update releases of Nano Server are expected two to three times per year. This model requires Software Assurance for Nano Servers deployed and operated in production. To maintain support, administrators must stay no more than two CBB releases behind. However, these releases do not auto-update existing deployments; administrators perform manual installation of a new CBB release at their convenience. For some additional information, see Windows Server 2016 new Current Branch for Business servicing option.

The Server Core and Server with Desktop Experience installation options are still serviced on the Long-Term Servicing Branch (LTSB) model, comprising 5 years of mainstream support and 5 years of extended support.

Installation scenarios

Evaluation

You can obtain a 180-day-licensed evaluation copy of Windows Server from Windows Server Evaluations. To try out Nano Server, choose the **Nano Server | 64-bit EXE option**, and then come back to either Nano Server Quick Start or Deploy Nano Server to get started.

Clean installation

Because you install Nano Server by configuring a VHD, a clean installation is the quickest and simplest deployment method.

- To get started quickly with a basic deployment of Nano Server using DHCP to obtain an IP address, see the Nano Server Quick Start
- If you're already familiar with the basics of Nano Server, the more detailed topics starting with Deploy Nano Server offer a full set of instructions for customizing images, working with domains, installing packages for server roles and other features both online and offline, and much more.

> **IMPORTANT**
> Once Setup has completed and immediately after you have installed all of the server roles and features you need, check for and install updates available for Windows Server 2016. For Nano Server, see the Managing updates in Nano Server section of Manage Nano Server.

Upgrade

Since Nano Server is new for Windows Server 2016, there isn't an upgrade path from older operating system versions to Nano Server.

Migration

Since Nano Server is new for Windows Server 2016, there isn't migration path from older operating system versions to Nano Server.

If you need a different installation option, you can head back to the main Windows Server 2016 page

Changes to Nano Server in Windows Server Semi-Annual Channel

10/25/2021 • 2 minutes to read • Edit Online

> Applies to: Windows Server, Semi-Annual Channel

If you're already running Nano Server, the Window Server Semi-Annual Channel servicing model will be familiar, since it was formerly serviced by the Current Branch for Business (CBB) model. Windows Server Semi-Annual Channel is just a new name for the same model. In this model, feature update releases of Nano Server are expected two to three times per year.

However, starting with Windows Server, version 1803, Nano Server is available only as a **container base OS image**. You must run it as a container in a container host, such as a Server Core installation of Windows Server. Running a container based on Nano Server in this release differs from earlier releases in these ways:

- Nano Server has been optimized for .NET Core applications.
- Nano Server is even smaller than the Windows Server 2016 version.
- PowerShell Core, .NET Core, and WMI are no longer included by default, but you can include PowerShell Core and .NET Core container packages when building your container.
- There is no longer a servicing stack included in Nano Server. Microsoft publishes an updated Nano container to Docker Hub that you redeploy.
- You troubleshoot the new Nano Container by using Docker.
- You can now run Nano containers on IoT Core.

Nano Server Quick Start

10/25/2021 • 5 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

Follow the steps in this section to get started quickly with a basic deployment of Nano Server using DHCP to obtain an IP address. You can run a Nano Server VHD either in a virtual machine or boot to it on a physical computer; the steps are slightly different.

Once you've tried out the basics with these quick-start steps, you can find details of creating your own custom images, package management by several methods, domain operations, and much more in Deploy Nano Server.

Nano Server in a virtual machine

Follow these steps to create a Nano Server VHD that will run in a virtual machine.

To quickly deploy Nano Server in a virtual machine

1. Copy *NanoServerImageGenerator* folder from the \NanoServer folder in the Windows Server 2016 ISO to a folder on your hard drive.
2. Start Windows PowerShell as an administrator, change directory to the folder where you have placed the NanoServerImageGenerator folder and then import the module with
   `Import-Module .\NanoServerImageGenerator -Verbose`

   NOTE

   You might have to adjust the Windows PowerShell execution policy. `Set-ExecutionPolicy RemoteSigned` should work well.
3. Create a VHD for the Standard edition that sets a computer name and includes the Hyper-V **guest drivers** by running the following command which will prompt you for an administrator password for the new VHD:

   ```
   New-NanoServerImage -Edition Standard -DeploymentType Guest -MediaPath <path to root of media> -
   BasePath .\Base -TargetPath .\NanoServerVM\NanoServerVM.vhd -ComputerName <computer name>
   ```

   where

   - **-MediaPath <path to root of media>** specifies a path to the root of the contents of the Windows Server 2016 ISO. For example if you have copied the contents of the ISO to d:\TP5ISO you would use that path.
   - **-BasePath** (optional) specifies a folder that will be created to copy the Nano Server WIM and packages to.
   - **-TargetPath** specifies a path, including the filename and extension, where the resulting VHD or VHDX will be created.

- **Computer_name** specifies the computer name that the Nano Server virtual machine you are creating will have.

Example:

```
New-NanoServerImage -Edition Standard -DeploymentType Guest -MediaPath f:\ -BasePath .\Base -
TargetPath .\Nano1\Nano.vhd -ComputerName Nano1
```

This example creates a VHD from an ISO mounted as f:\. When creating the VHD it will use a folder called Base in the same directory where you ran New-NanoServerImage; it will place the VHD (called Nano.vhd) in a folder called Nano1 in the folder from where the command is run. The computer name will be Nano1. The resulting VHD will contain the Standard edition of Windows Server 2016 and will be suitable for Hyper-V virtual machine deployment. If you want a Generation 1 virtual machine, create a VHD image by specifying a **.vhd** extension for -TargetPath. For a Generation 2 virtual machine, create a VHDX image by specifying a **.vhdx** extension for -TargetPath. You can also directly generate a WIM file by specifying a **.wim** extension for -TargetPath.

> **NOTE**
> New-NanoServerImage is supported on Windows 8.1, Windows 10, Windows Server 2012 R2, and Windows Server 2016.

4. In Hyper-V Manager, create a new virtual machine and use the VHD created in Step 3.
5. Boot the virtual machine and in Hyper-V Manager connect to the virtual machine.
6. Log on to the Recovery Console (see the Nano Server Recovery Console section in this guide), using the administrator and password you supplied while running the script in Step 3.

> **NOTE**
> The Recovery Console only supports basic keyboard functions. Keyboard lights, 10-key sections, and keyboard layout switching such as caps lock and number lock are not supported.

7. Obtain the IP address of the Nano Server virtual machine and use Windows PowerShell remoting or other remote management tool to connect to and remotely manage the virtual machine.

Nano Server on a physical computer

You can also create a VHD that will run Nano Server on a physical computer, using the pre-installed device drivers. If your hardware requires a driver that is not already provided in order to boot or connect to a network, follow the steps in the Adding Additional Drivers section of this guide.

To quickly deploy Nano Server on a physical computer

1. Copy *NanoServerImageGenerator* folder from the \NanoServer folder in the Windows Server 2016 ISO to a folder on your hard drive.
2. Start Windows PowerShell as an administrator, change directory to the folder where you have placed the NanoServerImageGenerator folder and then import the module with
   `Import-Module .\NanoServerImageGenerator -Verbose`

> **NOTE**
> You might have to adjust the Windows PowerShell execution policy. `Set-ExecutionPolicy RemoteSigned` should work well.

3. Create a VHD that sets a computer name and includes the OEM drivers and Hyper-V by running the following command which will prompt you for an administrator password for the new VHD:

```
New-NanoServerImage -Edition Standard -DeploymentType Host -MediaPath <path to root of media> -
BasePath .\Base -TargetPath .\NanoServerPhysical\NanoServer.vhd -ComputerName <computer name> -
OEMDrivers -Compute -Clustering
```

   where

   - **-MediaPath <path to root of media>** specifies a path to the root of the contents of the Windows Server 2016 ISO. For example if you have copied the contents of the ISO to d:\TP5ISO you would use that path.
   - **BasePath** specifies a folder that will be created to copy the Nano Server WIM and packages to. (This parameter is optional.)
   - **TargetPath** specifies a path, including the filename and extension, where the resulting VHD or VHDX will be created.
   - **Computer_name** is the computer name for the Nano Server you are creating.

   Example:

```
New-NanoServerImage -Edition Standard -DeploymentType Host -MediaPath F:\ -BasePath .\Base -TargetPath
.\Nano1\NanoServer.vhd -ComputerName Nano-srv1 -OEMDrivers -Compute -Clustering
```

   This example creates a VHD from an ISO mounted as F:\. When creating the VHD it will use a folder called Base in the same directory where you ran New-NanoServerImage; it will place the VHD in a folder called Nano1 in the folder from where the command is run. The computer name will be Nano-srv1 and will have OEM drivers installed for most common hardware and has the Hyper-V role and clustering feature enabled. The Standard Nano edition is used.

4. Log in as an administrator on the physical server where you want to run the Nano Server VHD.
5. Copy the VHD that this script creates to the physical computer and configure it to boot from this new VHD. To do that, follow these steps:

   a. Mount the generated VHD. In this example, it's mounted under D:\.

   b. Run **bcdboot d:\windows**.

   c. Unmount the VHD.

6. Boot the physical computer into the Nano Server VHD.
7. Log on to the Recovery Console (see the Nano Server Recovery Console section in this guide), using the administrator and password you supplied while running the script in Step 3.

   > **NOTE**
   > The Recovery Console only supports basic keyboard functions. Keyboard lights, 10-key sections, and keyboard layout switching such as caps lock and number lock are not supported.

8. Obtain the IP address of the Nano Server computer and use Windows PowerShell remoting or other remote management tool to connect to and remotely manage the virtual machine.

Deploy Nano Server

10/25/2021 • 28 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

This topic covers information you need to deploy Nano Server images that are more customized to your needs compared to the simple examples in the Nano Server Quick Start topic. You'll find information about making a custom Nano Server image with exactly the features you want, installing Nano Server images from VHD or WIM, editing files, working with domains, dealing with packages by several methods, and working with server roles.

Nano Server Image Builder

The Nano Server Image Builder is a tool that helps you create a custom Nano Server image and bootable USB media with the aid of a graphical interface. Based on the inputs you provide, it generates reusable PowerShell scripts that allow you easily automate consistent installations of Nano Server running either Windows Server 2016 Datacenter or Standard editions.

Obtain the tool from the Download Center.

The tool also requires Windows Assessment and Deployment Kit (ADK).

Nano Server Image Builder creates customized Nano Server images in VHD, VHDX, or ISO formats and can create bootable USB media to deploy Nano server or detect the hardware configuration of a server. It also can do the following:

- Accept the license terms
- Create VHD, VHDX or ISO formats
- Add server roles
- Add device drivers
- Set machine name, administrator password, logfile path, and timezone
- Join a domain by using an existing Active Directory account or a harvested domain-join blob
- Enable WinRM for communication outside the local subnet
- Enable Virtual LAN IDs and configure static IP addresses
- Inject new servicing packages on the fly
- Add a setupcomplete.cmd or other customer scripts to run after the unattend.xml is processed
- Enable Emergency Management Services (EMS) for serial port console access
- Enable development services to enable test signed drivers and unsigned applications, PowerShell default shell
- Enable debugging over serial, USB, TCP/IP, or IEEE 1394 protocols
- Create USB media using WinPE that will partition the server and install the Nano image
- Create USB media using WinPE that will detect your existing Nano Server hardware configuration and report all the details in a log and on-screen. This includes network adapters, MAC addresses, and firmware Type

(BIOS or UEFI). The detection process will also list all of the volumes on the system and the devices that do not have a driver included in the Server Core drivers package.

If any of these are unfamiliar to you, review the remainder of this topic and the other Nano Server topics so that you'll be prepared to provide the tool with the information it will need.

Creating a custom Nano Server image

For Windows Server 2016, Nano Server is distributed on the physical media, where you will find a **NanoServer** folder; this contains a .wim image and a sub-folder called **Packages**. It is these package files that you use to add server roles and features to the VHD image, which you then boot to.

You can also find and install these packages with the NanoServerPackage provider of PackageManagement (OneGet) PowerShell module. See the Installing roles and features online section of this topic.

This table shows the roles and features that are available in this release of Nano Server, along with the Windows PowerShell options that will install the packages for them. Some packages are installed directly with their own Windows PowerShell switches (such as -Compute); others you install by passing package names to the -Package parameter, which you can combine in a comma-separated list. You can dynamically list available packages using the Get-NanoServerPackage cmdlet.

ROLE OR FEATURE	OPTION
Hyper-V role (including NetQoS)	-Compute
Failover Clustering and other components, detailed after this table	-Clustering
Basic drivers for a variety of network adapters and storage controllers. This is the same set of drivers included in a Server Core installation of Windows Server 2016.	-OEMDrivers
File Server role and other storage components, detailed after this table	-Storage
Windows Defender, including a default signature file	-Defender
Reverse forwarders for application compatibility, for example common application frameworks such as Ruby, Node.js, etc.	Now included by default
DNS Server role	-Package Microsoft-NanoServer-DNS-Package
PowerShell Desired State Configuration (DSC)	-Package Microsoft-NanoServer-DSC-Package **Note:** For full details, see Using DSC on Nano Server.
Internet Information Server (IIS)	-Package Microsoft-NanoServer-IIS-Package **Note:** See IIS on Nano Server for details about working with IIS.
Host support for Windows Containers	-Containers

ROLE OR FEATURE	OPTION
System Center Virtual Machine Manager agent	-Package Microsoft-NanoServer-SCVMM-Package -Package Microsoft-NanoServer-SCVMM-Compute-Package **Note:** Use the SCVMM Compute package only if you are monitoring Hyper-V. For hyper-converged deployments in VMM, you should also specify the -Storage parameter. For more details, see the VMM documentation.
System Center Operations Manager agent	Installed separately. See the System Center Operations Manager documentation for more details at https://technet.microsoft.com/system-center-docs/om/manage/install-agent-on-nano-server.
Data Center Bridging (including DCBQoS)	-Package Microsoft-NanoServer-DCB-Package
Deploying on a virtual machine	-Package Microsoft-NanoServer-Guest-Package
Deploying on a physical machine	- Package Microsoft-NanoServer-Host-Package
BitLocker, trusted platform module (TPM), volume encryption, platform identification, cryptography providers, and other functionality related to secure startup	-Package Microsoft-NanoServer-SecureStartup-Package
Hyper-V support for Shielded VMs	-Package Microsoft-NanoServer-ShieldedVM-Package **Note:** This package is only available for the Datacenter edition of Nano Server.
Simple Network Management Protocol (SNMP) agent	-Package Microsoft-NanoServer-SNMP-Agent-Package.cab **Note:** Not included with Windows Server 2016 installation media. Available online only. See Installing roles and features online for details.
IPHelper service which provides tunnel connectivity using IPv6 transition technologies (6to4, ISATAP, Port Proxy, and Teredo), and IP-HTTPS	-Package Microsoft-NanoServer-IPHelper-Service-Package.cab **Note:** Not included with Windows Server 2016 installation media. Available online only. See Installing roles and features online for details.

NOTE

When you install packages with these options, a corresponding language pack is also installed based on selected server media locale. You can find the available language packs and their locale abbreviations in the installation media in sub-folders named for the locale of the image.

NOTE

When you use the -Storage parameter to install File Services, File Services is not actually enabled. Enable this feature from a remote computer with Server Manager.

Failover Clustering items installed by the -Clustering parameter

- Failover Clustering role
- VM Failover Clustering
- Storage Spaces Direct (S2D)
- Storage Quality of Service
- Volume Replication Clustering
- SMB Witness Service

File and storage items installed by the -Storage parameter

- File Server role
- Data Deduplication
- Multipath I/O, including a driver for Microsoft Device-Specific Module (MSDSM)
- ReFS (v1 and v2)
- iSCSI Initiator (but not iSCSI Target)
- Storage Replica
- Storage Management Service with SMI-S support
- SMB Witness Service
- Dynamic Volumes
- Basic Windows storage providers (for Windows Storage Management)

Installing a Nano Server VHD

This example creates a GPT-based VHDX image with a given computer name and including Hyper-V guest drivers, starting with Nano Server installation media on a network share. In an elevated Windows PowerShell prompt, start with this cmdlet:

```
Import-Module <Server media location>\NanoServer\NanoServerImageGenerator; New-NanoServerImage -
DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\server_en-us -BasePath .\Base -TargetPath
.\FirstStepsNano.vhdx -ComputerName FirstStepsNano
```

The cmdlet will accomplish all of these tasks:

1. Select Standard as a base edition
2. Prompt you for the Administrator password
3. Copy installation media from \\Path\To\Media\server_en-us into .\Base
4. Convert the WIM image to a VHD. (The file extension of the target path argument determines whether it creates an MBR-based VHD for Generation 1 virtual machines versus a GPT-based VHDX for Generation 2 virtual machines.)
5. Copy the resulting VHD into .\FirstStepsNano.vhdx
6. Set the Administrator password for the image as specified
7. Set the computer name of the image to FirstStepsNano
8. Install the Hyper-V guest drivers

All of this results in an image of .\FirstStepsNano.vhdx.

The cmdlet generates a log as it runs and will let you know where this log is located once it is finished. The WIM-to-VHD conversion accomplished by the companion script generates its own log in %TEMP%\Convert-WindowsImage\<GUID> (where <GUID> is a unique identifier per conversion session).

As long as you use the same base path, you can omit the media path parameter every time you run this cmdlet, since it will use cached files from the base path. If you don't specify a base path, the cmdlet will generate a default one in the TEMP folder. If you want to use different source media, but the same base path, you should specify the media path parameter, however.

> **NOTE**
> You now have the option to specify the Nano Server edition to build either the Standard or Datacenter edition. Use the -Edition parameter to specify *Standard* or *Datacenter* editions.

Once you have an existing image, you can modify it as needed using the *Edit-NanoServerImage* cmdlet.

If you do not specify a computer name, a random name will be generated.

Installing a Nano Server WIM

1. Copy the *NanoServerImageGenerator* folder from the \NanoServer folder in the Windows Server 2016 ISO a local folder on your computer.
2. Start Windows PowerShell as an administrator, change directory to the folder where you placed the NanoServerImageGenerator folder and then import the module with `Import-Module .\NanoServerImageGenerator -Verbose`.

   > **NOTE**
   > You might have to adjust the Windows PowerShell execution policy. `Set-ExecutionPolicy RemoteSigned` should work well.

To create a Nano Server image to serve as a Hyper-V host, run the following:

```
New-NanoServerImage -Edition Standard -DeploymentType Host -MediaPath <path to root of media> -BasePath
.\Base -TargetPath .\NanoServerPhysical\NanoServer.wim -ComputerName <computer name> -OEMDrivers -Compute -
Clustering`
```

Where

- MediaPath is the root of the DVD media or ISO image containing Windows Server 2016 .
- `-BasePath` will contain a copy of the Nano Server binaries, so you can use New-NanoServerImage -BasePath without having to specify -MediaPath in future runs.
- `-TargetPath` will contain the resulting .wim file containing the roles & features you selected. Make sure to specify the .wim extension.
- `-Compute` adds the Hyper-V role.
- `-OemDrivers` adds a number of common drivers.

You will be prompted to enter an administrator password.

For more information, run `Get-Help New-NanoServerImage -Full`.

Boot into WinPE and ensure that the .wim file just created is accessible from WinPE. (You could, for example, copy the .wim file to a bootable WinPE image on a USB flash drive.)

Once WinPE boots, use Diskpart.exe to prepare the target computer's hard drive. Run the following Diskpart commands (modify accordingly, if you're not using UEFI & GPT):

WARNING

These commands will delete all data on the hard drive:

```
Diskpart.exe
Select disk 0
Clean
Convert GPT
Create partition efi size=100
Format quick FS=FAT32 label=System
Assign letter=s
Create partition msr size=128
Create partition primary
Format quick FS=NTFS label=NanoServer
Assign letter=n
List volume
Exit
```

Apply the Nano Server image (adjust the path of the .wim file):

```
Dism.exe /apply-image /imagefile:.\NanoServer.wim /index:1 /applydir:n:\
Bcdboot.exe n:\Windows /s s:
```

Remove the DVD media or USB drive and reboot your system with **Wpeutil.exe Reboot**

Editing files on Nano Server locally and remotely

In either case, connect to Nano Server, such as with Windows PowerShell remoting.

Once you've connected to Nano Server, you can edit a file residing on your local computer by passing the file's relative or absolute path to the psEdit command, for example: `psEdit C:\Windows\Logs\DISM\dism.log` or `psEdit .\myScript.ps1`

Edit a file residing on the remote Nano Server by starting a remote session with `Enter-PSSession -ComputerName 192.168.0.100 -Credential ~\Administrator` and then passing the file's relative or absolute path to the psEdit command like this: `psEdit C:\Windows\Logs\DISM\dism.log`

Installing roles and features online

NOTE

If you install an optional Nano Server package from media or online repository, it won't have recent security fixes included. To avoid a version mismatch between the optional packages and base operating system, you should install the latest cumulative update immediately after installing any optional packages and **before** restarting the server.

Installing roles and features from a package repository

You can find and install Nano Server packages from the online package repository by using the NanoServerPackage provider of the PackageManagement PowerShell module. To install this provider, use these cmdlets:

```
Install-PackageProvider NanoServerPackage
Import-PackageProvider NanoServerPackage
```

> **NOTE**
> If you experience errors when running Install-PackageProvider, check that you have installed the latest cumulative update (KB3206632 or later), or use Save-Module as follows:

```
Save-Module -Path $Env:ProgramFiles\WindowsPowerShell\Modules\ -Name NanoServerPackage -MinimumVersion
1.0.1.0
Import-PackageProvider NanoServerPackage
```

Once this provider is installed and imported, you can search for, download, and install Nano Server packages using cmdlets designed specifically for working with Nano Server packages:

```
Find-NanoServerPackage
Save-NanoServerPackage
Install-NanoServerPackage
```

You can also use the generic PackageManagement cmdlets and specify the NanoServerPackage provider:

```
Find-Package -ProviderName NanoServerPackage
Save-Package -ProviderName NanoServerPackage
Install-Package -ProviderName NanoServerPackage
Get-Package -ProviderName NanoServerPackage
```

To use any of these cmdlets with Nano Server packages on Nano Server, add `-ProviderName NanoServerPackage`. If you don't add the -ProviderName parameter, PackageManagement will iterate all of the providers. For more details on these cmdlets, run `Get-Help <cmdlet>`. Here are some common usage examples:

Searching for Nano Server packages

You can use either `Find-NanoServerPackage` or `Find-Package -ProviderName NanoServerPackage` to search for and return a list of Nano Server packages that are available in the online repository. For example, you can get a list of all the latest packages:

```
Find-NanoServerPackage
```

Running `Find-Package -ProviderName NanoServerPackage -DisplayCulture` displays all available cultures.

If you need a specific locale version, such as US English, you could use `Find-NanoServerPackage -Culture en-us` or `Find-Package -ProviderName NanoServerPackage -Culture en-us` or `Find-Package -Culture en-us -DisplayCulture`.

To find a specific package by package name, use the -Name parameter. This parameter also accepts wildcards. For example, to find all packages with VMM in the name, use `Find-NanoServerPackage -Name *VMM*` or `Find-Package -ProviderName NanoServerPackage -Name *VMM*`.

You can find a particular version with the -RequiredVersion, -MinimumVersion, or -MaximumVersion parameters. To find all available versions, use -AllVersions. Otherwise, only the latest version is returned. For example: `Find-NanoServerPackage -Name *VMM* -RequiredVersion 10.0.14393.0`. Or, for all versions: `Find-Package -ProviderName NanoServerPackage -Name *VMM* -AllVersions`

Installing Nano Server packages

You can install a Nano Server package (including its dependency packages, if any) to Nano Server either locally or an offline image with either `Install-NanoServerPackage` or `Install-Package -ProviderName NanoServerPackage`.

Both of these accept input from the pipeline.

To install the latest version of a Nano Server package to an online Nano Server, use either `Install-NanoServerPackage -Name Microsoft-NanoServer-Containers-Package` or `Install-Package -Name Microsoft-NanoServer-Containers-Package`. PackageManagement will use the culture of the Nano Server.

You can install a Nano Server package to an offline image while specifying a particular version and culture, like this:

```
Install-NanoServerPackage -Name Microsoft-NanoServer-DCB-Package -Culture de-de -RequiredVersion 10.0.14393.0
-ToVhd C:\MyNanoVhd.vhd
```

or:

```
Install-Package -Name Microsoft-NanoServer-DCB-Package -Culture de-de -RequiredVersion 10.0.14393.0 -ToVhd
C:\MyNanoVhd.vhd
```

Here are some examples of pipelining package search results to the installation cmdlet:

`Find-NanoServerPackage *dcb* | Install-NanoServerPackage` finds any packages with dcb in the name and then installs them.

`Find-Package *nanoserver-compute-* | Install-Package` finds packages with nanoserver-compute- in the name and installs them.

`Find-NanoServerPackage -Name *nanoserver-compute* | Install-NanoServerPackage -ToVhd C:\MyNanoVhd.vhd` finds packages with compute in the name and installs them to an offline image.

`Find-Package -ProviderName NanoserverPackage *nanoserver-compute-* | Install-Package -ToVhd C:\MyNanoVhd.vhd` does the same thing with any package that has nanoserver-compute- in the name.

Downloading Nano Server packages

`Save-NanoServerPackage` or `Save-Package` allow you to download packages and save them without installing them. Both cmdlets accept input from the pipeline.

For example, to download and save a Nano Server package to a directory that matches the wildcard path, use `Save-NanoServerPackage -Name Microsoft-NanoServer-DNS-Package -Path C:\` In this example, -Culture wasn't specified, so the culture of the local machine will be used. No version was specified, so the latest version will be saved.

```
Save-Package -ProviderName NanoServerPackage -Name Microsoft-NanoServer-IIS-Package -Path C:\ -Culture it-IT
-MinimumVersion 10.0.14393.0
```

saves a particular version and for the Italian language and locale.

You can send search results through the pipeline as in these examples:

```
Find-NanoServerPackage -Name *containers* -MaximumVersion 10.2 -MinimumVersion 1.0 -Culture es-ES | Save-
NanoServerPackage -Path C:\
```

or

```
Find-Package -ProviderName NanoServerPackage -Name *shield* -Culture es-ES | Save-Package -Path
```

Inventory installed packages

You can discover which Nano Server packages are installed with `Get-Package`. For example, see which packages are on Nano Server with `Get-Package -ProviderName NanoserverPackage`.

To check the Nano Server packages that are installed in an offline image, run `Get-Package -ProviderName NanoserverPackage -FromVhd C:\MyNanoVhd.vhd`.

Installing roles and features from local source

Though offline installation of server roles and other packages is recommended, you might need to install them online (with the Nano Server running) in container scenarios. To do this, follow these steps:

1. Copy the Packages folder from the installation media locally to the running Nano Server (for example, to C:\packages).

2. Create a new Unattend.xml file on another computer and then copy it to Nano Server. You can copy and paste this XML content into the XML file you created (this example shows installing the IIS package):

```
<?xml version=1.0 encoding=utf-8?>
    <unattend xmlns=urn:schemas-microsoft-com:unattend>
    <servicing>
        <package action=install>
            <assemblyIdentity name=Microsoft-NanoServer-IIS-Feature-Package version=10.0.14393.0
processorArchitecture=amd64 publicKeyToken=31bf3856ad364e35 language=neutral />
            <source location=c:\packages\Microsoft-NanoServer-IIS-Package.cab />
        </package>
        <package action=install>
            <assemblyIdentity name=Microsoft-NanoServer-IIS-Feature-Package version=10.0.14393.0
processorArchitecture=amd64 publicKeyToken=31bf3856ad364e35 language=en-US />
            <source location=c:\packages\en-us\Microsoft-NanoServer-IIS-Package_en-us.cab />
        </package>
    </servicing>
    <cpi:offlineImage cpi:source= xmlns:cpi=urn:schemas-microsoft-com:cpi />
</unattend>
```

3. In the new XML file you created (or copied), edit C:\packages to the directory you copied the content of Packages to.

4. Switch to the directory with the newly created XML file and run:
   `dism /online /apply-unattend:.\unattend.xml`

5. Confirm that the package and its associated language pack is installed correctly by running:
   `dism /online /get-packages`

   You should see Package Identity : Microsoft-NanoServer-IIS-Package~31bf3856ad364e35~amd64~en-US~10.0.10586.0 listed twice, once for Release Type : Language Pack and once for Release Type : Feature Pack.

Customizing an existing Nano Server VHD

You can change the details of an existing VHD by using the Edit-NanoServerImage cmdlet, as in this example:
`Edit-NanoServerImage -BasePath .\Base -TargetPath .\BYOVHD.vhd`

This cmdlet does the same things as New-NanoServerImage, but changes the existing image instead of converting a WIM to a VHD. It supports the same parameters as New-NanoServerImage with the exception of -MediaPath and -MaxSize, so the initial VHD must have been created with those parameters before you can make changes with Edit-NanoServerImage.

Additional tasks you can accomplish with New-NanoServerImage and Edit-NanoServerImage

Joining domains

New-NanoServerImage offers two methods of joining a domain; both rely on offline domain provisioning, but one harvests a blob to accomplish the join. In this example, the cmdlet harvests a domain blob for the Contoso domain from the local computer (which of course must be part of the Contoso domain), then it performs offline provisioning of the image using the blob:

```
New-NanoServerImage -Edition Standard -DeploymentType Host -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\JoinDomHarvest.vhdx -ComputerName JoinDomHarvest -DomainName Contoso
```

When this cmdlet completes, you should find a computer named JoinDomHarvest in the Active Directory computer list.

You can also use this cmdlet on a computer that is not joined to a domain. To do this, harvest a blob from any computer that is joined to the domain, and then provide the blob to the cmdlet yourself. Note that when you harvest such a blob from another computer, the blob already includes that computer's name--so if you try to add the *-ComputerName* parameter, an error will result.

You can harvest the blob with this command:

```
djoin
/Provision
/Domain Contoso
/Machine JoiningDomainsNoHarvest
/SaveFile JoiningDomainsNoHarvest.djoin
```

Run New-NanoServerImage using the harvested blob:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\JoinDomNoHrvest.vhd -DomainBlobPath .\Path\To\Domain\Blob\JoinDomNoHrvestContoso.djoin
```

In the event that you already have a node in the domain with the same computer name as your future Nano Server, you could reuse the computer name by adding the `-ReuseDomainNode` parameter.

Adding additional drivers

Nano Server offers a package that includes a set of basic drivers for a variety of network adapters and storage controllers; it's possible that drivers for your network adapters might not be included. You can use these steps to find drivers in a working system, extract them, and then add them to the Nano Server image.

1. Install Windows Server 2016 on the physical computer where you will run Nano Server.
2. Open Device Manager and identify devices in the following categories:
3. Network adapters
4. Storage controllers
5. Disk drives
6. For each device in these categories, right-click the device name, and click **Properties**. In the dialog that opens, click the **Driver** tab, and then click **Driver Details**.
7. Note the filename and path of the driver file that appears. For example, let's say the driver file is e1i63x64.sys, which is in C:\Windows\System32\Drivers.
8. In a command prompt, search for the driver file and search for all instances with dir e1i*.sys /s /b. In this example, the driver file is also present in the path C:\Windows\System32\DriverStore\FileRepository\net1ic64.inf_amd64_fafa7441408bbecd\e1i63x64.sys.
9. In an elevated command prompt, navigate to the directory where the Nano Server VHD is and run the following commands:

```
md mountdir
dism\dism /Mount-Image /ImageFile:.\NanoServer.vhd /Index:1 /MountDir:.\mountdir
dism\dism /Add-Driver /image:.\mountdir /driver:
C:\Windows\System32\DriverStore\FileRepository\net1ic64.inf_amd64_fafa7441408bbecd
dism\dism /Unmount-Image /MountDir:.\MountDir /Commit
```

10. Repeat these steps for each driver file you need.

> **NOTE**
> In the folder where you keep your drivers, both the SYS files and corresponding INF files must be present. Also, Nano Server only supports signed, 64-bit drivers.

Injecting drivers

Nano Server offers a package that includes a set of basic drivers for a variety of network adapters and storage controllers; it's possible that drivers for your network adapters might not be included. You can use this syntax to have New-NanoServerImage search the directory for available drivers and inject them into the Nano Server image:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\InjectingDrivers.vhdx -DriverPath .\Extra\Drivers
```

> **NOTE**
> In the folder where you keep your drivers, both the SYS files and corresponding INF files must be present. Also, Nano Server only supports signed, 64-bit drivers.

Using the -DriverPath parameter, you can also pass a array of paths to driver .inf files:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\InjectingDrivers.vhdx -DriverPath .\Extra\Drivers\netcard64.inf
```

Connecting with WinRM

To be able to connect to a Nano Server computer using Windows Remote Management (WinRM) (from another computer that is not on the same subnet), open port 5985 for inbound TCP traffic on the Nano Server image. Use this cmdlet:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\ConnectingOverWinRM.vhd -EnableRemoteManagementPort
```

Setting static IP addresses

To configure a Nano Server image to use static IP addresses, first find the name or index of the interface you want to modify by using Get-NetAdapter, netsh, or the Nano Server Recovery Console. Use the -Ipv6Address, -Ipv6Dns, -Ipv4Address, -Ipv4SubnetMask, -Ipv4Gateway and -Ipv4Dns parameters to specify the configuration, as in this example:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\StaticIpv4.vhd -InterfaceNameOrIndex Ethernet -Ipv4Address 192.168.1.2 -Ipv4SubnetMask
255.255.255.0 -Ipv4Gateway 192.168.1.1 -Ipv4Dns 192.168.1.1
```

Custom image size

You can configure the Nano Server image to be a dynamically expanding VHD or VHDX with the -MaxSize parameter, as in this example:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\BigBoss.vhd -MaxSize 100GB
```

Embedding custom data

To embed your own script or binaries in the Nano Server image, use the -CopyPath parameter to pass an array of files and directories to be copied. The -CopyPath parameter can also accept a hashtable to specify the destination path for files and directories.

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\BigBoss.vhd -CopyPath .\tools
```

Running custom commands after the first boot

To run custom commands as part of setupcomplete.cmd, use the -SetupCompleteCommand parameter to pass an array of commands:

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -SetupCompleteCommand @(echo foo, echo bar)
```

Running custom PowerShell scripts as part of image creation

To run custom PowerShell scripts as part of the image creation process, use the -OfflineScriptPath parameter to pass an array of paths to .ps1 scripts. If those scripts take arguments, use the -OfflineScriptArgument to pass a hashtable of additional arguments to the scripts.

```
New-NanoServerImage -DeploymentType Host -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -OfflineScriptPath C:\MyScripts\custom.ps1 -OfflineScriptArgument
@{Param1=Value1; Param2=Value2}
```

Support for development scenarios

If you want to develop and test on Nano Server, you can use the -Development parameter. This will enable PowerShell as the default local shell, enable installation of unsigned drivers, copy debugger binaries, open a port for debugging, enable test signing, and enable installation of AppX packages without a developer license:

```
New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -Development
```

Custom unattend file

If you want to use your own unattend file, use the -UnattendPath parameter:

```
New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -UnattendPath \\path\to\unattend.xml
```

Specifying an administrator password or computer name in this unattend file will override the values set by -AdministratorPassword and -ComputerName.

> **NOTE**
>
> Nano Server does not support setting TCP/IP settings via unattend files. You can use Setupcomplete.cmd to configure TCP/IP settings.

Collecting log files

If you want to collect the log files during image creation, use the -LogPath parameter to specify a directory where all the log files are copied.

```
New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -LogPath C:\Logs
```

> **NOTE**
>
> Some parameters on New-NanoServerImage and Edit-NanoServerImage are for internal use only and can be safely ignored. These include the -SetupUI and -Internal parameters.

Windows Server App installer

Windows Server App (WSA) installer provides a reliable installation option for Nano Server. Since Windows Installer (MSI) is not supported on Nano Server, WSA is also the only installation technology available for non-Microsoft products. WSA leverages Windows app package technology designed to install and service applications safely and reliably, using a declarative manifest. It extends the Windows app package installer to support Windows Server-specific extensions, with the limitation that WSA does not support installing drivers.

Creating and installing a WSA package on Nano Server involves steps for both the publisher and the consumer of the package.

The package publisher should do the following:

1. Install Windows 10 SDK, which includes the tools needed to create a WSA package: MakeAppx, MakeCert, Pvk2Pfx, SignTool.
2. Declare a manifest: Follow the WSA manifest extension schema to create the manifest file, AppxManifest.xml.
3. Use the **MakeAppx** tool to create a WSA package.
4. Use **MakeCert** and **Pvk2Pfx** tools to create the certificate, and then use **Signtool** to sign the package.

Next, the package consumer should follow these steps:

1. Run the *Import-Certificate* PowerShell cmdlet to import the publisher's certificate from Step 4 above to Nano Server with the certStoreLocation at Cert:\LocalMachine\TrustedPeople. For example:
   `Import-Certificate -FilePath .\xyz.cer -CertStoreLocation Cert:\LocalMachine\TrustedPeople`
2. Install the app on Nano Server by running the **Add-AppxPackage** PowerShell cmdlet to install a WSA package on Nano Server. For example: `Add-AppxPackage wsaSample.appx`

Additional resources for creating apps

WSA is server extension of Windows app package technology (though it is not hosted in Microsoft Store). If you want to publish apps with WSA,these topics will help you familiarize yourself with the app package pipeline:

- How to create a basic package manifest
- App Packager (MakeAppx.exe)
- How to create an app package signing certificate
- SignTool

Installing drivers on Nano Server

You can install non-Microsoft drivers on Nano Server by using INF driver packages. These include both Plug-and-Play (PnP) driver packages and File System Filter driver packages. Network Filter drivers are not currently supported on Nano Server.

Both PnP and File System Filter driver packages must follow the Universal driver requirements and installation process, as well as general driver package guidelines such as signing. They are documented at these locations:

- Driver Signing
- Using a Universal INF File

Installing driver packages offline

Supported driver packages can be installed on Nano Server offline via DISM.exe or DISM PowerShell cmdlets.

Installing driver packages online

PnP driver packages can be installed to Nano Server online by using PnpUtil. Online driver installation for non-PnP driver packages is not currently supported on Nano Server.

Joining Nano Server to a domain

To add Nano Server to a domain online

1. Harvest a data blob from a computer in the domain that is already running Windows Threshold Server using this command:

   ```
   djoin.exe /provision /domain <domain-name> /machine <machine-name> /savefile .\odjblob
   ```

   This saves the data blob in a file called odjblob.

2. Copy the odjblob file to the Nano Server computer with these commands:

   net use z: \\<ip address of Nano Server>\c$

NOTE

If the net use command fails, you probably need to adjust Windows Firewall rules. To do this, first open an elevated command prompt, start Windows PowerShell and then connect to the Nano Server computer with Windows PowerShell Remoting with these commands:

```
Set-Item WSMan:\localhost\Client\TrustedHosts <IP address of Nano Server>
```

```
$ip = <ip address of Nano Server>
```

```
Enter-PSSession -ComputerName $ip -Credential $ip\Administrator
```

When prompted, provide the Administrator password, then run this command to set the firewall rule:

netsh advfirewall firewall set rule group=File and Printer Sharing new enable=yes

Exit Windows PowerShell with `Exit-PSSession`, and then retry the net use command. If successful, continue copying the odjblob file contents to the Nano Server.

md z:\Temp

copy odjblob z:\Temp

3. Check the domain you want to join Nano Server to and ensure that DNS is configured. Also, verify that name resolution of the domain or a domain controller works as expected. To do this, open an elevated command prompt, start Windows PowerShell and then connect to the Nano Server computer with Windows PowerShell remoting with these commands:

   ```
   Set-Item WSMan:\localhost\Client\TrustedHosts <IP address of Nano Server>
   ```

   ```
   $ip = <ip address of Nano Server>
   ```

   ```
   Enter-PSSession -ComputerName $ip -Credential $ip\Administrator
   ```

   When prompted, provide the Administrator password. Nslookup is not available on Nano Server, so you can verify name resolution with Resolve-DNSName.

4. If name resolution succeeds, then in the same Windows PowerShell session, run this command to join the domain:

   ```
   djoin /requestodj /loadfile c:\Temp\odjblob /windowspath c:\windows /localos
   ```

5. Restart the Nano Server computer, and then exit the Windows PowerShell session:

   ```
   shutdown /r /t 5
   ```

   ```
   Exit-PSSession
   ```

6. After you have joined Nano Server to a domain, add the domain user account to the Administrators group on the Nano Server.

7. For security, remove the Nano Server from the trusted hosts list with this command:
   `Set-Item WSMan:\localhost\client\TrustedHosts`

Alternate method to join a domain in one step

First, harvest the data blob from another computer running Windows Threshold Server that is already in your domain using this command:

```
djoin.exe /provision /domain <domain-name> /machine <machine-name> /savefile .\odjblob
```

Open the file odjblob (perhaps in Notepad), copy its contents, and then paste the contents into the

<AccountData> section of the Unattend.xml file below.

Put this Unattend.xml file into the C:\NanoServer folder, and then use the following commands to mount the VHD and apply the settings in the `offlineServicing` section:

```
dism\dism /Mount-ImagemediaFile:.\NanoServer.vhd /Index:1 /MountDir:.\mountdir
dism\dismmedia:.\mountdir /Apply-Unattend:.\unattend.xml
```

Create a Panther folder (used by Windows systems for storing files during setup; see Windows 7, Windows Server 2008 R2, and Windows Vista setup log file locations if you're curious), copy the Unattend.xml file to it, and then unmount the VHD with these commands:

```
md .\mountdir\windows\panther
copy .\unattend.xml .\mountdir\windows\panther
dism\dism /Unmount-Image /MountDir:.\mountdir /Commit
```

The first time you boot Nano Server from this VHD, the other settings will be applied.

After you have joined Nano Server to a domain, add the domain user account to the Administrators group on the Nano Server.

Working with server roles on Nano Server

Using Hyper-V on Nano Server

Hyper-V works the same on Nano Server as it does on Windows Server in Server Core mode, with two exceptions:

- You must perform all management remotely and the management computer must be running the same build of Windows Server as the Nano Server. Older versions of Hyper-V Manager or Hyper-V Windows PowerShell cmdlets will not work.
- RemoteFX is not available.

In this release, these features of Hyper-V have been verified:

- Enabling Hyper-V
- Creation of Generation 1 and Generation 2 virtual machines
- Creation of virtual switches
- Starting virtual machines and running Windows guest operating systems
- Hyper-V Replica

If you want to perform a live migration of virtual machines, create a virtual machine on an SMB share, or connect resources on an existing SMB share to an existing virtual machine, it is vital that you configure authentication correctly. You have two options for doing this:

Constrained delegation

Constrained delegation works exactly the same as in previous releases. Refer to these articles for more information:

- Enabling Hyper-V Remote Management - Configuring Constrained Delegation For SMB and Highly Available SMB
- Enabling Hyper-V Remote Management - Configuring Constrained Delegation For Non-Clustered Live

Migration

CredSSP

First, refer to the Using Windows PowerShell remoting section of this topic to enable and test CredSSP. Then, on the management computer, you can use Hyper-V Manager and select the option to connect as another user. Hyper-V Manager will use CredSSP. You should do this even if you are using your current account.

Windows PowerShell cmdlets for Hyper-V can use CimSession or Credential parameters, either of which work with CredSSP.

Using Failover Clustering on Nano Server

Failover clustering works the same on Nano Server as it does on Windows Server in Server Core mode, but keep these caveats in mind:

- Clusters must be managed remotely with Failover Cluster Manager or Windows PowerShell.
- All Nano Server cluster nodes must be joined to the same domain, similar to cluster nodes in Windows Server.
- The domain account must have Administrator privileges on all Nano Server nodes, as with cluster nodes in Windows Server.
- All commands must be run in an elevated command prompt.

> **NOTE**
> Additionally, certain features are not supported in this release:
>
> - You cannot run failover clustering cmdlets on a local Nano Server through Windows PowerShell.
> - Clustering roles other than Hyper-V and File Server.

You'll find these Windows PowerShell cmdlets useful in managing Failover clusters:

You can create a new cluster with `New-Cluster -Name <clustername> -Node <comma-separated cluster node list>`

Once you've established a new cluster, you should run `Set-StorageSetting -NewDiskPolicy OfflineShared` on all nodes.

Add an additional node to the cluster with
`Add-ClusterNode -Name <comma-separated cluster node list> -Cluster <clustername>`

Remove a node from the cluster with
`Remove-ClusterNode -Name <comma-separated cluster node list> -Cluster <clustername>`

Create a Scale-Out File Server with `Add-ClusterScaleoutFileServerRole -name <sofsname> -cluster <clustername>`

You can find additional cmdlets for failover clustering at Microsoft.FailoverClusters.PowerShell.

Using DNS Server on Nano Server

To provide Nano Server with the DNS Server role, add the Microsoft-NanoServer-DNS-Package to the image (see the Creating a custom Nano Server image section of this topic. Once the Nano Server is running, connect to it and run this command from and elevated Windows PowerShell console to enable the feature:

```
Enable-WindowsOptionalFeature -Online -FeatureName DNS-Server-Full-Role
```

Using IIS on Nano Server

For steps to use the Internet Information Services (IIS) role, see IIS on Nano Server.

Using MPIO on Nano Server

For steps to use MPIO, see MPIO on Nano Server

Using SSH on Nano Server

For instructions on how to install and use SSH on Nano Server with the OpenSSH project, see the Win32-OpenSSH wiki.

Appendix: Sample Unattend.xml file that joins Nano Server to a domain

> **NOTE**
>
> Be sure to delete the trailing space in the contents of odjblob once you paste it into the Unattend file.

```
<?xml version='1.0' encoding='utf-8'?>
<unattend xmlns=urn:schemas-microsoft-com:unattend
xmlns:wcm=https://schemas.microsoft.com/WMIConfig/2002/State xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance>

  <settings pass=offlineServicing>
    <component name=Microsoft-Windows-UnattendedJoin processorArchitecture=amd64
publicKeyToken=31bf3856ad364e35 language=neutral versionScope=nonSxS>
        <OfflineIdentification>
           <Provisioning>
             <AccountData>
AAAAAAARUABLEABLEABAoAAAAAAAMABSUABLEABLEABAwAAAAAAAAABbMAAdYABc8ABYkABLAABbMAAEAAAAMAAA0ABY4ABZ8ABbIABa0AAc
IABY4ABb8ABZUABAsAAAAAAQAAZoABNUABOYABZYAANQABMoAAOEAAMIAAOkAANoAAMAAAXwAAJAAAAYAAA0ABY4ABZ8ABbIABa0AAcIABY
4ABb8ABZUABLEAALMABLQABU0AATMABXAAAAAAAKdf/mhfXoAAUAAAQAAAAb8ABLQABbMABcMABb4ABc8ABAIAAAAAAb8ABLQABbMABcMABb
4ABc8ABLQABb0ABZIAAGAAAAsAAR4ABTQABUAAAAAAACAAAQwABZMAAZcAAUgABVcAAegAARcABKkABVIAASwAAY4ABbcABW8ABQoAAT0ABN
8AAO8ABekAAJMAAVkAAZUABckABXEABJUAAQ8AAJ4AAIsABZMABdoAAOsABIsABKkABQEABUEABIwABKoAAaABBXgABNwAAegAAAkAAAAABA
MABLIABdIABc8ABY4AADAAAA4AAZ4ABbQABcAAAAAAACAAkKBW0ID8nJDWYAHnBAXE77j7BAEWEkl+lKB98XC2G0/9+Wd1DJQW4IYAkKBAAD
hAnKBWEwhiDAAAM2zzDCEAM6IAAAgAAAAAAQAAAAAAAAAAABwzzAAA
              </AccountData>
           </Provisioning>
         </OfflineIdentification>
    </component>
  </settings>

  <settings pass=oobeSystem>
    <component name=Microsoft-Windows-Shell-Setup processorArchitecture=amd64
publicKeyToken=31bf3856ad364e35 language=neutral versionScope=nonSxS>
      <UserAccounts>
        <AdministratorPassword>
           <Value>Tuva</Value>
           <PlainText>true</PlainText>
        </AdministratorPassword>
      </UserAccounts>
      <TimeZone>Pacific Standard Time</TimeZone>
    </component>
  </settings>

  <settings pass=specialize>
    <component name=Microsoft-Windows-Shell-Setup processorArchitecture=amd64
publicKeyToken=31bf3856ad364e35 language=neutral versionScope=nonSxS>
      <RegisteredOwner>My Team</RegisteredOwner>
      <RegisteredOrganization>My Corporation</RegisteredOrganization>
    </component>
  </settings>
</unattend>
```

IIS on Nano Server

10/25/2021 • 10 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

You can install the Internet Information Services (IIS) server role on Nano Server by using the -Package parameter with Microsoft-NanoServer-IIS-Package. For information about configuring Nano Server, including installing packages, see Install Nano Server.

In this release of Nano Server, the following IIS features are available:

FEATURE	ENABLED BY DEFAULT
Common HTTP Features	
Default document	x
Directory browsing	x
HTTP Errors	x
Static content	x
HTTP redirection	
Health and Diagnostics	
HTTP logging	x
Custom logging	
Request monitor	
Tracing	
Performance	
Static content compression	x
Dynamic content compression	
Security	

FEATURE	ENABLED BY DEFAULT
Request filtering	x
Basic authentication	
Client certificate mapping authentication	
Digest authentication	
IIS client certificate mapping authentication	
IP and domain restrictions	
URL authorization	
Windows authentication	
Application Development	
Application initialization	
CGI	
ISAPI extensions	
ISAPI filters	
Server-side includes	
WebSocket protocol	
Management Tools	
IISAdministration module for Windows PowerShell	x

A series of articles on other configurations of IIS (such as using ASP.NET, PHP, and Java) and other related content is published at https://iis.net/learn.

Installing IIS on Nano Server

You can install this server role either offline (with the Nano Server off) or online (with the Nano Server running); offline installation is the recommended option.

For offline installation, add the package with the -Packages parameter of New-NanoServerImage, as in this example:

```
New-NanoServerImage -Edition Standard -DeploymentType Guest -MediaPath f:\ -BasePath .\Base -TargetPath
.\Nano1.vhd -ComputerName Nano1 -Package Microsoft-NanoServer-IIS-Package
```

If you have an existing VHD file, you can install IIS offline with DISM.exe by mounting the VHD, and then using the **Add-Package** option. The following example steps assume that you are running from the directory specified by BasePath option, which was created after running New-NanoServerImage.

1. mkdir mountdir
2. .\Tools\dism.exe /Mount-Image /ImageFile:.\NanoServer.vhd /Index:1 /MountDir:.\mountdir
3. .\Tools\dism.exe /Add-Package /PackagePath:.\packages\Microsoft-NanoServer-IIS-Package.cab /Image:.\mountdir
4. .\Tools\dism.exe /Add-Package /PackagePath:.\packages\en-us\Microsoft-NanoServer-IIS-Package_en-us.cab /Image:.\mountdir
5. .\Tools\dism.exe /Unmount-Image /MountDir:.\MountDir /Commit

> **NOTE**
> Note that Step 4 adds the language pack--this example installs EN-US.

At this point, you can start Nano Server with IIS.

Installing IIS on Nano Server online

Though offline installation of the server role is recommended, you might need to install it online (with the Nano Server running) in container scenarios. To do this, follow these steps:

1. Copy the Packages folder from the installation media locally to the running Nano Server (for example, to C:\packages).

2. Create a new Unattend.xml file on another computer and then copy it to the Nano Server. You can copy and paste this XML content into the XML file you created:

```
    <unattend xmlns=urn:schemas-microsoft-com:unattend>
    <servicing>
        <package action=install>
            <assemblyIdentity name=Microsoft-NanoServer-IIS-Package version=10.0.14393.0
processorArchitecture=amd64 publicKeyToken=31bf3856ad364e35 language=neutral />
            <source location=c:\packages\Microsoft-NanoServer-IIS-Package.cab />
        </package>
        <package action=install>
            <assemblyIdentity name=Microsoft-NanoServer-IIS-Package version=10.0.14393.0
processorArchitecture=amd64 publicKeyToken=31bf3856ad364e35 language=en-US />
            <source location=c:\packages\en-us\Microsoft-NanoServer-IIS-Package_en-us.cab />
        </package>
    </servicing>
    <cpi:offlineImage cpi:source= xmlns:cpi=urn:schemas-microsoft-com:cpi />
</unattend>
```

3. In the new XML file you created (or copied), edit C:\packages to the directory you copied the content of Packages to.

4. Switch to the directory with the newly created XML file and run

   dism /online /apply-unattend:.\unattend.xml

5. Confirm that the IIS package and its associated language pack are installed correctly by running:

   dism /online /get-packages

   You should see Package Identity : Microsoft-NanoServer-IIS-Package~31bf3856ad364e35~amd64~~10.0.14393.1000 listed twice, once for Release Type : Language Pack and once for Release Type : Feature Pack.

6. Start the W3SVC service either with **net start w3svc** or by restarting the Nano Server.

Starting IIS

Once IIS is installed and running, it is ready to serve web requests. Verify that IIS is running by browsing the default IIS web page at http://<IP address of Nano Server>. On a physical computer, you can determine the IP address by using the Recovery Console. On a virtual machine, you can get the IP address by using a Windows PowerShell prompt and running:

```
Get-VM -name <VM name> | Select -ExpandProperty networkadapters | select IPAddresses
```

If you are not able to access the default IIS web page, double-check the IIS installation by looking for the **c:\inetpub** directory on the Nano Server.

Enabling and disabling IIS features

A number of IIS features are enabled by default when you install the IIS role (see the table in the Overview of IIS on Nano Server section of this topic). You can enable (or disable) additional features using DISM.exe

Each feature of IIS exists as a set of configuration elements. For example, the Windows authentication feature comprises these elements:

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=WindowsAuthenticationModule image=%windir%\System32\inetsrv\authsspi.dll`
`<modules>`	`<add name=WindowsAuthenticationModule lockItem=true \/>`
`<windowsAuthentication>`	`<windowsAuthentication enabled=false authPersistNonNTLM\=true><providers><add value=Negotiate /><add value=NTLM />  </providers> </windowsAuthentication>`

The full set of IIS sub-features is included in Appendix 1 of this topic and their corresponding configuration elements is included in Appendix 2 of this topic.

Example: installing Windows authentication

1. Open a Windows PowerShell remote session console on the Nano Server.
2. Use `DISM.exe` to install the Windows authentication module:

   ```
   dism /Enable-Feature /online /featurename:IIS-WindowsAuthentication /all
   ```

   The `/all` switch will install any feature that the chosen feature depends on.

Example: uninstalling Windows authentication

1. Open a Windows PowerShell remote session console on the Nano Server.
2. Use `DISM.exe` to uninstall the Windows authentication module:

   ```
   dism /Disable-Feature /online /featurename:IIS-WindowsAuthentication
   ```

Other common IIS configuration tasks

Creating websites

Use this cmdlet:

```
PS D:\> New-IISSite -Name TestSite -BindingInformation *:80:TestSite -PhysicalPath c:\test
```

You can then run `Get-IISSite` to verify the state of the site (returns the web site name, ID, state, physical path, and bindings).

Deleting web sites

Run `Remove-IISSite -Name TestSite -Confirm:$false`.

Creating virtual directories

You can create virtual directories by using the IISServerManager object returned by Get-IISServerManager, which exposes the .NET Microsoft.Web.Administration.ServerManager API. In this example, these commands access the Default Web Site element of the Sites collection and the root application element (/) of the Applications section. They then call the Add() method of the VirtualDirectories collection for that application element to create the new directory:

```
PS C:\> $sm = Get-IISServerManager
PS C:\> $sm.Sites["Default Web Site"].Applications[/].VirtualDirectories.Add(/DemoVirtualDir1,
c:\test\virtualDirectory1)
PS C:\> $sm.Sites["Default Web Site"].Applications[/].VirtualDirectories.Add(/DemoVirtualDir2,
c:\test\virtualDirectory2)
PS C:\> $sm.CommitChanges()
```

Creating application pools

Similarly you can use Get-IISServerManager to create application pools:

```
PS C:\> $sm = Get-IISServerManager
PS C:\> $sm.ApplicationPools.Add(DemoAppPool)
```

Configuring HTTPS and certificates

Use the Certoc.exe utility to import certificates, as in this example, which shows configuring HTTPS for a website on a Nano Server:

1. On another computer that is not running Nano Server, create a certificate (using your own certificate name and password), and then export it to c:\temp\test.pfx.

   `$newCert = New-SelfSignedCertificate -DnsName www.foo.bar.com -CertStoreLocation cert:\LocalMachine\my`

   `$mypwd = ConvertTo-SecureString -String YOUR_PFX_PASSWD -Force -AsPlainText`

   `Export-PfxCertificate -FilePath c:\temp\test.pfx -Cert $newCert -Password $mypwd`

2. Copy the test.pfx file to the Nano Server computer.

3. On the Nano Server, import the certificate to the My store with this command:

   certoc.exe -ImportPFX -p YOUR_PFX_PASSWD My c:\temp\test.pfx

4. Retrieve the thumbprint of this new certificate (in this example, 61E71251294B2A7BB8259C2AC5CF7BA622777E73) with `Get-ChildItem Cert:\LocalMachine\my`.

5. Add the HTTPS binding to the Default Web Site (or whatever website you want to add the binding to) by using these Windows PowerShell commands:

```
$certificate = get-item Cert:\LocalMachine\my\61E71251294B2A7BB8259C2AC5CF7BA622777E73
# Use your actual thumbprint instead of this example
$hash = $certificate.GetCertHash()

Import-Module IISAdministration
$sm = Get-IISServerManager
$sm.Sites["Default Web Site"].Bindings.Add("*:443:", $hash, "My", "0")    # My is the certificate
store name
$sm.CommitChanges()
```

You could also use Server Name Indication (SNI) with a specific host name with this syntax:

```
$sm.Sites["Default Web Site"].Bindings.Add("*:443:www.foo.bar.com", $hash, "My", "SNI")
```

Appendix 1: List of IIS sub-features

- IIS-WebServer
- IIS-CommonHttpFeatures
- IIS-StaticContent
- IIS-DefaultDocument
- IIS-DirectoryBrowsing
- IIS-HttpErrors
- IIS-HttpRedirect
- IIS-ApplicationDevelopment
- IIS-CGI
- IIS-ISAPIExtensions
- IIS-ISAPIFilter
- IIS-ServerSideIncludes
- IIS-WebSockets
- IIS-ApplicationInit
- IIS-Security
- IIS-BasicAuthentication
- IIS-WindowsAuthentication
- IIS-DigestAuthentication
- IIS-ClientCertificateMappingAuthentication
- IIS-IISCertificateMappingAuthentication
- IIS-URLAuthorization
- IIS-RequestFiltering
- IIS-IPSecurity
- IIS-CertProvider
- IIS-Performance
- IIS-HttpCompressionStatic
- IIS-HttpCompressionDynamic
- IIS-HealthAndDiagnostics
- IIS-HttpLogging
- IIS-LoggingLibraries
- IIS-RequestMonitor
- IIS-HttpTracing
- IIS-CustomLogging

Appendix 2: Elements of HTTP features

Each feature of IIS exists as a set of configuration elements. This appendix lists the configuration elements for all of the features in this release of Nano Server

Common HTTP features

Default document

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=DefaultDocumentModule image=%windir%\System32\inetsrv\defdoc.dll />`
`<modules>`	`<add name=DefaultDocumentModule lockItem=true />`
`<handlers>`	`<add name=StaticFile path=* verb=* modules=DefaultDocumentModule resourceType=EiSecther requireAccess=Read />`
`<defaultDocument>`	`<defaultDocument enabled=true> <files>  <add value=Default.htm />  <add value=Default.asp />  <add value=index.htm />   <add value=index.html />  <add value=iisstart.htm />  </files>  </defaultDocument>`

The `StaticFile <handlers>` entry might already be present; if so, just add DefaultDocumentModule to the <modules> attribute, separated by a comma.

Directory browsing

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=DirectoryListingModule image=%windir%\System32\inetsrv\dirlist.dll />`
`<modules>`	`<add name=DirectoryListingModule lockItem=true />`
`<handlers>`	`<add name=StaticFile path=* verb=* modules=DirectoryListingModule resourceType=Either requireAccess=Read />`

The `StaticFile <handlers>` entry might already be present; if so, just add DirectoryListingModule to the <modules> attribute, separated by a comma.

HTTP errors

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=CustomErrorModule image=%windir%\System32\inetsrv\custerr.dll />`
`<modules>`	`<add name=CustomErrorModule lockItem=true />`

SECTION	CONFIGURATION ELEMENTS
`<httpErrors>`	`<httpErrors lockAttributes=allowAbsolutePathsWhenDelegated,defaultPath> <error statusCode=401 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=401.htm > <error statusCode=403 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=403.htm /> <error statusCode=404 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=404.htm /> <error statusCode=405 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=405.htm /> <error statusCode=406 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=406.htm /> <error statusCode=412 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=412.htm /> <error statusCode=500 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=500.htm /> <error statusCode=501 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=501.htm /> <error statusCode=502 prefixLanguageFilePath=%SystemDrive%\inetpub\custerr path=502.htm /> </httpErrors>`

Static content

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=StaticFileModule image=%windir%\System32\inetsrv\static.dll />`
`<modules>`	`<add name=StaticFileModule lockItem=true />`
`<handlers>`	`<add name=StaticFile path=* verb=* modules=StaticFileModule resourceType=Either requireAccess=Read />`

The `StaticFile \<handlers>` entry might already be present; if so, just add StaticFileModule to the <modules> attribute, separated by a comma.

HTTP redirection

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=HttpRedirectionModule image=%windir%\System32\inetsrv\redirect.dll />`
`<modules>`	`<add name=HttpRedirectionModule lockItem=true />`
`<httpRedirect>`	`<httpRedirect enabled=false />`

Health and diagnostics

HTTP logging

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=HttpLoggingModule image=%windir%\System32\inetsrv\loghttp.dll />`
`<modules>`	`<add name=HttpLoggingModule lockItem=true />`

SECTION	CONFIGURATION ELEMENTS
`<httpLogging>`	`<httpLogging dontLog=false />`

Custom logging

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=CustomLoggingModule image=%windir%\System32\inetsrv\logcust.dll />`
`<modules>`	`<add name=CustomLoggingModule lockItem=true />`

Request monitor

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=RequestMonitorModule image=%windir%\System32\inetsrv\iisreqs.dll />`

Tracing

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=TracingModule image=%windir%\System32\inetsrv\iisetw.dll \/> <add name=FailedRequestsTracingModule image=%windir%\System32\inetsrv\iisfreb.dll />`
`<modules>`	`<add name=FailedRequestsTracingModule lockItem=true />`
`<traceProviderDefinitions>`	`<traceProviderDefinitions> <add name=WWW Server guid\={3a2a4e84-4c21-4981-ae10-3fda0d9b0f83}> <areas> <clear /> <add name=Authentication value=2 /> <add name=Security value=4 /> <add name=Filter value=8 /> <add name=StaticFile value=16 /> <add name=CGI value=32 /> <add name=Compression value=64 /> <add name=Cache value=128 /> <add name=RequestNotifications value=256 /> <add name=Module value=512 /> <add name=FastCGI value=4096 /> <add name=WebSocket value=16384 /> </areas> </add> <add name=ISAPI Extension guid={a1c2040e-8840-4c31-ba11-9871031a19ea}> <areas> <clear /> </areas> </add> </traceProviderDefinitions>`

Performance

Static content compression

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=StaticCompressionModule image=%windir%\System32\inetsrv\compstat.dll />`
`<modules>`	`<add name=StaticCompressionModule lockItem=true />`

SECTION	CONFIGURATION ELEMENTS
`<httpCompression>`	`<httpCompression directory=%SystemDrive%\inetpub\temp\IIS Temporary Compressed Files> <scheme name=gzip dll=%Windir%\system32\inetsrv\gzip.dll /> <staticTypes> <add mimeType=text/* enabled=true /> <add mimeType=message/* enabled=true /> <add mimeType=application/javascript enabled=true \/> <add mimeType=application/atom+xml enabled=true /> <add mimeType=application/xaml+xml enabled=true /> <add mimeType=** enabled=false /> </staticTypes> </httpCompression>`

Dynamic content compression

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=DynamicCompressionModule image=%windir%\System32\inetsrv\compdyn.dll />`
`<modules>`	`<add name=DynamicCompressionModule lockItem=true />`
`<httpCompression>`	`<httpCompression directory\=%SystemDrive%\inetpub\temp\IIS Temporary Compressed Files> <scheme name=gzip dll=%Windir%\system32\inetsrv\gzip.dll \/> \ <dynamicTypes> <add mimeType=text/* enabled=true \/> <add mimeType=message/* enabled=true /> <add mimeType=application/x-javascript enabled=true /> <add mimeType=application/javascript enabled=true /> <add mimeType=*/* enabled=false /> <\/dynamicTypes> </httpCompression>`

Security

Request filtering

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=RequestFilteringModule image=%windir%\System32\inetsrv\modrqflt.dll />`
`<modules>`	`<add name=RequestFilteringModule lockItem=true />`
`<requestFiltering>`	`<requestFiltering>  <fileExtensions allowUnlisted=true applyToWebDAV=true />  <verbs allowUnlisted=true applyToWebDAV=true />  <hiddenSegments applyToWebDAV=true>  <add segment=web.config />  </hiddenSegments>  </requestFiltering>`

Basic authentication

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=BasicAuthenticationModule image=%windir%\System32\inetsrv\authbas.dll />`
`<modules>`	`<add name=WindowsAuthenticationModule lockItem=true />`

SECTION	CONFIGURATION ELEMENTS
`<basicAuthentication>`	`<basicAuthentication enabled=false />`

Client certificate mapping authentication

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=CertificateMappingAuthentication image=%windir%\System32\inetsrv\authcert.dll />`
`<modules>`	`<add name=CertificateMappingAuthenticationModule lockItem=true />`
`<clientCertificateMappingAuthentication>`	`<clientCertificateMappingAuthentication enabled=false />`

Digest authentication

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=DigestAuthenticationModule image=%windir%\System32\inetsrv\authmd5.dll />`
`<modules>`	`<add name=DigestAuthenticationModule lockItem=true />`
`<other>`	`<digestAuthentication enabled=false />`

IIS client certificate mapping authentication

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=CertificateMappingAuthenticationModule image=%windir%\System32\inetsrv\authcert.dll />`
`<modules>`	`<add name=CertificateMappingAuthenticationModule lockItem=true` />`
`<clientCertificateMappingAuthentication>`	`<clientCertificateMappingAuthentication enabled=false />`

IP and domain restrictions

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=IpRestrictionModule image=%windir%\System32\inetsrv\iprestr.dll /> <add name=DynamicIpRestrictionModule image=%windir%\System32\inetsrv\diprestr.dll />`
`<modules>`	`<add name=IpRestrictionModule lockItem=true \/> <add name=DynamicIpRestrictionModule lockItem=true \/>`
`<ipSecurity>`	`<ipSecurity allowUnlisted=true />`

URL authorization

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=UrlAuthorizationModule image=%windir%\System32\inetsrv\urlauthz.dll />`
`<modules>`	`<add name=UrlAuthorizationModule lockItem=true />`
`<authorization>`	`<authorization>  <add accessType=Allow users=* /> </authorization>`

Windows authentication

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=WindowsAuthenticationModule image=%windir%\System32\inetsrv\authsspi.dll />`
`<modules>`	`<add name=WindowsAuthenticationModule lockItem=true />`
`<windowsAuthentication>`	`<windowsAuthentication enabled=false authPersistNonNTLM\=true> <providers> <add value=Negotiate /> <add value=NTLM /> <\providers> <\windowsAuthentication> <windowsAuthentication enabled=false authPersistNonNTLM\=true> <providers> <add value=Negotiate /> <add value=NTLM /> <\/providers> <\/windowsAuthentication>`

Application development

Application initialization

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=ApplicationInitializationModule image=%windir%\System32\inetsrv\warmup.dll />`
`<modules>`	`<add name=ApplicationInitializationModule lockItem=true />`

CGI

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=CgiModule image=%windir%\System32\inetsrv\cgi.dll />  <add name=FastCgiModule image=%windir%\System32\inetsrv\iisfcgi.dll />`
`<modules>`	`<add name=CgiModule lockItem=true /> <add name=FastCgiModule lockItem=true />`
`<handlers>`	`<add name=CGI-exe path=*.exe verb=\* modules=CgiModule resourceType=File requireAccess=Execute allowPathInfo=true />`

ISAPI extensions

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=IsapiModule image=%windir%\System32\inetsrv\isapi.dll />`
`<modules>`	`<add name=IsapiModule lockItem=true />`
`<handlers>`	`<add name=ISAPI-dll path=*.dll verb=* modules=IsapiModule resourceType=File requireAccess=Execute allowPathInfo=true />`

ISAPI filters

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=IsapiFilterModule image=%windir%\System32\inetsrv\filter.dll />`
`<modules>`	`<add name=IsapiFilterModule lockItem=true />`

Server-side includes

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	< `add name=ServerSideIncludeModule image=%windir%\System32\inetsrv\iis_ssi.dll />`
`<modules>`	`<add name=ServerSideIncludeModule lockItem=true />`
`<handlers>`	`<add name=SSINC-stm path=*.stm verb=GET,HEAD,POST modules=ServerSideIncludeModule resourceType=File \/> <add name=SSINC-shtm path=*.shtm verb=GET,HEAD,POST modules=ServerSideIncludeModule resourceType=File /> <add name=SSINC-shtml path=*.shtml verb=GET,HEAD,POST modules=ServerSideIncludeModule resourceType=File />`
`<serverSideInclude>`	`<serverSideInclude ssiExecDisable=false />`

WebSocket protocol

SECTION	CONFIGURATION ELEMENTS
`<globalModules>`	`<add name=WebSocketModule image=%windir%\System32\inetsrv\iiswsock.dll />`
`<modules>`	`<add name=WebSocketModule lockItem=true />`

MPIO on Nano Server

10/25/2021 • 5 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

This topic introduces the use of MPIO in Nano Server installations of Windows Server 2016. For general information about MPIO in Windows Server, see Multipath I/O Overview.

Using MPIO on Nano Server

You can use MPIO on Nano Server, but with these differences:

- Only MSDSM is supported.
- The Load Balancing Policy is chosen dynamically and cannot be modified. The policy has these characteristics:
  - Default -- RoundRobin (active/active)
  - SAS HDD -- LeastBlocks
  - ALUA -- RoundRobin with Subset
- Path states (active/passive) for ALUA arrays are picked up from the target array.
- Storage devices are claimed by bus type (for example, FC, iSCSI, or SAS). When MPIO is installed on Nano Server, disks are still exposed as duplicates (one available per path) until MPIO is configured to claim and manage particular disks. The sample script in this topic will claim or unclaim disks for MPIO.
- iSCSI boot is not supported.

Enable MPIO with this Windows PowerShell cmdlet:

```
Enable-WindowsOptionalFeature -Online -FeatureName MultiPathIO
```

This sample script will allow the caller to claim or unclaim disks for MPIO by changing certain registry keys. Though you can claim other storage devices by adding them to these keys, manipulating the keys directly is not recommended.

```
#
#  Copyright (c) 2015 Microsoft Corporation.  All rights reserved.
#
#  THIS CODE AND INFORMATION IS PROVIDED AS IS WITHOUT WARRANTY
#  OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED
#  TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
#  PARTICULAR PURPOSE
#

<#
.Synopsis
```

```
    This powershell script allows you to enable Multipath-IO support using Microsoft's
    in-box DSM (MSDSM) for storage devices attached by certain bus types.

    After running this script you will have to either:
    1. Disable and then re-enable the relevant Host Bus Adapters (HBAs); or
    2. Reboot the system.

.Description

.Parameter BusType
    Specifies the bus type for which the claim/unclaim should be done.

    If omitted, this parameter defaults to All.

    All - Will claim/unclaim storage devices attached through Fibre Channel, iSCSI, or SAS.

    FC - Will claim/unclaim storage devices attached through Fibre Channel.

    iSCSI - Will claim/unclaim storage devices attached through iSCSI.

    SAS - Will claim/unclaim storage devices attached through SAS.

.Parameter Server
    Allows you to specify a remote system, either via computer name or IP address.

    If omitted, this parameter defaults to the local system.

.Parameter Unclaim
    If specified, the script will unclaim storage devices of the bus type specified by the
    BusType parameter.

    If omitted, the script will default to claiming storage devices instead.

.Example
MultipathIoClaim.ps1

Claims all storage devices attached through Fibre Channel, iSCSI, or SAS.

.Example
MultipathIoClaim.ps1 FC

Claims all storage devices attached through Fibre Channel.

.Example
MultipathIoClaim.ps1 SAS -Unclaim

Unclaims all storage devices attached through SAS.

.Example
MultipathIoClaim.ps1 iSCSI 12.34.56.78

Claims all storage devices attached through iSCSI on the remote system with IP address 12.34.56.78.

#>
[CmdletBinding()]
param
(
    [ValidateSet('all','fc','iscsi','sas')]
    [string]$BusType='all',

    [string]$Server=127.0.0.1,

    [switch]$Unclaim
)

#
# Constants
#
$type = [Microsoft.Win32.RegistryHive]::LocalMachine
```

```
[string]$mpioKeyName = SYSTEM\CurrentControlSet\Control\MPDEV
[string]$mpioValueName = MpioSupportedDeviceList
[string]$msdsmKeyName = SYSTEM\CurrentControlSet\Services\msdsm\Parameters
[string]$msdsmValueName = DsmSupportedDeviceList

[string]$fcHwid = MSFT2015FCBusType_0x6
[string]$sasHwid = MSFT2011SASBusType_0xA
[string]$iscsiHwid = MSFT2005iSCSIBusType_0x9

#
# Functions
#

function AddHardwareId
{
    param
    (
        [Parameter(Mandatory=$True)]
        [string]$Hwid,

        [string]$Srv=127.0.0.1,

        [string]$KeyName=SYSTEM\CurrentControlSet\Control\MultipathIoClaimTest,

        [string]$ValueName=DeviceList
    )

    $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $Srv)
    $key = $regKey.OpenSubKey($KeyName, 'true')
    $val = $key.GetValue($ValueName)
    $val += $Hwid
    $key.SetValue($ValueName, [string[]]$val, 'MultiString')
}

function RemoveHardwareId
{
    param
    (
        [Parameter(Mandatory=$True)]
        [string]$Hwid,

        [string]$Srv=127.0.0.1,

        [string]$KeyName=SYSTEM\CurrentControlSet\Control\MultipathIoClaimTest,

        [string]$ValueName=DeviceList
    )

    [string[]]$newValues = @()
    $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $Srv)
    $key = $regKey.OpenSubKey($KeyName, 'true')
    $values = $key.GetValue($ValueName)
    foreach($val in $values)
    {
        # Only copy values that don't match the given hardware ID.
        if ($val -ne $Hwid)
        {
            $newValues += $val
            Write-Debug $($val) will remain in the key.
        }
        else
        {
            Write-Debug $($val) will be removed from the key.
        }
    }
    $key.SetValue($ValueName, [string[]]$newValues, 'MultiString')
}

function HardwareIdClaimed
```

```
{
    param
    (
        [Parameter(Mandatory=$True)]
        [string]$Hwid,

        [string]$Srv=127.0.0.1,

        [string]$KeyName=SYSTEM\CurrentControlSet\Control\MultipathIoClaimTest,

        [string]$ValueName=DeviceList
    )

    $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $Srv)
    $key = $regKey.OpenSubKey($KeyName)
    $values = $key.GetValue($ValueName)
    foreach($val in $values)
    {
        if ($val -eq $Hwid)
        {
            return 'true'
        }
    }

    return 'false'
}

function GetBusTypeName
{
    param
    (
        [Parameter(Mandatory=$True)]
        [string]$Hwid
    )

    if ($Hwid -eq $fcHwid)
    {
        return Fibre Channel
    }
    elseif ($Hwid -eq $sasHwid)
    {
        return SAS
    }
    elseif ($Hwid -eq $iscsiHwid)
    {
        return iSCSI
    }

    return Unknown
}

#
# Execution starts here.
#

#
# Create the list of hardware IDs to claim or unclaim.
#
[string[]]$hwids = @()

if ($BusType -eq 'fc')
{
    $hwids += $fcHwid
}
elseif ($BusType -eq 'iscsi')
{
    $hwids += $iscsiHwid
}
elseif ($BusType -eq 'sas')
```

```
{
    $hwids += $sasHwid
}
elseif ($BusType -eq 'all')
{
    $hwids += $fcHwid
    $hwids += $sasHwid
    $hwids += $iscsiHwid
}
else
{
    Write-Host Please provide a bus type (FC, iSCSI, SAS, or All).
}

$changed = 'false'

#
# Attempt to claim or unclaim each of the hardware IDs.
#
foreach($hwid in $hwids)
{
    $busTypeName = GetBusTypeName $hwid

    #
    # The device is only considered claimed if it's in both the MPIO and MSDSM lists.
    #
    $mpioClaimed = HardwareIdClaimed $hwid $Server $mpioKeyName $mpioValueName
    $msdsmClaimed = HardwareIdClaimed $hwid $Server $msdsmKeyName $msdsmValueName
    if ($mpioClaimed -eq 'true' -and $msdsmClaimed -eq 'true')
    {
        $claimed = 'true'
    }
    else
    {
        $claimed = 'false'
    }

    if ($mpioClaimed -eq 'true')
    {
        Write-Debug $($hwid) is in the MPIO list.
    }
    else
    {
        Write-Debug $($hwid) is NOT in the MPIO list.
    }

    if ($msdsmClaimed -eq 'true')
    {
        Write-Debug $($hwid) is in the MSDSM list.
    }
    else
    {
        Write-Debug $($hwid) is NOT in the MSDSM list.
    }

    if ($Unclaim)
    {
        #
        # Unclaim this hardware ID.
        #
        if ($claimed -eq 'true')
        {
            RemoveHardwareId $hwid $Server $mpioKeyName $mpioValueName
            RemoveHardwareId $hwid $Server $msdsmKeyName $msdsmValueName
            $changed = 'true'
            Write-Host $($busTypeName) devices will not be claimed.
        }
        else
        {
```

```
        {
            Write-Host $($busTypeName) devices are not currently claimed.
        }

    }
    else
    {
        #
        # Claim this hardware ID.
        #
        if ($claimed -eq 'true')
        {
            Write-Host $($busTypeName) devices are already claimed.
        }
        else
        {
            AddHardwareId $hwid $Server $mpioKeyName $mpioValueName
            AddHardwareId $hwid $Server $msdsmKeyName $msdsmValueName
            $changed = 'true'
            Write-Host $($busTypeName) devices will be claimed.
        }
    }
}

#
# Finally, if we changed any of the registry keys remind the user to restart.
#
if ($changed -eq 'true')
{
    Write-Host The system must be restarted for the changes to take effect.
}
```

Manage Nano Server

10/25/2021 • 12 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

Nano Server is managed remotely. There is no local logon capability at all, nor does it support Terminal Services. However, you have a wide variety of options for managing Nano Server remotely, including Windows PowerShell, Windows Management Instrumentation (WMI), Windows Remote Management, and Emergency Management Services (EMS).

To use any remote management tool, you will probably need to know the IP address of the Nano Server. Some ways to find out the IP address include:

- Use the Nano Recovery Console (see the Using the Nano Server Recovery Console section of this topic for details).
- Connect a serial cable to the computer and use EMS.
- Using the computer name you assigned to the Nano Server while configuring it, you can get the IP address with ping. For example, `ping NanoServer-PC /4`.

Using Windows PowerShell remoting

To manage Nano Server with Windows PowerShell remoting, you need to add the IP address of the Nano Server to your management computer's list of trusted hosts, add the account you are using to the Nano Server's administrators, and enable CredSSP if you plan to use that feature.

NOTE

If the target Nano Server and your management computer are in the same AD DS forest (or in forests with a trust relationship), you should not add the Nano Server to the trusted hosts list--you can connect to the Nano Server by using its fully qualified domain name, for example: PS C:> Enter-PSSession -ComputerName nanoserver.contoso.com -Credential (Get-Credential)

To add the Nano Server to the list of trusted hosts, run this command at an elevated Windows PowerShell prompt:

```
Set-Item WSMan:\localhost\Client\TrustedHosts <IP address of Nano Server>
```

To start the remote Windows PowerShell session, start an elevated local Windows PowerShell session, and then run these commands:

```
$ip = <IP address of Nano Server>
$user = $ip\Administrator
Enter-PSSession -ComputerName $ip -Credential $user
```

You can now run Windows PowerShell commands on the Nano Server as normal.

> **NOTE**
>
> Not all Windows PowerShell commands are available in this release of Nano Server. To see which are available, run `Get-Command -CommandType Cmdlet`

Stop the remote session with the command `Exit-PSSession`

Using Windows PowerShell CIM sessions over WinRM

You can use CIM sessions and instances in Windows PowerShell to run WMI commands over Windows Remote Management (WinRM).

Start the CIM session by running these commands in a Windows PowerShell prompt:

```
$ip = <IP address of the Nano Server\>
$user = $ip\Administrator
$cim = New-CimSession -Credential $user -ComputerName $ip
```

With the session established, you can run various WMI commands, for example:

```
Get-CimInstance -CimSession $cim -ClassName Win32_ComputerSystem | Format-List *
Get-CimInstance -CimSession $Cim -Query SELECT * from Win32_Process WHERE name LIKE 'p%'
```

Windows Remote Management

You can run programs remotely on the Nano Server with Windows Remote Management (WinRM). To use WinRM, first configure the service and set the code page with these commands at an elevated command prompt:

```
winrm quickconfig
winrm set winrm/config/client @{TrustedHosts=<ip address of Nano Server>}
chcp 65001
```

Now you can run commands remotely on the Nano Server. For example:

```
winrs -r:<IP address of Nano Server> -u:Administrator -p:<Nano Server administrator password> ipconfig
```

For more information about Windows Remote Management, see Windows Remote Management (WinRM) Overview.

Running a network trace on Nano Server

Netsh trace, Tracelog.exe, and Logman.exe are not available in Nano Server. To capture network packets, you can use these Windows PowerShell cmdlets:

```
New-NetEventSession [-Name]
Add-NetEventPacketCaptureProvider -SessionName
Start-NetEventSession [-Name]
Stop-NetEventSession [-Name]
```

These cmdlets are documented in detail at Network Event Packet Capture Cmdlets in Windows PowerShell

Installing servicing packages

If you want install a servicing packages, use the -ServicingPackagePath parameter (you can pass an array of paths to .cab files):

```
New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath .\Base
-TargetPath .\NanoServer.wim -ServicingPackagePath \\path\to\kb123456.cab
```

Often, a servicing package or hotfix is downloaded as a KB item which contains a .cab file. Follow these steps to extract the .cab file, which you can then install with the -ServicingPackagePath parameter:

1. Download the servicing package (from the associated Knowledge Base article or from Microsoft Update Catalog. Save it to a local directory or network share, for example: C:\ServicingPackages
2. Create a folder in which you will save the extracted servicing package. Example: c:\KB3157663_expanded
3. Open a Windows PowerShell console and use the `Expand` command specifying the path to the .msu file of the servicing package, including the `-f:*` parameter and the path where you want servicing package to be extracted to. For example:
   ```
   Expand C:\ServicingPackages\Windows10.0-KB3157663-x64.msu -f:* C:\KB3157663_expanded
   ```
   The expanded files should look similar to this: C:>dir C:\KB3157663_expanded Volume in drive C is OS Volume Serial Number is B05B-CC3D

   Directory of C:\KB3157663_expanded

   04/19/2016 01:17 PM <DIR> . 04/19/2016 01:17 PM <DIR> .. 04/17/2016 12:31 AM 517 Windows10.0-KB3157663-x64-pkgProperties.txt 04/17/2016 12:30 AM 93,886,347 Windows10.0-KB3157663-x64.cab 04/17/2016 12:31 AM 454 Windows10.0-KB3157663-x64.xml 04/17/2016 12:36 AM 185,818 WSUSSCAN.cab 4 File(s) 94,073,136 bytes 2 Dir(s) 328,559,427,584 bytes free
4. Run `New-NanoServerImage` with the -ServicingPackagePath parameter pointing to the .cab file in this directory, for example:
   ```
   New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath
   .\Base -TargetPath .\NanoServer.wim -ServicingPackagePath C:\KB3157663_expanded\Windows10.0-KB3157663-
   x64.cab
   ```

Managing updates in Nano Server

Currently you can use the Windows Update provider for Windows Management Instrumentation (WMI) to find the list of applicable updates, and then install all or a subset of them. If you use Windows Server Update Services (WSUS), you can also configure Nano Server to contact the WSUS server to obtain updates.

In all cases, first establish a remote Windows PowerShell session to the Nano Server computer. These examples use *$sess* for the session; if you are using something else, replace that element as needed.

View all available updates

Obtain the full list of applicable updates with these commands:

```
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession

$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ScanForUpdates -Arguments
@{SearchCriteria=IsInstalled=0;OnlineScan=$true}
```

Note: If no updates are available, this command will return the following error:

```
Invoke-CimMethod : A general error occurred that is not covered by a more specific error code.

At line:1 char:16

+ ... anResults = Invoke-CimMethod -InputObject $sess -MethodName ScanForUp ...

+                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    + CategoryInfo          : NotSpecified: (MSFT_WUOperatio...-5b842a3dd45d)

   :CimInstance) [Invoke-CimMethod], CimException

    + FullyQualifiedErrorId : MI RESULT 1,Microsoft.Management.Infrastructure.

   CimCmdlets.InvokeCimMethodCommand
```

Install all available updates

You can detect, download, and install **all** available updates at one time by using these commands:

```
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession

$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ApplyApplicableUpdates

Restart-Computer
```

Note: Windows Defender will prevent updates from installing. To work around this, uninstall Windows Defender, install the updates, and then reinstall Windows Defender. Alternately, you can download the updates on another computer, copy them to the Nano Server, and then apply them with DISM.exe.

Verify installation of updates

Use these commands to get a list of the updates currently installed:

```
$sess = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName MSFT_WUOperationsSession

$scanResults = Invoke-CimMethod -InputObject $sess -MethodName ScanForUpdates -Arguments
@{SearchCriteria=IsInstalled=1;OnlineScan=$true}
```

Note: These commands list what is installed, but do not specifically quote installed in the output. If you need output including that, such as for a report, you can run

```
Get-WindowsPackage -Online
```

Using WSUS

The commands listed above will query the Windows Update and Microsoft Update serviceon the Internet to find and download updates. If you use WSUS, you can set registry keys on the Nano Server to use your WSUS server instead.

See the Windows Update Agent Environment Options Registry Keys table in Configure Automatic Updates in a Non-Active-Directory Environment

You should set at least the **WUServer** and **WUStatusServer** registry keys, but depending on how you have implemented WSUS, other values might be needed. You can always confirm these settings by examining another Windows Server in the same environment.

Once these values are set for your WSUS, the commands in the section above will query that server for updates

and use it the download source.

Automatic Updates

Currently, the way to automate update installation is to convert the steps above into a local Windows PowerShell script and then create a scheduled task to run it and restart the system on your schedule.

Performance and event monitoring on Nano Server

Nano Server fully supports the Event Tracing for Windows (ETW) framework, but some familiar tools used to manage tracing and performance counters are not currently available on Nano Server. However, Nano Server has tools and cmdlets to accomplish most common performance analysis scenarios.

The high-level workflow remains the same as on any Window Server installation -- low-overhead tracing is performed on the target (Nano Server) computer, and the resulting trace files and/or logs are post-processed offline on a separate computer using tools such as Windows Performance Analyzer, Message Analyzer, or others.

> **NOTE**
> Refer to How to copy files to and from Nano Server for a refresher on how to transfer files using PowerShell remoting.

The following sections list the most common performance data collection activities along with a supported way to accomplish them on Nano Server.

Query available event providers

Windows Performance Recorder is tool to query available event providers as follows:

```
wpr.exe -providers
```

You can filter the output on the type of events that are of interest. For example:

```
PS C:\> wpr.exe -providers | select-string Storage

       595f33ea-d4af-4f4d-b4dd-9dacdd17fc6e                            : Microsoft-Windows-
StorageManagement-WSP-Host
       595f7f52-c90a-4026-a125-8eb5e083f15e                            : Microsoft-Windows-StorageSpaces-
Driver
       69c8ca7e-1adf-472b-ba4c-a0485986b9f6                            : Microsoft-Windows-StorageSpaces-
SpaceManager
       7e58e69a-e361-4f06-b880-ad2f4b64c944                            : Microsoft-Windows-
StorageManagement
       88c09888-118d-48fc-8863-e1c6d39ca4df                            : Microsoft-Windows-
StorageManagement-WSP-Spaces
```

Record traces from a single ETW provider

You can use new Event Tracing Management cmdlets for this. Here is an example workflow:

Create and start the trace, specifying a file name for storing the events.

```
PS C:\> New-EtwTraceSession -Name ExampleTrace -LocalFilePath c:\etrace.etl
```

Add a provider GUID to the trace. Use `wpr.exe -providers` for Provider Name to GUID translation.

```
PS C:\> wpr.exe -providers | select-string Kernel-Memory

       d1d93ef7-e1f2-4f45-9943-03d245fe6c00                              : Microsoft-Windows-Kernel-Memory

PS C:\> Add-EtwTraceProvider -Guid {d1d93ef7-e1f2-4f45-9943-03d245fe6c00} -SessionName ExampleTrace
```

Remove the trace -- this stops the trace session, flushing events to the associated log file.

```
PS C:\> Remove-EtwTraceSession -Name ExampleTrace

PS C:\> dir .\etrace.etl

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/14/2016  11:17 AM       16515072 etrace.etl
```

> **NOTE**
>
> This example shows adding a single trace provider to the session, but you can also use the `Add-EtwTraceProvider` cmdlet multiple times on a trace session with different provider GUIDs to enable tracing from multiple sources. Another alternative is to use `wpr.exe` profiles described below.

Record traces from multiple ETW providers

The `-profiles` option of Windows Performance Recorder enables tracing from multiple providers at the same time. There are a number of built-in profiles like CPU, Network, and DiskIO to choose from:

```
PS C:\Users\Administrator\Documents> wpr.exe -profiles

Microsoft Windows Performance Recorder Version 10.0.14393 (CoreSystem)
Copyright (c) 2015 Microsoft Corporation. All rights reserved.

        GeneralProfile              First level triage
        CPU                         CPU usage
        DiskIO                      Disk I/O activity
        FileIO                      File I/O activity
        Registry                    Registry I/O activity
        Network                     Networking I/O activity
        Heap                        Heap usage
        Pool                        Pool usage
        VirtualAllocation           VirtualAlloc usage
        Audio                       Audio glitches
        Video                       Video glitches
        Power                       Power usage
        InternetExplorer            Internet Explorer
        EdgeBrowser                 Edge Browser
        Minifilter                  Minifilter I/O activity
        GPU                         GPU activity
        Handle                      Handle usage
        XAMLActivity                XAML activity
        HTMLActivity                HTML activity
        DesktopComposition          Desktop composition activity
        XAMLAppResponsiveness       XAML App Responsiveness analysis
        HTMLResponsiveness          HTML Responsiveness analysis
        ReferenceSet                Reference Set analysis
        ResidentSet                 Resident Set analysis
        XAMLHTMLAppMemoryAnalysis   XAML/HTML application memory analysis
        UTC                         UTC Scenarios
        DotNET                      .NET Activity
        WdfTraceLoggingProvider     WDF Driver Activity
```

For detailed guidance on creating custom profiles, see the WPR.exe documentation.

Record ETW traces during operating system boot time

Use the `New-AutologgerConfig` cmdlet to collect events during system boot. Usage is very similar to the `New-EtwTraceSession` cmdlet, but providers added to the Autologger's configuration will only be enabled early at next boot. The overall workflow looks like this:

First, create a new Autologger config.

```
PS C:\> New-AutologgerConfig -Name BootPnpLog -LocalFilePath c:\bootpnp.etl
```

Add a ETW provider to it. This example uses the Kernel PnP provider. Invoke `Add-EtwTraceProvider` again, specifying the same Autologger name but a different GUID to enable boot trace collection from multiple sources.

```
Add-EtwTraceProvider -Guid {9c205a39-1250-487d-abd7-e831c6290539} -AutologgerName BootPnpLog
```

This does not start an ETW session immediately, but rather configures one to start at next boot. After rebooting, a new ETW session with the Autologger configuration name is automatically started with the added trace providers enabled. After Nano Server boots, the following command will stop the trace session after flushing the logged events to the associated trace file:

```
PS C:\> Remove-EtwTraceSession -Name BootPnpLog
```

To prevent another trace session from being auto-created at next boot, remove the Autologger configuration as follows:

```
PS C:\> Remove-AutologgerConfig -Name BootPnpLog
```

To collect boot and setup traces across a number of systems or on a diskless system, consider using Setup and Boot Event Collection.

Capture performance counter data

Usually, you monitor performance counter data with Perfmon.exe GUI. On Nano Server, use the `Typeperf.exe` command-line equivalent. For example:

Query available counters--you can filter the output to easily find the ones of interest.

```
PS C:\> typeperf.exe -q | Select-String UDPv6

\UDPv6\Datagrams/sec
\UDPv6\Datagrams Received/sec
\UDPv6\Datagrams No Port/sec
\UDPv6\Datagrams Received Errors
\UDPv6\Datagrams Sent/sec
```

Options allow specifying the number of times and the interval at which counter values are collected. In the example below, Processor Idle Time is collected 5 times every 3 seconds.

```
PS C:\> typeperf.exe \Processor Information(0,0)\% Idle Time -si 3 -sc 5

(PDH-CSV 4.0),\\ns-g2\Processor Information(0,0)\% Idle Time
09/15/2016 09:20:56.002,99.982990
09/15/2016 09:20:59.002,99.469634
09/15/2016 09:21:02.003,99.990081
09/15/2016 09:21:05.003,99.990454
09/15/2016 09:21:08.003,99.998577
Exiting, please wait...
The command completed successfully.
```

Other command-line options allow you to specify performance counter names of interest in a configuration file, redirecting output to a log file, among other things. See the typeperf.exe documentation for details.

You can also use Perfmon.exe's graphical interface remotely with Nano Server targets. When adding performance counters to the view, specify the Nano Server target in the computer name instead of the default *<Local computer>*.

Interact with the Windows Event Log

Nano Server supports the `Get-WinEvent` cmdlet, which provides Windows Event Log filtering and querying capabilities, both locally as well as on a remote computer. Detailed options and examples are available at the Get-WinEvent documentation page. This simple example retrieves the *Errors* noted in the *System* log during the past two days.

```
PS C:\> $StartTime = (Get-Date) - (New-TimeSpan -Day 2)
PS C:\> Get-WinEvent -FilterHashTable @{LogName='System'; Level=2; StartTime=$StartTime} | select
TimeCreated, Message

TimeCreated           Message
-----------           -------
9/15/2016 11:31:19 AM Task Scheduler service failed to start Task Compatibility module. Tasks may not be
able to reg...
9/15/2016 11:31:16 AM The Virtualization Based Security enablement policy check at phase 6 failed with
status: {File...
9/15/2016 11:31:16 AM The Virtualization Based Security enablement policy check at phase 0 failed with
status: {File...
```

Nano Server also supports `wevtutil.exe` which allows retrieving information about event logs and publishers. See wevtutil.exe documentation for more details.

Graphical interface tools

Web-based server management tools can be used to remotely manage Nano Server targets and present a Nano Server Event Log by using a web browser. Finally, the MMC snap-in Event Viewer (eventvwr.msc) can also be used to view logs -- just open it on a computer with a desktop and point it to a remote Nano Server.

Using Windows PowerShell Desired State Configuration with Nano Server

You can manage Nano Server as target nodes with Windows PowerShell Desired State Configuration (DSC). Currently, you can manage nodes running Nano Server with DSC in push mode only. Not all DSC features function with Nano Server.

For full details, see Using DSC on Nano Server.

Updating Nano Server

10/25/2021 • 6 minutes to read • Edit Online

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

Nano Server offers a variety of methods for staying up to date. Compared to other installation options of Windows Server, Nano Server follows a more active servicing model similar to that of Windows 10. These periodic releases are known as **Current Branch for Business (CBB)** releases. This approach supports customers who want to innovate more quickly and move at a cloud cadence of rapid development lifecycles. More information about CBB is available on the Windows Server Blog.

Between these CBB releases, Nano Server stays current with a series of *cumulative updates*. For example, the first cumulative update for Nano Server was released on September 26, 2016 with KB4093120. With this and subsequent cumulative updates, we provide various options for installing these updates on Nano Server. In this article, we'll use the KB3192366 update as an example to illustrate how to obtain and apply cumulative updates to Nano Server. For more information on the cumulative update model, see the Microsoft Update blog.

NOTE

If you install an optional Nano Server package from media or online repository, it won't have recent security fixes included. To avoid a version mismatch between the optional packages and base operating system, you should install the latest cumulative update immediately after installing any optional packages and **before** restarting the server.

In the case of the Cumulative Update for Windows Server 2016: September 26, 2016 (KB3192366), you should first install the latest Servicing Stack Update for Windows 10 Version 1607: August 23, 2016 as a prerequisite (KB3176936). For most of the options below, you need the .msu files containing the .cab update packages. Visit the Microsoft Update Catalog to download each of these update packages:

- https://catalog.update.microsoft.com/v7/site/Search.aspx?q=KB3192366
- https://catalog.update.microsoft.com/v7/site/Search.aspx?q=KB3176936

After downloading the .msu files from the Microsoft Update Catalog, save them to a network share or local directory such as C:\ServicingPackages. You can rename the .msu files based on their KB number as we've done below to make them easier to identify. Then use the EXPAND utility to extract the .cab files from the .msu files into separate directories and copy the .cabs into a single folder.

```
mkdir C:\ServicingPackages_expanded
mkdir C:\ServicingPackages_expanded\KB3176936
mkdir C:\ServicingPackages_expanded\KB3192366
Expand C:\ServicingPackages\KB3176936.msu -F:* C:\ServicingPackages_expanded\KB3176936
Expand C:\ServicingPackages\KB3192366.msu -F:* C:\ServicingPackages_expanded\KB3192366
mkdir C:\ServicingPackages_cabs
copy C:\ServicingPackages_expanded\KB3176936\Windows10.0-KB3176936-x64.cab C:\ServicingPackages_cabs
copy C:\ServicingPackages_expanded\KB3192366\Windows10.0-KB3192366-x64.cab C:\ServicingPackages_cabs
```

Now you can use the extracted .cab files to apply the updates to a Nano Server image in a few different ways, depending on your needs. The following options are presented in no particular order of preference - use the

option that makes the most sense for your environment.

> **NOTE**
> When using the DISM tools to service Nano Server, you must use a version of DISM that is the same as or newer than the version of Nano Server you're servicing. You can achieve this by running DISM from a matching version of Windows, installing a matching version of the Windows Asssessment and Deployment Kit (ADK), or running DISM on Nano Server itself.

Option 1: Integrate a cumulative update into a new image

If you are building a new Nano Server image, you can integrate the latest cumulative update directly into the image so that it's fully patched on first boot.

```
New-NanoServerImage -ServicingPackagePath 'C:\ServicingPackages_cabs\Windows10.0-KB3176936-x64.cab',
'C:\ServicingPackages_cabs\Windows10.0-KB3192366-x64.cab' -<other parameters>
```

Option 2: Integrate a cumulative update into an existing image

If you have an existing Nano Server image that you use as a baseline for creating specific instances of Nano Server, you can integrate the latest cumulative update directly into your existing baseline image so that machines created using the image are fully patched on first boot.

```
Edit-NanoServerImage -ServicingPackagePath 'C:\ServicingPackages_cabs\Windows10.0-KB3176936-x64.cab',
'C:\ServicingPackages_cabs\Windows10.0-KB3192366-x64.cab' -TargetPath .\NanoServer.wim
```

Option 3: Apply the cumulative update to an existing offline VHD or VHDX

If you have an existing virtual hard disk (VHD or VHDX), you can use the DISM tools to apply the update to the virtual hard disk. You need to make sure the disk is not in use either by shutting down any VMs using the disk or unmounting the virtual hard disk file.

- Using PowerShell

```
Mount-WindowsImage -ImagePath .\NanoServer.vhdx -Path .\MountDir -Index 1
Add-WindowsPackage -Path .\MountDir -PackagePath  C:\ServicingPackages_cabs
Dismount-WindowsImage -Path .\MountDir -Save
```

- Using dism.exe

```
dism.exe /Mount-Image /ImageFile:C:\NanoServer.vhdx /Index:1 /MountDir:C:\MountDir
dism.exe /Image:C:\MountDir /Add-Package /PackagePath:C:\ServicingPackages_cabs
dism.exe /Unmount-Image /MountDir:C:\MountDir /Commit
```

Option 4: Apply the cumulative update to a running Nano Server

If you have a running Nano Server VM or physical host and you've downloaded the .cab file for the update, you can use the DISM tools to apply the update while the operating system is online. You will need to copy the .cab file locally on the Nano Server or to an accessible network location. If you're applying a servicing stack update, make sure to restart the server after applying the servicing stack update before applying additional updates.

NOTE

If you've created the Nano Server VHD or VHDX image using the New-NanoServerImage cmdlet and didn't specify a MaxSize for the virtual hard disk file, the default size of 4GB is too small to apply the cumulative update. Prior to installing the update, use Hyper-V Manager, Disk Management, PowerShell, or other tool to expand the size of the virtual hard disk and system volume to at least 10GB, or use the ScratchDir parameter on the DISM tools to set the scratch directory to a volume with at least 10GB of free space.

```
$s = New-PSSession -ComputerName (Read-Host "Enter Nano Server IP address") -Credential (Get-Credential)
Copy-Item -ToSession $s -Path C:\ServicingPackages_cabs -Destination C:\ServicingPackages_cabs -Recurse
Enter-PSSession $s
```

- Using PowerShell

```
# Apply the servicing stack update first and then restart
Add-WindowsPackage -Online -PackagePath C:\ServicingPackages_cabs\Windows10.0-KB3176936-x64.cab
Restart-Computer; exit

# After restarting, apply the cumulative update and then restart
Enter-PSSession -ComputerName (Read-Host "Enter Nano Server IP address") -Credential (Get-Credential)
Add-WindowsPackage -Online -PackagePath C:\ServicingPackages_cabs\Windows10.0-KB3192366-x64.cab
Restart-Computer; exit
```

- Using dism.exe

```
# Apply the servicing stack update first and then restart
dism.exe /Online /Add-Package /PackagePath:C:\ServicingPackages_cabs\Windows10.0-KB3176936-x64.cab

# After the operation completes successfully and you are prompted to restart, it's safe to
# press Ctrl+C to cancel the pipeline and return to the prompt
Restart-Computer; exit

# After restarting, apply the cumulative update and then restart
Enter-PSSession -ComputerName (Read-Host "Enter Nano Server IP address") -Credential (Get-Credential)
dism.exe /Online /Add-Package /PackagePath:C:\ServicingPackages_cabs\Windows10.0-KB3192366-x64.cab
Restart-Computer; exit
```

Option 5: Download and install the cumulative update to a running Nano Server

If you have a running Nano Server VM or physical host, you can use the Windows Update WMI provider to download and install the update while the operating system is online. With this method, you don't need to download the .msu file separately from the Microsoft Update Catalog. The WMI provider will detect, download, and install all available updates at once.

```
Enter-PSSession -ComputerName (Read-Host "Enter Nano Server IP address") -Credential (Get-Credential)
```

- Scan for available updates

```
$ci = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName
MSFT_WUOperationsSession
$result = $ci | Invoke-CimMethod -MethodName ScanForUpdates -Arguments
@{SearchCriteria="IsInstalled=0";OnlineScan=$true}
$result.Updates
```

- Install all available updates

```
$ci = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName
MSFT_WUOperationsSession
Invoke-CimMethod -InputObject $ci -MethodName ApplyApplicableUpdates
Restart-Computer; exit
```

- Get a list of installed updates

```
$ci = New-CimInstance -Namespace root/Microsoft/Windows/WindowsUpdate -ClassName
MSFT_WUOperationsSession
$result = $ci | Invoke-CimMethod -MethodName ScanForUpdates -Arguments
@{SearchCriteria="IsInstalled=1";OnlineScan=$true}
$result.Updates
```

Additional Options

Other methods for updating Nano Server might overlap or complement the options above. Such options include using Windows Server Update Services (WSUS), System Center Virtual Machine Manager (VMM), Task Scheduler, or a non-Microsoft solution.

- Configuring Windows Update for WSUS by setting the following registry keys:
  - WUServer
  - WUStatusServer (generally uses the same value as WUServer)
  - UseWUServer
  - AUOptions
- Managing Fabric Updates in VMM
- Registering a Scheduled Task

Developing for Nano Server

10/25/2021 • 2 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

These topics explain important differences in PowerShell on Nano Server and also provide guidance for developing your own PowerShell cmdlets for use on Nano Server.

- PowerShell on Nano Server
- Developing PowerShell Cmdlets for Nano Server

Using Windows PowerShell remoting

To manage Nano Server with Windows PowerShell remoting, you need to add the IP address of the Nano Server to your management computer's list of trusted hosts, add the account you are using to the Nano Server's administrators, and enable CredSSP if you plan to use that feature.

NOTE

If the target Nano Server and your management computer are in the same AD DS forest (or in forests with a trust relationship), you should not add the Nano Server to the trusted hosts list--you can connect to the Nano Server by using its fully qualified domain name, for example: PS C:> Enter-PSSession -ComputerName nanoserver.contoso.com -Credential (Get-Credential)

To add the Nano Server to the list of trusted hosts, run this command at an elevated Windows PowerShell prompt:

```
Set-Item WSMan:\localhost\Client\TrustedHosts <IP address of Nano Server>
```

To start the remote Windows PowerShell session, start an elevated local Windows PowerShell session, and then run these commands:

```
$ip = \<IP address of Nano Server>
$user = $ip\Administrator
Enter-PSSession -ComputerName $ip -Credential $user
```

You can now run Windows PowerShell commands on the Nano Server as normal.

NOTE

Not all Windows PowerShell commands are available in this release of Nano Server. To see which are available, run `Get-Command -CommandType Cmdlet`

Stop the remote session with the command `Exit-PSSession`

Using Windows PowerShell CIM sessions over WinRM

You can use CIM sessions and instances in Windows PowerShell to run WMI commands over Windows Remote Management (WinRM).

Start the CIM session by running these commands in a Windows PowerShell prompt:

```
$ip = <IP address of the Nano Server\>
$ip\Administrator
$cim = New-CimSession -Credential $user -ComputerName $ip
```

With the session established, you can run various WMI commands, for example:

```
Get-CimInstance -CimSession $cim -ClassName Win32_ComputerSystem | Format-List *
Get-CimInstance -CimSession $Cim -Query SELECT * from Win32_Process WHERE name LIKE 'p%'
```

PowerShell on Nano Server

10/25/2021 • 3 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

PowerShell Editions

Starting with version 5.1, PowerShell is available in different editions which denote varying feature sets and platform compatibility.

- **Desktop Edition:** Built on .NET Framework and provides compatibility with scripts and modules targeting versions of PowerShell running on full footprint editions of Windows such as Server Core and Windows Desktop.
- **Core Edition:** Built on .NET Core and provides compatibility with scripts and modules targeting versions of PowerShell running on reduced footprint editions of Windows such as Nano Server and Windows IoT.

The running edition of PowerShell is shown in the PSEdition property of $PSVersionTable.

```
$PSVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.14300.1000
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
CLRVersion                     4.0.30319.42000
BuildVersion                   10.0.14300.1000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

Module authors can declare their modules to be compatible with one or more PowerShell editions using the CompatiblePSEditions module manifest key. This key is only supported on PowerShell 5.1 or later.

```
New-ModuleManifest -Path .\TestModuleWithEdition.psd1 -CompatiblePSEditions Desktop,Core -PowerShellVersion
5.1
$moduleInfo = Test-ModuleManifest -Path \TestModuleWithEdition.psd1
$moduleInfo.CompatiblePSEditions
Desktop
Core

$moduleInfo | Get-Member CompatiblePSEditions

   TypeName: System.Management.Automation.PSModuleInfo

Name                 MemberType Definition
----                 ---------- ----------
CompatiblePSEditions Property   System.Collections.Generic.IEnumerable[string] CompatiblePSEditions {get;}
```

When getting a list of available modules, you can filter the list by PowerShell edition.

```
Get-Module -ListAvailable | ? CompatiblePSEditions -Contains Desktop

    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   1.0        ModuleWithPSEditions

Get-Module -ListAvailable | ? CompatiblePSEditions -Contains Core | % CompatiblePSEditions
Desktop
Core
```

Script authors can prevent a script from executing unless it is run on a compatible edition of PowerShell using the PSEdition parameter on a #requires statement.

```
Set-Content C:\script.ps1 -Value #requires -PSEdition Core
Get-Process -Name PowerShell
Get-Content C:\script.ps1
#requires -PSEdition Core
Get-Process -Name PowerShell

C:\script.ps1
C:\script.ps1 : The script 'script.ps1' cannot be run because it contained a #requires statement for
PowerShell editions 'Core'. The edition of PowerShell that is required by the script does not match the
currently running PowerShell Desktop edition.
At line:1 char:1
+ C:\script.ps1
+ ~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (script.ps1:String) [], RuntimeException
    + FullyQualifiedErrorId : ScriptRequiresUnmatchedPSEdition
```

Differences in PowerShell on Nano Server

Nano Server includes PowerShell Core by default in all Nano Server installations. PowerShell Core is a reduced footprint edition of PowerShell that is built on .NET Core and runs on reduced footprint editions of Windows, such as Nano Server and Windows IoT Core. PowerShell Core functions in the same way as other editions of PowerShell, such as Windows PowerShell running on Windows Server 2016. However, the reduced footprint of Nano Server means that not all PowerShell features from Windows Server 2016 are available in PowerShell Core on Nano Server.

Windows PowerShell features not available in Nano Server

- ADSI, ADO, and WMI type adapters
- Enable-PSRemoting, Disable-PSRemoting (PowerShell remoting is enabled by default; see the Using Windows PowerShell Remoting section of Install Nano Server).
- Scheduled jobs and PSScheduledJob module
- Computer cmdlets for joining a domain { Add | Remove } (for different methods to join Nano Server to a domain, see the Joining Nano Server to a domain section of Install Nano Server).
- Reset-ComputerMachinePassword, Test-ComputerSecureChannel
- Profiles (you can add a startup script for incoming remote connections with `Set-PSSessionConfiguration`)
- Clipboard cmdlets
- EventLog cmdlets { Clear | Get | Limit | New | Remove | Show | Write } (use the New-WinEvent and Get-WinEvent cmdlets instead).
- Get-PfxCertificate cmdlet
- TraceSource cmdlets { Get | Set }
- Counter cmdlets { Get | Export | Import }
- Some web-related cmdlets { New-WebServiceProxy, Send-MailMessage, ConvertTo-Html }
- Logging and tracing using PSDiagnostics module
- Get-HotFix (to obtain and manage updates on Nano Server, see Manage Nano Server).
- Implicit remoting cmdlets { Export-PSSession | Import-PSSession }
- New-PSTransportOption
- PowerShell transactions and Transaction cmdlets { Complete | Get | Start | Undo | Use }
- PowerShell Workflow infrastructure, modules, and cmdlets
- Out-Printer
- Update-List
- WMI v1 cmdlets: Get-WmiObject, Invoke-WmiMethod, Register-WmiEvent, Remove-WmiObject, Set-WmiInstance (use CimCmdlets module instead.)

Using Windows PowerShell Desired State Configuration with Nano Server

You can manage Nano Server as target nodes with Windows PowerShell Desired State Configuration (DSC). Currently, you can manage nodes running Nano Server with DSC in push mode only. Not all DSC features function with Nano Server.

For full details, see Using DSC on Nano Server.

Developing PowerShell Cmdlets for Nano Server

10/25/2021 • 9 minutes to read • Edit Online

Applies to: Windows Server 2016

IMPORTANT

Starting in Windows Server, version 1709, Nano Server will be available only as a container base OS image. Check out Changes to Nano Server to learn what this means.

Overview

Nano Server includes PowerShell Core by default in all Nano Server installations. PowerShell Core is a reduced-footprint edition of PowerShell that is built on .NET Core and runs on reduced-footprint editions of Windows, such as Nano Server and Windows IoT Core. PowerShell Core functions in the same way as other editions of PowerShell, such as Windows PowerShell running on Windows Server 2016. However, the reduced footprint of Nano Server means that not all PowerShell features from Windows Server 2016 are available in PowerShell Core on Nano Server.

If you have existing PowerShell cmdlets that you'd like to run on Nano Server, or are developing new ones for that purpose, this topic includes tips and suggestions that should help make that easier.

PowerShell editions

Starting with version 5.1, PowerShell is available in different editions which denote varying feature sets and platform compatibility.

- **Desktop Edition:** Built on .NET Framework and provides compatibility with scripts and modules targeting versions of PowerShell running on full footprint editions of Windows such as Server Core and Windows Desktop.
- **Core Edition:** Built on .NET Core and provides compatibility with scripts and modules targeting versions of PowerShell running on reduced footprint editions of Windows such as Nano Server and Windows IoT.

The running edition of PowerShell is shown in the PSEdition property of $PSVersionTable.

```
$PSVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.14300.1000
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
CLRVersion                     4.0.30319.42000
BuildVersion                   10.0.14300.1000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

Module authors can declare their modules to be compatible with one or more PowerShell editions using the CompatiblePSEditions module manifest key. This key is only supported on PowerShell 5.1 or later.

```
New-ModuleManifest -Path .\TestModuleWithEdition.psd1 -CompatiblePSEditions Desktop,Core -PowerShellVersion
5.1
$moduleInfo = Test-ModuleManifest -Path \TestModuleWithEdition.psd1
$moduleInfo.CompatiblePSEditions
Desktop
Core

$moduleInfo | Get-Member CompatiblePSEditions

   TypeName: System.Management.Automation.PSModuleInfo

Name                 MemberType Definition
----                 ---------- ----------
CompatiblePSEditions Property   System.Collections.Generic.IEnumerable[string] CompatiblePSEditions {get;}
```

When getting a list of available modules, you can filter the list by PowerShell edition.

```
Get-Module -ListAvailable | ? CompatiblePSEditions -Contains Desktop

    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   1.0        ModuleWithPSEditions

Get-Module -ListAvailable | ? CompatiblePSEditions -Contains Core | % CompatiblePSEditions
Desktop
Core
```

Script authors can prevent a script from executing unless it is run on a compatible edition of PowerShell using the PSEdition parameter on a #requires statement.

```
Set-Content C:\script.ps1 -Value #requires -PSEdition Core
Get-Process -Name PowerShell
Get-Content C:\script.ps1
#requires -PSEdition Core
Get-Process -Name PowerShell

C:\script.ps1
C:\script.ps1 : The script 'script.ps1' cannot be run because it contained a #requires statement for
PowerShell editions 'Core'. The edition of PowerShell that is required by the script does not match the
currently running PowerShell Desktop edition.
At line:1 char:1
+ C:\script.ps1
+ ~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (script.ps1:String) [], RuntimeException
    + FullyQualifiedErrorId : ScriptRequiresUnmatchedPSEdition
```

Installing Nano Server

Quick-start and detailed steps for installing Nano Server on virtual or physical machines are provided in Install Nano Server, which is the parent topic for this one.

NOTE

For development work on Nano Server, you might find it useful to install Nano Server by using the -Development parameter of New-NanoServerImage. This will enable installation of unsigned drivers, copy debugger binaries, open a port for debugging, enable test signing and enable installation of AppX packages without a developer license. For example:

```
New-NanoServerImage -DeploymentType Guest -Edition Standard -MediaPath \\Path\To\Media\en_us -BasePath
.\Base -TargetPath .\NanoServer.wim -Development
```

Determining the type of cmdlet implementation

PowerShell supports a number of implementation types for cmdlets, and the one you've used determines the process and tools involved in creating or porting it to work on Nano Server. Supported implementation types are:

- CIM - consists of CDXML files layered over CIM (WMIv2) providers
- .NET - consists of .NET assemblies implementing managed cmdlet interfaces, typically written in C#
- PowerShell Script - consists of script modules (.psm1) or scripts (.ps1) written in the PowerShell language

If you're not sure which implementation you've used for existing cmdlets you want to port, install your product or feature and then look for the PowerShell module folder in one of the following locations:

- %windir%\system32\WindowsPowerShell\v1.0\Modules
- %ProgramFiles%\WindowsPowerShell\Modules
- %UserProfile%\Documents\WindowsPowerShell\Modules
- <your product installation location>

  Check in these locations for these details:

  - CIM cmdlets have .cdxml file extensions.
  - .NET cmdlets have .dll file extensions, or have assemblies installed to the GAC listed in the .psd1 file under the RootModule, ModuleToProcess, or NestedModules fields.
- PowerShell script cmdlets have .psm1 or .ps1 file extensions.

Porting CIM cmdlets

Generally, these cmdlets should work in Nano Server without any conversion necessary. However, you must port the underlying WMI v2 provider to run on Nano Server if that has not already been done.

Building C++ for Nano Server

To get C++ DLLs working on Nano Server, compile them for Nano Server rather than for a specific edition.

For prerequisites and a walkthrough of developing C++ on Nano Server, see Developing Native Apps on Nano Server.

Porting .NET cmdlets

Most C# code is supported on Nano Server. You can use ApiPort to scan for incompatible APIs.

Powershell Core SDK

The module Microsoft.PowerShell.NanoServer.SDK is available in the PowerShell Gallery to facilitate developing .NET cmdlets using Visual Studio 2015 Update 2 that target the versions of CoreCLR and PowerShell Core available in Nano Server. You can install the module using PowerShellGet with this command:

```
Find-Module Microsoft.PowerShell.NanoServer.SDK -Repository PSGallery | Install-Module -Scope <scope>
```

The PowerShell Core SDK module exposes cmdlets to set up the correct CoreCLR and PowerShell Core reference assemblies, create a C# project in Visual Studio 2015 targeting those reference assemblies, and set up the remote debugger on a Nano Server machine so that developers can debug their .NET cmdlets running on Nano Server remotely in Visual Studio 2015.

The PowerShell Core SDK module requires Visual Studio 2015 Update 2. If you do not have Visual Studio 2015 installed, you can install Visual Studio Community 2015.

The SDK module also depends on the following feature to be installed in Visual Studio 2015:

- Windows and Web Development -> Universal Windows App Development Tools -> Tools (1.3.1) and Windows 10 SDK

Review your Visual Studio installation before using the SDK module to ensure these prerequisites are satisfied. Make sure you select to install the above feature during the Visual Studio installation, or modify your existing Visual Studio 2015 installation to install it.

The PowerShell Core SDK module includes the following cmdlets:

- New-NanoCSharpProject: Creates a new Visual Studio C# project targeting CoreCLR and PowerShell Core included in the Windows Server 2016 release of Nano Server.
- Show-SdkSetupReadMe: Opens the SDK root folder in File Explorer and opens the README.txt file for manual setup.
- Install-RemoteDebugger: Installs and configures the Visual Studio remote debugger on a Nano Server machine.
- Start-RemoteDebugger: Starts the remote debugger on a remote machine running Nano Server.
- Stop-RemoteDebugger: Stops the remote debugger on a remote machine running Nano Server.

For detailed information about how to use those cmdlets, run Get-Help on each cmdlet after installing and importing the module as follows:

```
Get-Command -Module Microsoft.PowerShell.NanoServer.SDK | Get-Help -Full
```

Searching for compatible APIs

You can search in the API catalog for .NET Core or disassemble Core CLR reference assemblies. For more information about platform portability of .NET APIs, see Platform Portability

PInvoke

In the Core CLR that Nano Server uses, some fundamental DLLs such as kernel32.dll and advapi32.dll were split into numerous API sets, so you'll need to ensure that your PInvokes reference the correct API. Any incompatibility will manifest as a runtime error.

For a list of native APIs supported on Nano Server, see Nano Server APIs.

Building C# for Nano Server

Once a C# project is created in Visual Studio 2015 by using `New-NanoCSharpProject`, you can simply build it in Visual Studio by clicking the **Build** menu and selecting **Build Project** or **Build Solution**. The generated assemblies will be targeting the correct CoreCLR and PowerShell Core shipped in Nano Server, and you can just copy the assemblies to a computer running Nano Server and use them.

Building managed C++ (CPP/CLI) for Nano Server

Managed C++ is not supported for CoreCLR. When porting to CoreCLR, rewrite managed C++ code in C# and make all native calls through PInvoke.

Porting PowerShell script cmdlets

PowerShell Core has full PowerShell language parity with other editions of PowerShell, including the edition running on Windows Server 2016 and Windows 10. However, when porting PowerShell script cmdlets to Nano Server, keep these factors in mind:

- Are there dependencies on other cmdlets? If so, are those cmdlets available on Nano Server. See PowerShell on Nano Server for information about what is not available.
- If you have dependencies on assemblies that are loaded at runtime, will they still work?
- How can you debug the script remotely?
- How can you migrate from WMI .Net to MI .Net?

Dependency on built-in cmdlets

Not all cmdlets in Windows Server 2016 are available on Nano Server (see PowerShell on Nano Server). The best approach is to set up a Nano Server virtual machine and discover whether the cmdlets you need are available. To do this, run `Enter-PSSession` to connect to the target Nano Server and then run `Get-Command -CommandType Cmdlet, Function` to get the list of available cmdlets.

Consider using PowerShell classes

Add-Type is supported on Nano Server for compiling inline C# code. If you're writing new code or porting existing code, you might also consider using PowerShell classes to define custom types. You can use PowerShell classes for property bag scenarios as well as for Enums. If you need to do a PInvoke, do this via C# using Add-Type or in a pre-compiled assembly. Here's a sample showing the use of Add-Type:

```
Add-Type -ReferencedAssemblies ([Microsoft.Management.Infrastructure.Ciminstance].Assembly.Location) -
TypeDefinition @'
public class TestNetConnectionResult
{
   // The compute name
   public string ComputerName = null;
   // The Remote IP address used for connectivity
   public System.Net.IPAddress RemoteAddress = null;
}
'@
# Create object and set properties
$result = New-Object TestNetConnectionResult
$result.ComputerName = Foo
$result.RemoteAddress = 1.1.1.1
```

This sample shows using PowerShell classes on Nano Server:

```
class TestNetConnectionResult
{
   # The compute name
  [string] $ComputerName

  #The Remote IP address used for connectivity
  [System.Net.IPAddress] $RemoteAddress
}
# Create object and set properties
$result = [TestNetConnectionResult]::new()
$result.ComputerName = Foo
$result.RemoteAddress = 1.1.1.1
```

Remotely debugging scripts

To remotely debug a script, connect to the remote computer using `Enter-PSsession` from the PowerShell ISE.

Once inside the session, you can run `psedit <file_path>` and a copy of the file will be open in your local PowerShell ISE. Then, you can debug the script as if it were running locally by setting breakpoints. Also, any changes you make to this file will be saved in the remote version.

Migrating from WMI .NET to MI .NET

WMI .NET is not supported, so all cmdlets using the old API must migrate to the supported WMI API: MI. NET. You can access MI .NET directly through C# or through the cmdlets in the CimCmdlets module.

CimCmdlets module

The WMI v1 cmdlets (e.g., Get-WmiObject) are not supported on Nano Server. However, the CIM cmdlets (e.g., Get-CimInstance) in the CimCmdlets module are supported. The CIM cmdlets map pretty closely to the WMI v1 cmdlets. For example, Get-WmiObject correlates with Get-CimInstance using very similar parameters. Method invocation syntax is slightly different, but is well documented via Invoke-CimMethod. Be careful regarding parameter typing. MI .NET has stricter requirements regarding method parameter types.

C# API

WMI .NET wraps the WMIv1 interface, while MI .NET wraps the WMIv2 (CIM) interface. The classes exposed might be different, but the underlying operations are very similar. You enumerate or get instances of objects and invoke operations on them to accomplish tasks.