6

Managing Windows Update

In this chapter, we cover the following recipes:

* Installing Windows Server Update Services
* Configuring WSUS update synchronization
* Configuring the Windows Update Client
* Creating computer target groups
* Configuring WSUS automatic approvals
* Managing WSUS updates

# Introduction

Keeping your systems, client and server, up to date with patches and updates is an important task undertaken by Windows administrators. Windows Server Update Services (WSUS) is a feature of Windows Server 2019 that enables you to manage the download and distribution of updates to your organization's computers.

In addition to updating Windows itself, WSUS also enables you to manage patches and updates for a wide variety of Microsoft software products. Thus, an update you download from Microsoft and distribute via WSUS may apply to Windows itself as well as Office and a huge range of other Microsoft software products.

In this chapter, you see how to install and configure both WSUS server and WSUS client computers. The recipes examine the management, approval, and installation of updates, and how you can report on the status of update installation.

# Installing Windows Update Services

WSUS is a feature within Windows Server 2019. To use Windows Update Services, you first install the WSUS Windows feature and then do basic configuration and setup. In this recipe, you install WSUS and review the results of that installation.

## Getting ready

This recipe uses the WSUS1 server, a member server in the Reskit.Org domain. At the start of this recipe, WSUS1 has no additional features or software loaded.

## How to do it...

1. Install the Windows Update feature and tools:

$IFHT = @{

Name = 'UpdateServices'

IncludeManagementTools = $true

}

Install-WindowsFeature @IFHT

1. Determine the features installed on the WSUS1 server after installation of WSUS:

Get-WindowsFeature | Where-Object Installed

1. Create a folder for WSUS update content:

$WSUSDir = 'C:\WSUS'

If (-Not (Test-Path -Path $WSUSDir -ErrorAction SilentlyContinue))

{New-Item -Path $WSUSDir -ItemType Directory| Out-Null}

1. Perform post-installation configuration using WsusUtil.Exe:

$CMD ="$env:ProgramFiles\" + 'Update Services\Tools\WsusUtil.exe'

& $CMD Postinstall CONTENT\_DIR=$WSUSDir

1. View the post-installation log file:

$LOG = "$env:localappdata\temp\WSUS\_Post\*.log"

Get-ChildItem -Path $LOG

1. View the WSUS website created on WSUS1:

Get-Website -Name ws\* | Format-Table -AutoSize

1. View the cmdlets in the UpdateServices module:

Get-Command -Module UpdateServices

1. Inspect TypeName and properties of the WSUS1 server:

$WSUSServer = Get-WsusServer

$WSUSServer.GetType().Fullname

$WSUSServer | Select-Object -Property \*

1. Examine the methods available:

($WSUSServer | Get-Member -MemberType Method).count

$WSUSServer | Get-Member -MemberType Method

1. Examine the configuration of the WSUS1 server:

$WSUSServer.GetConfiguration() |

Select-Object -Property SyncFromMicrosoftUpdate,LogFilePath

1. View product categories after initial install:

$WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer

$WSUSProducts.Count

$WSUSProducts

1. Display subscription information:

$WSUSSubscription = $WSUSServer.GetSubscription()

$WSUSSubscription | Select-Object -Property \* |

Format-List

1. Update the categories of products available:

$WSUSSubscription.StartSynchronization()

Do {

Write-Output $WSUSSubscription.GetSynchronizationProgress()

Start-Sleep -Seconds 5

}

While ($WSUSSubscription.GetSynchronizationStatus() -ne

'NotProcessing')

1. Once synchronization is complete, check the results of the synchronization:

$WSUSSubscription.GetLastSynchronizationInfo()

1. Examine the categories available after synchronization:

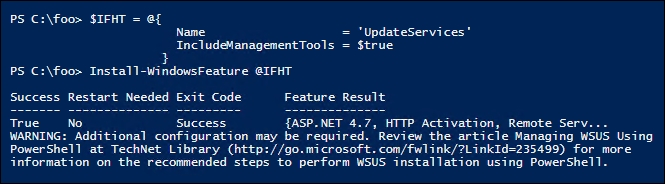
$WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer

$WSUSProducts.Count

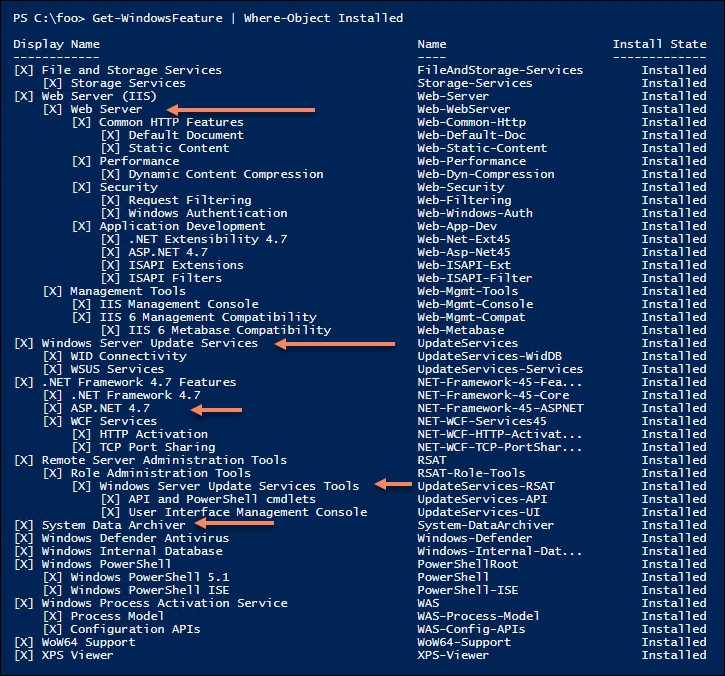
$WSUSProducts

## How it works...

In step 1, you install the Windows Update Services feature and the associated tools, which looks like this:

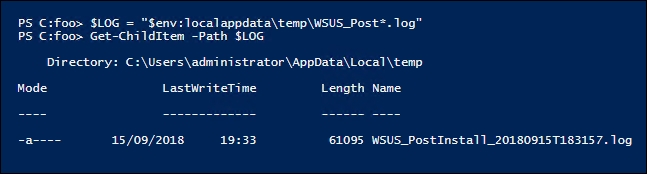


Adding the Update Services feature adds a number of additional related services, as you can see in step 2, which looks like this:

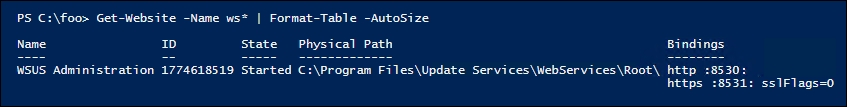


In step 3, you create, silently, a folder that you are going to use to hold WSUS content. In step 4, you perform the post-installation task using the wsusutil.exe console command, which also produces no output.

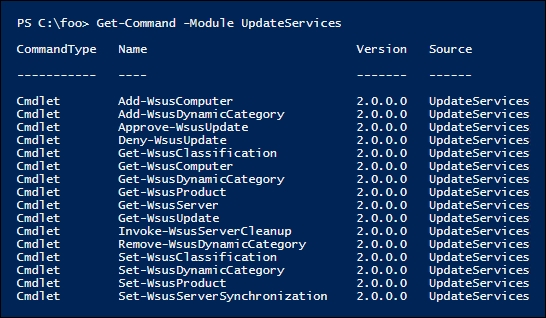
The wsusutil.exe application creates an output log file containing details of the actions taken by the application. In step 5, you look at the log file that wsusutil.exe created in step 4:



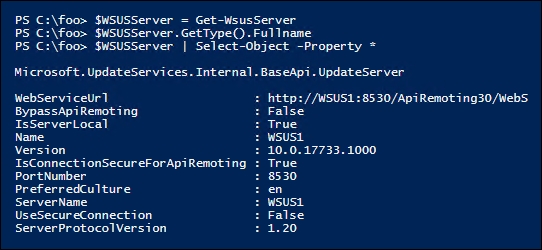
The WSUS installer also installs a website on a WSUS to communicate with WSUS clients. In step 6, you view the site, as you can see here:



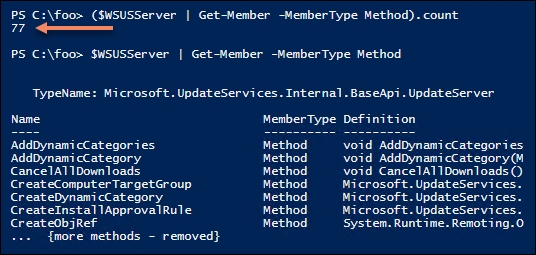
In step 7, you examine the commands contained in the UpdateServices module that you installed as part of step 1. The output of this step looks like this:



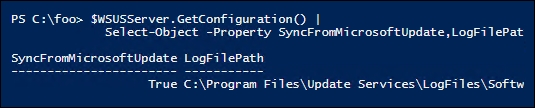
You examine the status of your WSUS server, in step 8, by using the Get-WsusServer cmdlet, which returns an UpdateServer object, which looks like this:



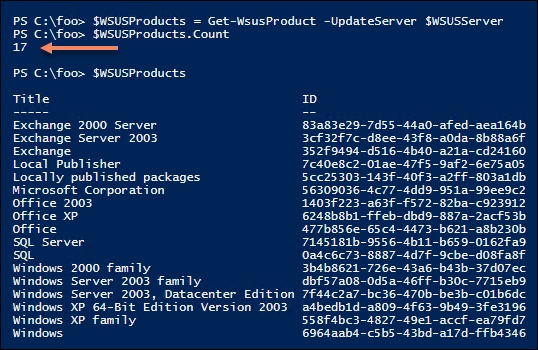
The $WSUSServer object you instantiated in step 8 contains a large number of methods you can call to manage aspects of the WSUS server. As you can see from the output of step 9, there are 77 methods—to save space, only a few are listed, as shown here:



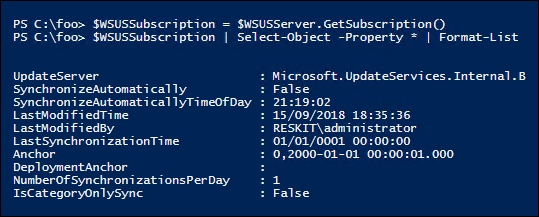
WSUS also generates an important log file, SoftwareDistribution.log, that can be invaluable for troubleshooting. You can see the filename, in step 10, which looks like this:



Following the initial installation and configuration done so far, in step 11, you can see that the WSUS1 server now is to get updates for a small set of products (17 in total), as you can see here:



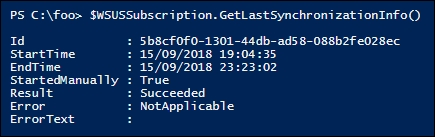
In step 12, you retrieve and view the WSUS server's subscription details, which looks like this:



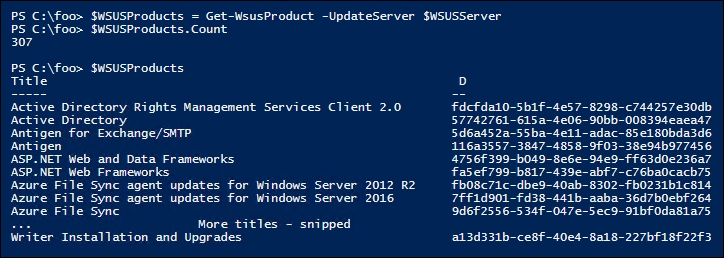
In step 13, you perform a full synchronization by invoking the StartSynchronization() method of the WSUS server object. This is an asynchronous operation—when you call the method, WSUS carries out the server update process in the background. You can call the GetSynchronizizationStatus() method to view the status, as you can see in step 13. The synchronization process is not overly fast and can take several hours to complete. Truncated for brevity, the output of this step looks something like this:



Once the synchronization has completed, in step 14, you review a summary of the results, which looks like this:



Now that this first full synchronization has taken place, WSUS is able to support a larger number of Microsoft products, as you can see in the output from step 15:



## There's more…

In step 2, you can see that installing WSUS also installs the web-server (Internet Information Server) along with ASP.NET 4.7 on WSUS1.

In step 3, you create a folder to hold downloaded updates that you intend to review then deploy to your organization. This folder can get large, especially when you implement multi-language updates. You should hold your updates on a volume that is likely to have adequate space going forward. Making the volume fault-tolerant is also important as you plan and deploy WSUS.

In step 13, you perform a full sync with the Windows Update servers. This can take several hours. You may wish to change the value used in the Start-Sleep command to a larger value (otherwise you could end up with thousands of lines of output!).

In this recipe, you installed WSUS on a single server. You can use WSUS on multiple servers, which is appropriate for supporting larger networks. You can set up a WSUS server to synchronize from other WSUS servers on the network, you can use web proxies, and you can work with SQL Server instead of the Windows Internal Database.

## See also

The WSUS server requirements and deployment scenarios are documented at <https://docs.microsoft.com/windows-server/administration/windows-server-update-services/plan/plan-your-wsus-deployment>.

The commands in the UpdateServices module are useful, but many of the tasks you are going to perform in PowerShell make use of the UpdateServer and Subscription objects and their methods.

MSDN contains documentation on the objects inside the Microsoft.UpdateServices.Administration namespace. You can view the documentation at <https://docs.microsoft.com/previous-versions/windows/desktop/ms748969(v=vs.85)>.

# Configuring WSUS update synchronization

After you install WSUS and do a basic synchronization, you configure WSUS to identify the products for which your organization requires product updates as well as the classifications of updates WSUS should download.

Once these are defined, you can synchronize updates manually or you can build an update schedule. This enables your WSUS server to download only the updates for the product categories and update classifications you have selected, both at a time of your choosing. The first initial synchronization can take hours, depending on your selections. Subsequent synchronizations pull only the newest updates since the last synchronization.

## Getting ready

This recipe configures the WSUS1 WSUS server, which is a domain-joined system. This recipe assumes you are starting with the just-installed WSUS as performed in the Installing Windows Update Services recipe.

## How to do it...

1. Discover the versions of Windows Server supported by Windows Update:

Get-WsusProduct |

Where-Object -FilterScript {$\_.Product.Title -match

'^Windows Server'}

1. Also, get update titles for Windows 10:

Get-WsusProduct -TitleIncludes 'Windows 10'

1. Create and view a list of software product titles to include:

$CHP =

(Get-WsusProduct |

Where-Object -FilterScript {$\_.product.title -match

'^Windows Server'}).Product.Title

$CHP += @('Microsoft SQL Server 2016','Windows 10')

$CHP

1. Assign the desired products to include in Windows Update:

Get-WsusProduct |

Where-Object {$PSItem.Product.Title -in $CHP} |

Set-WsusProduct

1. Get a list of the distinct categories of updates you can retrieve from Windows Update for distribution to your client hosts:

Get-WsusClassification

1. Create and view a list of desired update classifications to make available on your WSUS server:

$CCL = @('Critical Updates',

'Definition Updates',

'Security Updates',

'Service Packs',

'Update Rollups',

'Updates')

1. Now set the list of desired update classifications in WSUS:

Get-WsusClassification |

Where-Object {$\_.Classification.Title -in

$CCL} |

Set-WsusClassification

1. Get current subscriptions:

$WSUSServer = Get-WsusServer

$WSUSSubscription = $WSUSServer.GetSubscription()

1. Start synchronizing available updates based on configured categories:

$WSUSSubscription.StartSynchronization()

1. Next, loop and wait for synchronization to complete:

$IntervalSeconds = 5

$NP = 'NotProcessing'

Do {

$WSUSSubscription.GetSynchronizationProgress()

Start-Sleep -Seconds $IntervalSeconds

} While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)

1. Synchronize the updates; this can take a long while to complete:

$IntervalSeconds = 1

$NP = 'NotProcessing'

# Wait for synchronizing to start

Do {

Write-Output $WSUSSubscription.GetSynchronizationProgress()

Start-Sleep -Seconds $IntervalSeconds

}

While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)

# Wait for all phases of process to end

Do {

Write-Output $WSUSSubscription.GetSynchronizationProgress()

Start-Sleep -Seconds $IntervalSeconds

}

Until ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)

1. When the final loop is complete, check the results of the synchronization:

$WSUSSubscription.GetLastSynchronizationInfo()

1. Finally, going forward, ensure that synchronization happens once a day:

$WSUSSubscription = $WSUSServer.GetSubscription()

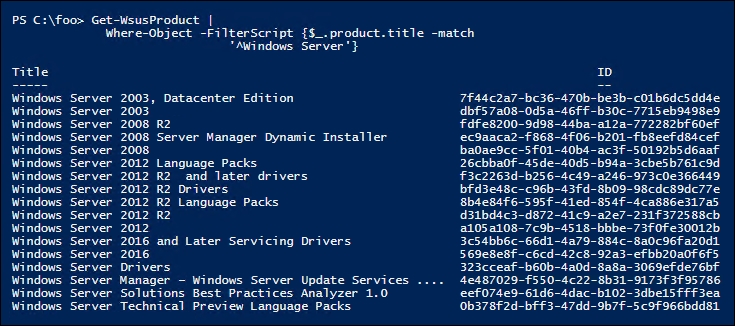
$WSUSSubscription.SynchronizeAutomatically = $true

$WSUSSubscription.NumberOfSynchronizationsPerDay = 1

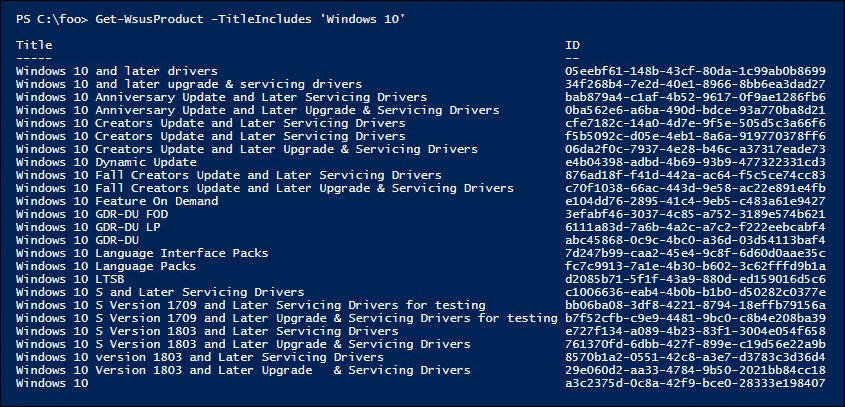
$WSUSSubscription.Save()

## How it works...

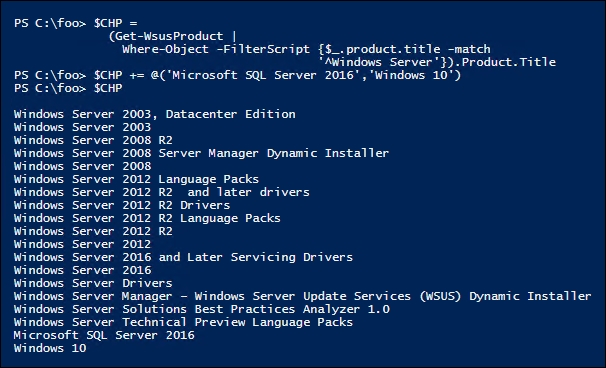
In step 1, you examine the product updates available:



In step 2, you review the version of Windows 10 that you can update using WSUS and Windows Update, like this:

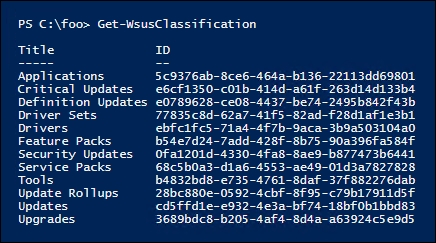


In most cases, you probably do not want to support all Microsoft products. To achieve that, you begin, in step 3, by creating a list of the products you do want to support. In this step, you include all versions of Windows Server, SQL Server 2016, and all versions of Windows 10, which looks like this:



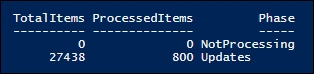
In step 4, you specify that your WSUS server should get updates for the products in the $CHP array. There is no output from this step.

For any given product supported, Windows Update can provide a number of different kinds, classifications, of updates. In step 6, you get the classifications of update types available, which looks like this:

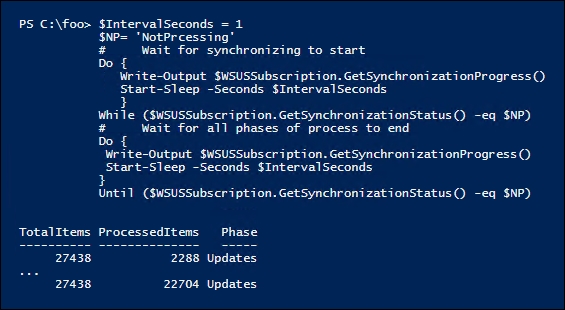


You may not want all these kinds of updates. To achieve this, in step 6, you build a list of the update classifications you do wish to support. In step 7, you configure your WSUS server with this list. In step 8, you obtain the synchronization status of WSUS1, and in step 9, you initiate synchronization of update categories of WSUS1 from Windows Update. These three steps produce no output.

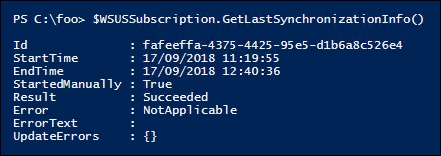
In step 10, you initiate a loop that gets the category synchronization status and, if it's still processing, wait a bit longer. This synchronization takes a long time—and looks like this (but with significant trimming!):



Next, in step 11, you now synchronize the updates available based on previous configuration, which, slightly trimmed to avoid pages of output, looks like this:



Once this synchronization is complete, in step 12, you can view the synchronization status, which now looks like this:



In step 13, you configure WSUS1 to download new updates every day, for those products and classifications you previously specified. This step produces no output.

## There's more…

In step 1, you examined the updates available for all versions of Windows Server. As you can see, this even includes very old versions of Windows Server, such as Windows Server 2003, which is now out of support and hopefully no longer being used in your organization. Inevitably, there are some organizations still running Windows Server 2003, hopefully for good business reasons. It's comforting to know that updates are still available even if the product should have been replaced years ago. You can also see that, as of the time of writing, Windows Update has no updates for Server 2019.

WSUS supports a range of products and different classifications of updates. Consider carefully what products you wish to get updates for and what update types to support. You could err on the side of caution, but that involves a lot of files and a very large number of updates you may never need.

# Configuring the Windows Update Client

By default, Windows computers, both the server and client version, download updates from Microsoft's Windows Update servers on the internet. In order to configure Windows hosts to take updates from an internal WSUS server, you need to update the configuration of the Windows Update Client that is built into Windows.

The easiest method of configuring the Windows Update Client is to use Group Policy. You create a Group Policy Object (GPO), configure the policy with server names, and so on, and then assign the policy.

You can apply a single GPO to the domain as a whole (configuring Windows Update Client on every domain-joined host) or apply policies at the site or OU level, depending on the complexity of your WSUS implementation. A small company located in a single site might apply just one policy at the domain level. Large multinational organizations may have multiple WSUS servers around the globe and might need multiple Windows Update policies applied throughout a large multi-forest network.

## Getting ready

You run this recipe from your client host, CL1, as configured by the Installing RSAT Tools on Windows 10 and Windows Server 2019 recipe.

## How to do it...

1. Create the WSUS server URL using the properties returned from the Get-WsusServer cmdlet:

$WSUSServer = Get-WsusServer -Name WSUS1.Reskit.Org -Port 8530

$FS = "http{2}://{0}:{1}"

$N = $WSUSServer.Name

$P = 8530 # default port

$WSUSServerURL = $FS -f $n, $p,

('','s')[$WSUSServer.UseSecureConnection]

$WSUSServerURL

1. Create a GPO and link it to the domain:

$PolicyName = 'Reskit WSUS Policy'

New-GPO -Name $PolicyName

New-GPLink -Name $PolicyName -Target 'DC=RESKIT,DC=Org'

1. Add registry key settings to the Group Policy to assign the WSUS server:

# Set computer to use WSUS not WU:

$KEY1 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate\AU'

$RVHT1 = @{

Name = $PolicyName

Key = $KEY1

ValueName = 'UseWUServer'

Type = 'DWORD'

Value = 1}

Set-GPRegistryValue @RVHT1 | Out-Null

# Set AU options:

$KEY2 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate\AU'

$RVHT2 = @{

Name = $PolicyName

Key = $KEY2

ValueName = 'AUOptions'

Type = 'DWORD'

Value = 2}

Set-GPRegistryValue @RVHT2 | Out-Null

# Set WSUS Server URL:

$KEY3 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate'

$RVHT3 = @{

Name = $PolicyName

Key = $KEY3

ValueName = 'WUServer'

Type = 'String'

Value = $WSUSServerURL}

Set-GPRegistryValue @RVHT3 | Out-Null

# Set WU Status server URL:

$KEY4 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate'

$RVHT4 = @{

Name = $PolicyName

Key = $KEY4

ValueName = 'WUStatusServer'

Type = 'String'

Value = $WSUSServerURL}

Set-GPRegistryValue @RVHT4 | Out-Null

1. Get a report on the GPO and view it:

$RHT = @{

Name = $PolicyName

ReportType = 'Html'

Path = 'C:\Foo\Out.htm'}

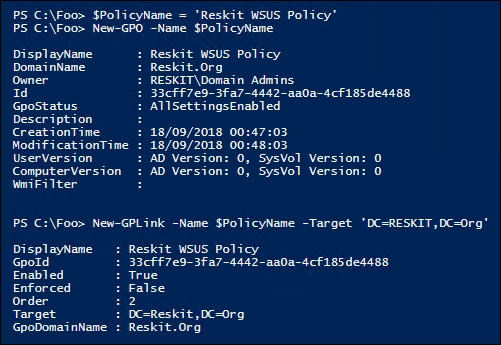
Get-GPOReport @RHT

Invoke-Item -Path $RHT.Path

## How it works...

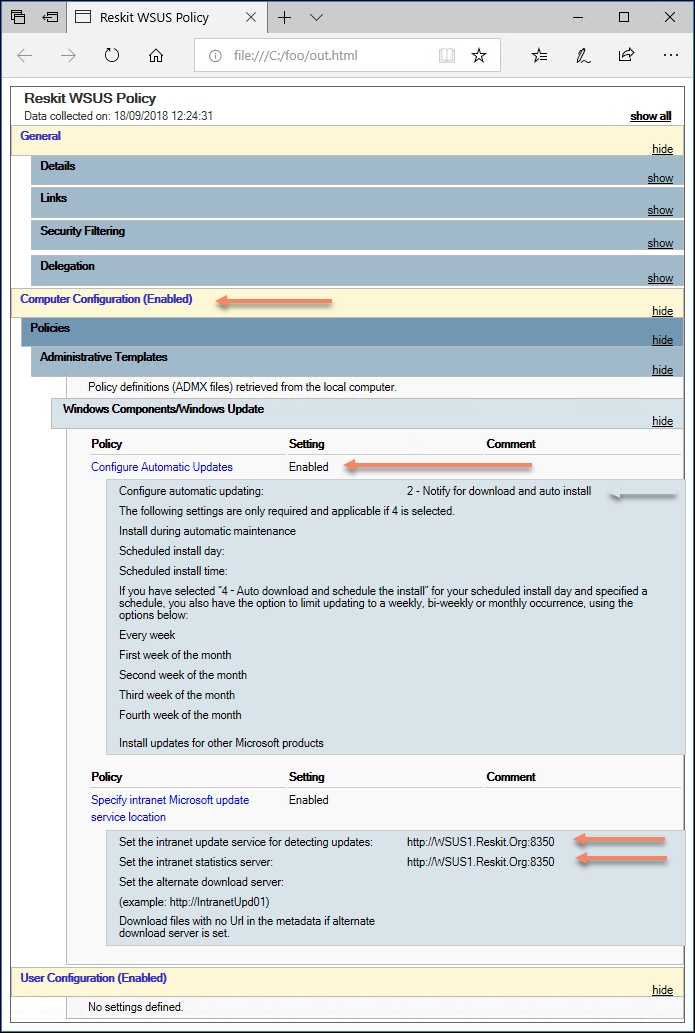
In step 1, you instantiate a WSUS server object that is used in later steps in the recipe. Then you use that object to create the URL that Windows Update Clients use to contact your WSUS server. There is no output from this step.

In step 2, you create a new GPO policy (Reskit WSUS Policy) and assign that policy to the Reskit.Org domain. This means that every domain-joined computer in the Reskit.Org domain is to get updates from WSUS1.Reskit.Org. This step produces output like this:



In step 4, you set values for the WSUS policy GPO. This configures the GPO with the necessary information to enable Windows Update to make use of WSUS in the organization. There is no output from this step.

In step 5, you view a GPO report of the WSUS policy GPO, which looks like this:



## There's more…

In step 2, you created the WSUS policy and linked it to the domain. For very large organizations, separate policies may be appropriate, each linked to separate OUs or sites in your AD. You may even wish, for very large organizations, multiple WSUS implementations around the world.

In step 3, you configured the GPO object with 4 registry-based settings. The recipe used Out-Null to limit the amount of output. If you experiment with this recipe, consider removing the pipe to null to see the output generated.

# Creating computer target groups

With the recipes so far in this chapter, you have set up a WSUS Server and created a GPO to configure the Windows Update Client on your computers. The next step is to create target groups—groups of computers you plan to use when targeting WSUS updates.

In any organization, different groups of hosts can have different update requirements. Your Windows client hosts run software such as Microsoft Office that is rarely seen on a server. Your mission critical servers might require a separate testing and sign-off process for updates that are approved for use.

For efficient management of updates, you define target groups (for example, domain controllers (DCs), SQL Servers, and so on) and then define the computers in the target group. In this recipe, you create a target group for DCs (that is, DC1, DC2).

## Getting ready

This recipe runs in the WSUS1 WSUS server. In the preceding recipes in this chapter, you have set up the WSUS server, configured WSUS updates, and configured the Windows Update Client computers via a GPO.

## How to do it...

1. Create a WSUS computer target group for the DCs:

$WSUSServer = Get-WsusServer -Name WSUS1 -port 8530

$WSUSServer.CreateComputerTargetGroup('Domain Controllers')

1. Examine existing target groups and view the new one:

$WSUSServer.GetComputerTargetGroups() |

Format-Table -Property Name

1. Find the DCs that have registered with WSUS1:

Get-WsusComputer -NameIncludes DC

1. Add DC1 and DC2 to the Domain Controllers target group:

Get-WsusComputer -NameIncludes DC |

Add-WsusComputer -TargetGroupName 'Domain Controllers'

1. Get the Domain Controllers target computer group:

$DCGroup = $WSUSServer.GetComputerTargetGroups() |

Where-Object Name -eq 'Domain Controllers'

1. Display the computers in the group:

Get-WsusComputer |

Where-Object ComputerTargetGroupIDs -Contains $DCGroup.id |

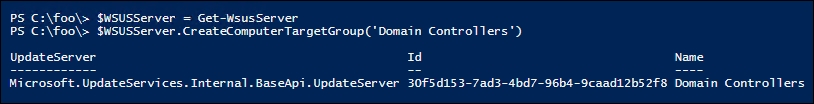
Sort-Object -Property FullDomainName |

Format-Table -Property FullDomainName, ClientVersion,

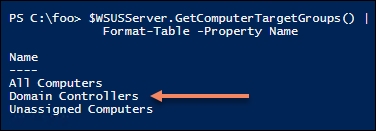
LastSyncTime

## How it works...

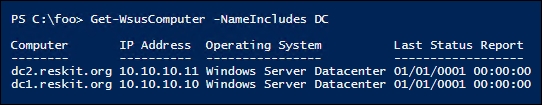
In step 1, you create a new computer target group called Domain Controllers, which looks like this:



In step 2, you use the $WSUSServer object to get and then display the current target groups, including the one you just created, which looks like this:

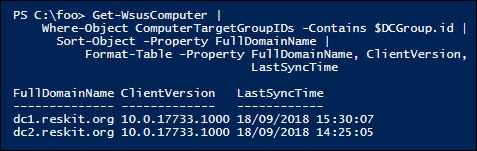


In step 3, you retrieve the computers whose name contains DC and that have registered with the WSUS server, which looks like this:



In step 4, which creates no output, you add the two DCs to the Domain Controllers computer target group. In step 5, which also creates no output, you get the target group from the WSUS computer.

Finally, in step 6, you examine the computers, their OS version, and the last synchronization time (with time in UTC), which looks like this:



## There's more…

In step 6, you display the computers in the Domain Controllers computer target group. Once you create the GPO object, it can take 24 hours or longer to have all the computers in your domain begin to work with WSUS for the computers in the Domain Controllers target group. Because of the time it can take to set up a WSUS server, it's a task possibly for a long weekend.

# Configuring WSUS automatic approvals

Microsoft's Windows Update can produce a large number of updates for you to manage (inspect, accept/decline, and deploy). Some update types, or example critical updates, may be ones you want to automatically approve, so as soon as you receive one of these, you can start deploying it.

## Getting ready

This recipe assumes you have a WSUS server, WSUS1, set up as per the previous recipes in this chapter. You can also use your own WSUS server, adapting this recipe as appropriate.

## How to do it...

1. Create the auto-approval rule:

$WSUSServer = Get-WsusServer

$ApprovalRule =

$WSUSServer.CreateInstallApprovalRule('Critical Updates')

1. Define a deadline for the rule:

$Type = 'Microsoft.UpdateServices.Administration.' +

'AutomaticUpdateApprovalDeadline'

$RuleDeadLine = New-Object -Typename $Type

$RuleDeadLine.DayOffset = 3

$RuleDeadLine.MinutesAfterMidnight = 180

$ApprovalRule.Deadline = $RuleDeadLine

1. Add update classifications to the rule:

$UC = $ApprovalRule.GetUpdateClassifications()

$C = $WSUSServer.GetUpdateClassifications() |

Where-Object -Property Title -eq 'Critical Updates'

$UC.Add($C)

$D = $WSUSServer.GetUpdateClassifications() |

Where-Object -Property Title -eq 'Definition Updates'

$UC.Add($D)

$ApprovalRule.SetUpdateClassifications($UpdateClassification)

1. Assign the rule to a computer target group:

$Type = 'Microsoft.UpdateServices.Administration.'+

'ComputerTargetGroupCollection'

$TargetGroups = New-Object $Type

$TargetGroups.Add(($WSUSServer.GetComputerTargetGroups() |

Where-Object -Property Name -eq "Domain Controllers"))

$ApprovalRule.SetComputerTargetGroups($TargetGroups)

1. Enable and save the rule:

$ApprovalRule.Enabled = $true

$ApprovalRule.Save()

1. Get a list of approval rules:

$WSUSServer.GetInstallApprovalRules() |

Format-Table -Property Name, Enabled, Action

## How it works...

In this recipe, you configured auto approval for some updates. This rule automatically approves updates that are either critical updates or definition updates. Any updates of these two types are automatically approved for use by clients.

In step 1, you create an in-memory object for an approval rule. Next, in step 2, you define a deadline for the rule. In step 3, you add some update classifications to the rule. Then, in step 4, you assign the rule to a computer target group. In step 5, you enable this new approval rule and save it. These five steps produce no output.

In step 6, you view the approval rules in place on your WSUS server, which looks like this:



## There's more…

This recipe uses the WSUS Server object and the many methods that the object provides. This is not dissimilar to older-style COM-object type programming. Other Windows features use cmdlets to perform management operations whereas here you are using object method calls. PowerShell's built-in help system does not provide much assistance for discovering details about the methods and how to use them. There is not much current up-to-date documentation on the methods and objects either.

# Managing WSUS updates

Each PowerShell module developer team, which includes the various feature teams inside the overall Windows Server product team, approach their problem domains slightly differently. Their product, their PowerShell module, has a certain usage style.

An important stylistic difference is the balance between cmdlets and object method calls. For some modules, you manage the service totally through cmdlets. The DNSServer and DHCPServer modules are examples of this.

The Windows Update module, on the other hand, makes use of method calls to perform the desired administrative task, such as approving or declining a specific update. Thus, many administrative functions are performed via method calls rather than cmdlets.

This recipe shows you how you can make use of the UpdateServer object and its rich collections of methods.

## Getting ready

This recipe runs on WSUS1, a WSUS server that you set up in the previous recipes in this chapter. You can certainly adapt this recipe to use your own local WSUS server.

## How to do it...

1. On WSUS1, open a session on the WSUS1 host and check overall status:

$WSUSServer = Get-WsusServer

$WSUSServer.GetStatus()

1. View the computer targets:

$WSUSServer.GetComputerTargets() |

Sort-Object -Property FullDomainName |

Format-Table -Property FullDomainName, IPAddress, Last\*

1. Search the WSUS server for updates with titles containing Windows Server 2016 that are classified as security updates, then use Get-Member, reviewing the properties and methods of the Microsoft.UpdateServices.Internal.BaseApi.Update object:

$ST = 'Windows Server 2016'

$SU = 'Security Updates'

$SecurityUpdates = $WSUSServer.SearchUpdates($ST) |

Where-Object UpdateClassificationTitle -eq $SU |

Sort-Object -Property CreationDate -Descending

1. View the first 10 security updates on WSUS1:

$SecurityUpdates |

Sort-Object -Property Title |

Select-Object -First 10 |

Format-Table -Property Title, Description

1. Select one of the updates to approve based on the KB article ID:

$SelectedUpdate = $SecurityUpdates |

Where-Object KnowledgebaseArticles -eq 3194798

1. Define the computer target group where you approve this update:

$DCTargetGroup = $WSUSServer.GetComputerTargetGroups() |

Where-Object -Property Name -eq 'Domain Controllers'

1. Approve the update for installation in the target group:

$SelectedUpdate.Approve('Install',$DCTargetGroup)

1. Select one of the updates to decline based on a KB article ID:

$DeclinedUpdate = $SecurityUpdates |

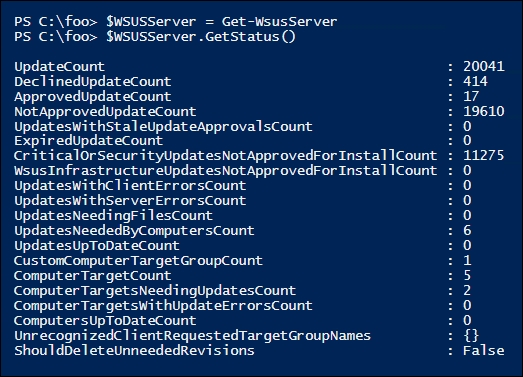
Where-Object -Property KnowledgebaseArticles -eq 4020821

1. Decline the update:

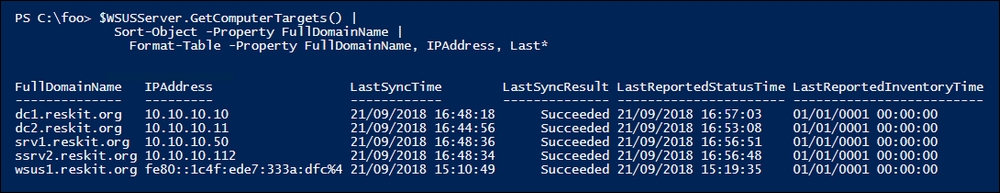
$DeclinedUpdate.Decline($DCTargetGroup)

## How it works...

In step 1, you use the Get-WsusServer cmdlet to return an UpdateServer object. This object and its methods are at the core of automating WSUS. You then use the GetStatus() method to return the status of your WSUS server, which looks like this:

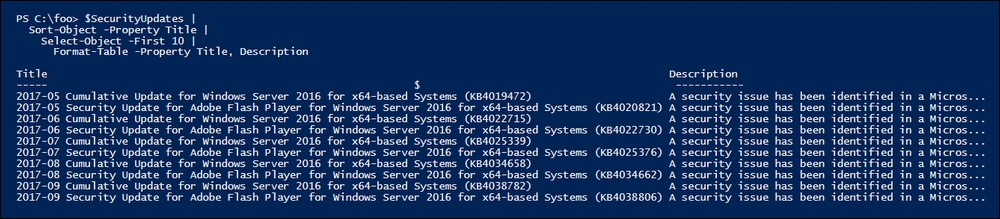


In step 2, you use the GetComputerTargets() method to get the host computers served by your WSUS server, which looks like this:



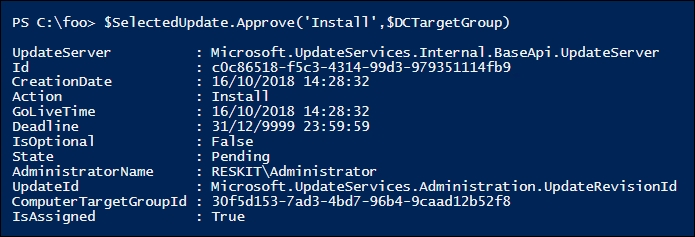
In step 3, you use the SearchUpdates() method to get the security updates for hosts running Windows Server 2016. This step produces no output.

In step 4, you review the first 10 security updates, which looks like this:



In step 5, which produces no output, you select a specific update, based on a KB article number. In step 6, you define a target group to which to apply the selected update. This step produces no output.

In step 7, you approve this selected patch for installation for all Domain Controllers computer target group. The output of this step looks like this:



In step 8, you select an update that you wish not to install. This step produces no output. In step 9, you decline the update for the Domain Controllers computer target group.

## There's more…

In step 3, you examined the security updates for Windows Server 2016. You could also have looked for any Updates or Critical Updates. You can also vary the value of the $ST parameter to search for different targets, such as Windows 10 or Office.

In step 5, you selected a specific update. If you are an IT Pro responsible for Windows Update Services inside your organization, you need to keep up to date on critical updates so you can deploy urgent patches as quickly as possible.

In step 9, you declined a specific update for one computer target group. As you administer WSUS, you are likely to discover certain updates that can be declined since they do not impact certain target groups. Keeping on top of which patches to approve or decline can be a lot of work, but is vital to ensure that your systems are updated promptly.