14

Managing Windows Update Services

This chapter covers 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 client and server systems updated 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, `WSUS` enables you to manage patches and updates for various Microsoft software products. Thus, an update you download from Microsoft and distribute via `WSUS` may apply to Windows, Microsoft Office, and many other Microsoft software products.

This chapter shows how to install and configure the `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.

Note: Windows Update is one of the few Windows services you can not administer using `PowerShell 7` directly. The design of the `WSUS` module means you can not use the cmdlets directly using `PowerShell 7` or the `Windows PowerShell` compatibility mechanism. `WSUS` cmdlets use `.NET Framework` classes that the `.NET` team has not migrated to work with the open-source `.NET` implementation used by `PowerShell 7`. Additionally, the `WSUS` team designed the cmdlets, so you use object instance methods rather than `cmdlets`. Thus, if you load the module using the compatibility mechanism, the methods are lost, and you cannot use them directly within `PowerShell 7`. You can, however, create a remoting session using a Windows PowerShell endpoint and run these otherwise incompatible `cmdlets` within that remoting session.

This chapter demonstrates that you can manage `WSUS` using `PowerShell 7` and Windows PowerShell remoting. This method is a lot more work, and it is more work to debug scripts – but it does work.

The systems used in the chapter

This chapter uses two primary hosts: `SRV1` and `WSUS1`. You run recipes remotely from `SRV1` against the `WSUS` host. You also need the domain's domain controller, `DC1`, as well. The hosts used in this chapter are as shown here:

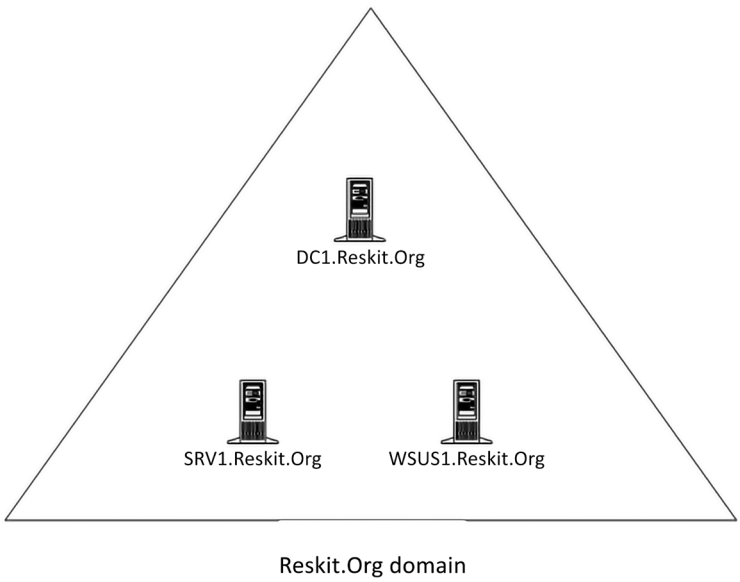

Figure 14.1: Hosts in use for this chapter

Installing Windows Server Update Services

WSUS is an installable feature within Windows Server 2022 that Windows setup does not install by default. To use Windows Update Services, you first install the WSUS Windows feature and do basic configuration and setup. Once installed, you can use the Windows PowerShell cmdlets to deploy WSUS in your environment.

In this recipe, you install WSUS and review the results of that installation. In subsequent recipes, you manage and use the WSUS service.

Note that this recipe uses remoting from SRV1 to WSUS1. The steps demonstrate the complexity of installing WSUS and configuring it for initial use in general and within PowerShell 7.

Getting ready

You run this recipe on SRV1, a domain-joined Windows Server 2022 host. The recipe also uses the WSUS1 server, another 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. Creating a remoting session for WSUS1:

```
$SessionHT = @{
  ConfigurationName = 'microsoft.powershell'
  ComputerName      = 'WSUS1'
  Name              = 'WSUS'
}
$Session = New-PSSession @SessionHT
```

2. Installing WSUS on WSUS1:

```
$ScriptBlock1 = {
  $InstallHT = @{
    Name                   = 'UpdateServices'
    IncludeManagementTools = $true
  }
  Install-WindowsFeature @InstallHT |
    Format-Table -AutoSize -Wrap
}
Invoke-Command -Session $Session -ScriptBlock $ScriptBlock1
```

3. Determining features installed on `WSUS1`:

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WindowsFeature |
    Where-Object Installed |
      Format-Table
}
```

4. Creating a folder for `WSUS` update content on `WSUS1`:

```
$ScriptBlock2 = {
  $WSUSDir = 'C:\WSUS'
  If (-Not (Test-Path -Path $WSUSDir -ErrorAction SilentlyContinue))
      {New-Item -Path $WSUSDir -ItemType Directory | Out-Null}
}
Invoke-Command -Session $Session -ScriptBlock $ScriptBlock2
```

5. Performing the post-installation configuration using `wsusutil.exe`:

```
$ScriptBlock3 = {
  $WSUSDir = 'C:\WSUS'
  $Child = 'Update Services\Tools\wsusutil.exe'
  $CMD = Join-Path -Path "$env:ProgramFiles\" -ChildPath $Child
  & $CMD Postinstall CONTENT_DIR="$WSUSDir"
}
Invoke-Command -ComputerName WSUS1 -ScriptBlock $ScriptBlock3
```

6. Viewing the `WSUS` website on `WSUS1`:

```
Invoke-Command -ComputerName WSUS1 -ScriptBlock {
  Get-Website -Name ws* | Format-Table -AutoSize
}
```

7. View the cmdlets in the `UpdateServices` module:

```
Invoke-Command -ComputerName WSUS1 -ScriptBlock {
  Get-Command -Module UpdateServices |
    Format-Table -AutoSize
}
```

8. Inspecting properties of the object created with `Get-WsusServer`:

```
Invoke-Command -Session $Session -ScriptBlock {
```

```
  $WSUSServer = Get-WsusServer
  $WSUSServer.GetType().Fullname
  $WSUSServer | Select-Object -Property *
}
```

9. Viewing details of the WSUS server object:

```
Invoke-Command -Session $Session -ScriptBlock {
  ($WSUSServer | Get-Member -MemberType Method).count
  $WSUSServer | Get-Member -MemberType Method
}
```

10. Viewing WSUS server configuration:

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer.GetConfiguration() |
    Select-Object -Property SyncFromMicrosoftUpdate, LogFilePath
}
```

11. Viewing product categories after initial install:

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer
  "{0} WSUS Products discovered" -f $WSUSProducts.Count
  $WSUSProducts |
    Select-Object -ExpandProperty Product |
     Format-Table -Property Title,
                            Description
}
```

12. Displaying subscription information:

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSSubscription = $WSUSServer.GetSubscription()
  $WSUSSubscription |
    Select-Object -Property * |
      Format-List
}
```

13. Getting latest categories of products available from Microsoft Update:

```
Invoke-Command -Session $Session -ScriptBlock {
```

```
  $WSUSSubscription.StartSynchronization()
  Do {
    Write-Output $WSUSSubscription.GetSynchronizationProgress()
    Start-Sleep -Seconds 30
  }
  While ($WSUSSubscription.GetSynchronizationStatus() -ne
                                          'NotProcessing')
}
```

14. Checking the results of the synchronization:

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSSubscription.GetLastSynchronizationInfo()
}
```

15. Reviewing the categories of the products available after synchronization:

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer
  "{0} Product found on WSUS1" -f $WSUSProducts.Count
  $WSUSProducts |
    Select-Object -ExpandProperty Product -First 25 |
      Format-Table -Property Title,
                             Description
}
```

How it works...

In *step 1*, you create a remoting session from `SRV1` to `WUS1`. This step produces no console output. Next, in *step 2*, you install the Windows Update Services feature and the associated tools via the remoting session. The output looks like this:

```
PS C:\Foo> # 2. Installing WSUS on WSUS1
PS C:\Foo> $ScriptBlock1 = {
             $InstallHT = @{
               Name                   = 'UpdateServices'
               IncludeManagementTools = $true
             }
             Install-WindowsFeature @InstallHT |
               Format-Table
           }
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock $ScriptBlock1

Success Restart Needed Exit Code Feature Result
------- -------------- --------- --------------
True    No             Success   {ASP.NET 4.8, HTTP Activation,
                                  Remote Server Administration Tools,
                                  Role Administration Tools...}

WARNING: Additional configuration may be required. Review the article Managing WSUS Using
PowerShell at TechNet Library (http://go.microsoft.com/fwlink/?LinkId=235499) for more
information on the recommended steps to perform WSUS installation using PowerShell.
```

Figure 14.2: Installing WSUS on WSUS1 via remoting

When you install the Windows Update Services, the installation process adds several additional related services, as you can see in *step 3*, which looks like this:

```
PS C:\Foo> # 3. Determining features installed on WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
               Get-WindowsFeature |
                 Where-Object Installed |
                   Format-Table
           }

Display Name                                            Name                       Install Sta
------------                                            ----                       -----------
[X] File and Storage Services                           FileAndStorage-Services    Installed
    [X] Storage Services                                Storage-Services           Installed
[X] Web Server (IIS)                                    Web-Server                 Installed
    [X] Web Server                                      Web-WebServer              Installed
        [X] Common HTTP Features                        Web-Common-Http            Installed
            [X] Default Document                        Web-Default-Doc            Installed
            [X] Static Content                          Web-Static-Content         Installed
        [X] Performance                                 Web-Performance            Installed
            [X] Dynamic Content Compression             Web-Dyn-Compression        Installed
        [X] Security                                    Web-Security               Installed
            [X] Request Filtering                       Web-Filtering              Installed
            [X] Windows Authentication                  Web-Windows-Auth           Installed
        [X] Application Development                     Web-App-Dev                Installed
            [X] .NET Extensibility 4.8                  Web-Net-Ext45              Installed
            [X] ASP.NET 4.8                             Web-Asp-Net45              Installed
            [X] ISAPI Extensions                        Web-ISAPI-Ext              Installed
            [X] ISAPI Filters                           Web-ISAPI-Filter           Installed
    [X] Management Tools                                Web-Mgmt-Tools             Installed
        [X] IIS Management Console                      Web-Mgmt-Console           Installed
        [X] IIS 6 Management Compatibility              Web-Mgmt-Compat            Installed
            [X] IIS 6 Metabase Compatibility            Web-Metabase               Installed
[X] Windows Server Update Services                      UpdateServices             Installed
    [X] WID Connectivity                                UpdateServices-WidDB       Installed
    [X] WSUS Services                                   UpdateServices-Services    Installed
[X] .NET Framework 4.8 Features                         NET-Framework-45-Featu…    Installed
    [X] .NET Framework 4.8                              NET-Framework-45-Core      Installed
    [X] ASP.NET 4.8                                     NET-Framework-45-ASPNET    Installed
    [X] WCF Services                                    NET-WCF-Services45         Installed
        [X] HTTP Activation                             NET-WCF-HTTP-Activatio…    Installed
        [X] TCP Port Sharing                            NET-WCF-TCP-PortSharin…    Installed
[X] Microsoft Defender Antivirus                        Windows-Defender           Installed
[X] Remote Server Administration Tools                  RSAT                       Installed
    [X] Role Administration Tools                       RSAT-Role-Tools            Installed
        [X] Windows Server Update Services Tools        UpdateServices-RSAT        Installed
            [X] API and PowerShell cmdlets              UpdateServices-API         Installed
            [X] User Interface Management Console       UpdateServices-UI          Installed
[X] System Data Archiver                                System-DataArchiver        Installed
[X] Windows Internal Database                           Windows-Internal-Datab…    Installed
[X] Windows PowerShell                                  PowerShellRoot             Installed
    [X] Windows PowerShell 5.1                          PowerShell                 Installed
[X] Windows Process Activation Service                  WAS                        Installed
    [X] Process Model                                   WAS-Process-Model          Installed
    [X] Configuration APIs                              WAS-Config-APIs            Installed
[X] WoW64 Support                                       WoW64-Support              Installed
[X] XPS Viewer                                          XPS-Viewer                 Installed
```

Figure 14.3: Viewing the installed features on `WSUS1`

In *step 4*, you create a folder that you use to hold `WSUS` content on `WSUS1`. This step produces no console output.

In *step 5*, you perform the post-installation task using the `wsusutil.exe` console command, which produces some limited output like this:

```
PS C:\Foo> # 5. Performing post-installation configuration using WsusUtil.exe
PS C:\Foo> $ScriptBlock3 = {
             $WSUSDir = 'C:\WSUS'
             $Child = 'Update Services\Tools\wsusutil.exe'
             $CMD = Join-Path -Path "$env:ProgramFiles\" -ChildPath $Child
             & $CMD ——% Postinstall CONTENT_DIR=$WSUSDir
           }
PS C:\Foo> Invoke-Command -ComputerName WSUS1 -ScriptBlock $ScriptBlock3
Log file is located at C:\Users\Administrator\ApoData\Local\Temp\\WSUS Postinstall_20221202T165613.10g
Post install is starting
Post install has successfully completed
```

Figure 14.4: Viewing Installed features on `WSUS1`

When you execute *step 5*, the `wsusutil.exe` utility creates an `IIS` website on `WSUS1` to communicate with `WSUS` clients. In *step 6*, you view the site, as you can see here:

```
PS C:\Foo> # 6. Viewing the WSUS website on WSUS1
PS C:\Foo> Invoke—Command —ComputerName WSUS1 —ScriptBlock {
             Get—Website —Name ws* | Format—Table —AutoSize
           }

Name                ID        State   Physical Path                                        Bindings
----                --        -----   -------------                                        --------
WSUS Administration 624931287 Started C:\Program Files\Update Services\WebServices\Root\ http :8530:
                                                                                           https :8531: sslFlags=0
```

Figure 14.5: Viewing the WSUS website

In *step 7*, you examine the commands contained in the `UpdateServices` module you installed earlier (in *step 1*). The output of this step looks like this:

```
PS C:\Foo> # 7. Viewing the cmdlets in the UpdateServices module
PS C:\Foo> Invoke-Command -ComputerName WSUS1 -ScriptBlock {
             Get-Command -Module UpdateServices |
               Format-Table -AutoSize
           }

CommandType Name                               Version Source
----------- ----                               ------- ------
Cmdlet      Add-WsusComputer                   2.0.0.0 UpdateServices
Cmdlet      Add-WsusDynamicCategory            2.0.0.0 UpdateServices
Cmdlet      Approve-WsusUpdate                 2.0.0.0 UpdateServices
Cmdlet      Deny-WsusUpdate                    2.0.0.0 UpdateServices
Cmdlet      Get-WsusClassification             2.0.0.0 UpdateServices
Cmdlet      Get-WsusComputer                   2.0.0.0 UpdateServices
Cmdlet      Get-WsusDynamicCategory            2.0.0.0 UpdateServices
Cmdlet      Get-WsusProduct                    2.0.0.0 UpdateServices
Cmdlet      Get-WsusServer                     2.0.0.0 UpdateServices
Cmdlet      Get-WsusUpdate                     2.0.0.0 UpdateServices
Cmdlet      Invoke-WsusServerCleanup           2.0.0.0 UpdateServices
Cmdlet      Remove-WsusDynamicCategory         2.0.0.0 UpdateServices
Cmdlet      Set-WsusClassification             2.0.0.0 UpdateServices
Cmdlet      Set-WsusDynamicCategory            2.0.0.0 UpdateServices
Cmdlet      Set-WsusProduct                    2.0.0.0 UpdateServices
Cmdlet      Set-WsusServerSynchronization      2.0.0.0 UpdateServices
```

Figure 14.6: Viewing the WSUS cmdlets

You examine the key properties of your `WSUS` server, in *step 8*, by using the `Get-WsusServer` cmdlet. The cmdlet returns a `UpdateServer` object, which looks like this:

```
PS C:\Foo> # 8. Inspecting properties of the object created with Get-WsusServer
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer = Get-WsusServer
             $WSUSServer.GetType().Fullname
             $WSUSServer | Select-Object -Property *
           }

   TypeName: Microsoft.UpdateServices.Internal.BaseApi.UpdateServer

WebServiceUrl                   : http://WSUS1:8530/ApiRemoting30/WebService.asmx
BypassApiRemoting               : False
IsServerLocal                   : True
Name                            : WSUS1
Version                         : 10.0.20348.143
IsConnectionSecureForApiRemoting : True
PortNumber                      : 8530
PreferredCulture                : en
ServerName                      : WSUS1
UseSecureConnection             : False
ServerProtocolVersion           : 1.20
PSComputerName                  : WSUS1
RunspaceId                      : 80977ba5-ba38-4d97-95d5-a5057610dbb1
```

Figure 14.7: Viewing the WSUS server properties

The `$WSUSServer` object you instantiated in *step 8* also contains many methods you can call to manage aspects of the `WSUS` server. There are a large number of methods, as you can see from the output of *step 9*:

```
PS C:\Foo> # 9. Viewing details of the WSUS Server object
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             ($WSUSServer | Get-Member -MemberType Method).count
             $WSUSServer | Get-Member -MemberType Method
           }

   TypeName: Microsoft.UpdateServices.Internal.BaseApi.UpdateServer

Name                                MemberType Definition
----                                ---------- ----------

AddDynamicCategories                Method     void AddDynamicCategories(Sys...
AddDynamicCategory                  Method     void AddDynamicCategory(Micro...
CancelAllDownloads                  Method     void CancelAllDownloads(), vo...
CreateComputerTargetGroup           Method     Microsoft.UpdateServices.Admi...
CreateDynamicCategory               Method     Microsoft.UpdateServices.Admi...
CreateInstallApprovalRule           Method     Microsoft.UpdateServices.Admi...
CreateObjRef                        Method     System.Runtime.Remoting.ObjRe...
DeleteDynamicCategory               Method     void DeleteDynamicCategory(st...
DeleteUpdate                        Method     void DeleteUpdate(guid update...
Equals                              Method     bool Equals(System.Object obj...
ExpirePackage                       Method     void ExpirePackage(Microsoft....
ExportPackageMetadata               Method     void ExportPackageMetadata(Mi...
ExportUpdates                       Method     void ExportUpdates(string pac...
GetChildServers                     Method     Microsoft.UpdateServices.Admi...
GetCleanupManager                   Method     Microsoft.UpdateServices.Admi...
GetComponentsWithErrors             Method     System.Collections.Specialize...
GetComputersNotContactedSinceCount  Method     int GetComputersNotContactedS...
GetComputersWithRecentNameChange    Method     System.Collections.Specialize...
GetComputerTarget                   Method     Microsoft.UpdateServices.Admi...
GetComputerTargetByName             Method     Microsoft.UpdateServices.Admi...
GetComputerTargetCount              Method     int GetComputerTargetCount(),...
GetComputerTargetGroup              Method     Microsoft.UpdateServices.Admi...
GetComputerTargets                  Method     Microsoft.UpdateServices.Admi...
GetContentDownloadProgress          Method     Microsoft.UpdateServices.Admi...
GetCurrentUserRole                  Method     Microsoft.UpdateServices.Admi...
GetDatabaseConfiguration            Method     Microsoft.UpdateServices.Admi...
GetDownstreamServer                 Method     Microsoft.UpdateServices.Admi...
GetDownstreamServers                Method     Microsoft.UpdateServices.Admi...
GetDynamicCategories                Method     System.Collections.Generic.IE...
GetDynamicCategory                  Method     Microsoft.UpdateServices.Admi...
GetEmailNotificationConfiguration   Method     Microsoft.UpdateServices.Admi...
GetFailedToDownloadUpdatesCount     Method     int GetFailedToDownloadUpdate...
GetFrontEndServers                  Method     System.Collections.ObjectMode...
GetHashCode                         Method     int GetHashCode()
```

Figure 14.8: Viewing the WSUS server object's methods

A key troubleshooting feature of `WSUS` is the `SoftwareDistribution.log`. In *step 10*, you view the `WSUS` configuration to discover the filename, which looks like this:

```
PS C:\Foo> # 10. Viewing WSUS server configuration
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer.GetConfiguration() |
               Select-Object -Property SyncFromMicrosoftUpdate,LogFilePath
           }

SyncFromMicrosoftUpdate : True
LogFilePath             : C:\Program Files\Update Services\LogFiles\SoftwareDistribution.log
PSComputerName          : WSUS1
RunspaceId              : 262cd894-b585-4b58-b7c4-6bdedebfb635
```

Figure 14.9: Viewing the WSUS configuration

Following the initial installation and configuration in *step 11*, you can see that the `WSUS1` server gets updates for a very small set of products (17 in all), as shown here:

```
PS C:\Foo> # 11. Viewing product categories after initial install
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer
             "{0} WSUS Products discovered" -f $WSUSProducts.Count
             $WSUSProducts |
               Select-Object -ExpandProperty Product |
                Format-Table -Property Title,
                                       Description
           }

17 WSUS Products discovered

Title                                  Description
-----                                  -----------
Exchange 2000 Server                   For Exchange 2000 Products
Exchange Server 2003                   For Exchange 2003 Products
Exchange                               Exchange
Local Publisher                        The local publisher (not Microsoft Update) of patches and applications.
Locally published packages             Patches and applications that are published locally, not synchronized from Microsoft Update.
Microsoft Corporation                  Microsoft Corporation
Office 2003                            Office 2003
Office XP                              Office XP
Office                                 Office
SQL Server                             SQL Server Catergory Description
SQL                                    SQL
Windows 2000 family                    Windows 2000 family
Windows Server 2003 family             Windows Server 2003 family
Windows Server 2003, Datacenter Edition Windows Server 2003, Datacenter Edition
Windows XP 64-Bit Edition Version 2003 Windows XP 64-Bit Edition Version 2003
Windows XP family                      Windows XP family
Windows                                Windows
```

Figure 14.10: Viewing the WSUS configuration

You can configure the `WSUS` server to subscribe to and automatically retrieve new updates. In *step 12*, you retrieve and view the `WSUS` server's subscription details, which look like this:

```
PS C:\Foo> # 12. Displaying subscription information
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSSubscription = $WSUSServer.GetSubscription()
             $WSUSSubscription |
               Select-Object -Property * |
                 Format-List
           }

UpdateServer                      : Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
SynchronizeAutomatically          : False
SynchronizeAutomaticallyTimeOfDay : 09:15:52
LastModifiedTime                  : 03/12/2022 11:31:41
LastModifiedBy                    : RESKIT\Administrator
LastSynchronizationTime           : 01/01/0001 00:00:00
Anchor                            : 0,2000-01-01 00:00:01.000
DeploymentAnchor                  :
NumberOfSynchronizationsPerDay    : 1
IsCategoryOnlySync                : False
```

Figure 14.11: Viewing the WSUS subscription information

In *step 13*, you perform a full synchronization by invoking the `StartSynchronization()` method of the `WSUS` server object. This method invokes an asynchronous operation – after calling this method, WSUS carries out the server update process in the background. You can call the `GetSynchronizationStatus()` 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:

```
PS C:\Foo> # 13. Getting latest categories of products available from Microsoft Update
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSSubscription.StartSynchronization()
             Do {
               Write-Output $WSUSSubscription.GetSynchronizationProgress()
               Start-Sleep -Seconds 30
             }
             While ($WSUSSubscription.GetSynchronizationStatus() -ne
                                                  'NotProcessing')
           }

PSComputerName : WSUS1
RunspaceId     : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems     : 0
ProcessedItems : 0
Phase          : NotProcessing

PSComputerName : WSUS1
RunspaceId     : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems     : 3954
ProcessedItems : 0
Phase          : Categories

... snipped for brevity

PSComputerName : WSUS1
RunspaceId     : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems     : 103642
ProcessedItems : 5451
Phase          : Updates

PSComputerName : WSUS1
RunspaceId     : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems     : 103642
ProcessedItems : 104202
Phase          : Updates
```

Figure 14.12: Synchronizing WSUS

After WSUS has completed the synchronization, in *step 14*, you review a summary of the results, showing the successful result. The output looks like this:

```
PS C:\Foo> # 14. Checking the results of the synchronization
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSSubscription.GetLastSynchronizationInfo()
           }

PSComputerName   : WSUS1
RunspaceId       : 77ec8cff-d1c6-4c90-a11d-4cef89201968
Id               : f2db6a4e-da3a-41b5-9330-2630cfb7d724
StartTime        : 04/12/2022 11:36:04
EndTime          : 04/12/2022 11:37:33
StartedManually  : True
Result           : Succeeded
Error            : NotApplicable
ErrorText        :
UpdateErrors     : {}
```

Figure 14.13: Reviewing results of the most recent synchronization

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

```
PS C:\Foo> # 15.Reviewing the categories of the products available after synchronzation
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer
             "{0} Product found on WSUS1" -f $WSUSProducts.Count
             $WSUSProducts |
               Select-Object -ExpandProperty Product -First 25 |
                 Format-Table -Property Title,
                                        Description
           }

391 Product found on WSUS1

Title                                                 Description
-----                                                 -----------
.NET 5.0                                              .NET 5.0
.NET 6.0                                              .NET 6.0
.NET Core 2.1                                         .NET Core 2.1
.NET Core 3.1                                         .NET Core 3.1
Active Directory Rights Management Services Client 2.0 Active Directory Rights Management Services Client 2.0 (AD...
Active Directory                                      Active Directory Product Family Category
Antigen for Exchange/SMTP                             Defines the category for Antigen Updates. This will make s...
Antigen                                               Antigen Product Family Category
ASP.NET Web and Data Frameworks                       ASP.NET Web and Data Frameworks product family
ASP.NET Web Frameworks                                ASP.NET Web Framework
Azure Connected Machine Agent 3                       Updates for the Azure Connected Machine Agent
Azure Connected Machine Agent                         Product Family for Azure Connected Machine Agent
Azure File Sync agent updates for Windows Server 2012 R2 Azure File Sync agent updates for Windows Server 2012 R2
Azure File Sync agent updates for Windows Server 2016 Azure File Sync agent updates for Windows Server 2016
Azure File Sync agent updates for Windows Server 2019 Azure File Sync agent updates for Windows Server 2019
Azure File Sync agent updates for Windows Server 2022 Azure File Sync agent updates for Windows Server 2022
Azure File Sync                                       Azure File Sync
Azure IoT Edge for Linux on Windows Category          Updates for Azure IoT Edge for Linux on Windows Category. ...
Azure IoT Edge for Linux on Windows                   Product Family for Azure IoT Edge for Linux on Windows
Azure Stack HCI                                       Azure Stack HCI and above
Bing Bar                                              Get quick access to Bing and MSN, as well as handy tools f...
Bing Growth                                           Product Family for Bing Growth
Bing Service v2.0                                     Bing Service 2.0 update
Bing                                                  Live Search Product Family Category
BizTalk Server 2002                                   Category for BizTalk 2002. It requires SP1 as the minimum ...
```

Figure 14.14: Reviewing product categories now available to WSUS

There's more...

In *step 2*, you use the `Install-WindowsFeature` command to install `WSUS`. As you can see in the output, you must perform additional configuration before using `WSUS`, which you do in later recipe steps, particularly in *step 5*. Also, note that the URL shown in the output is invalid at the time of writing. Microsoft has removed the mentioned TechNet Library article; sadly, the associated content also appears lost.

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

In *step 5*, you run the `wsusutil.exe` command remotely on `WSUS1` to complete the service installation. This executable comes with the `WSUS` installation. This command is useful and valuable because the `WSUS` team did not support certain operations via Windows PowerShell cmdlets. In this step, you run the command to perform the initial configuration of your WSUS server, including creating the `WSUS` database. If the command does not complete successfully, recovery is not particularly straightforward. If you use a `VM` to run `WSUS1` (both in your testing and production scenario), taking `VM` snapshots after each step makes it much easier to recover.

In *step 13*, you perform a full sync with the Windows Update servers. The initial synchronization 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, use web proxies, and work with `SQL Server` instead of the Windows Internal Database.

The objects created by `WSUS` are complex. In most cases, you can use standard PowerShell discovery techniques to discover more about these objects. There is more detailed information about the `WSUS` objects at `https://learn.microsoft.com/previous-versions/windows/desktop/mt748187(v=vs.85)`. Microsoft last updated this documentation in 2016 and does not update it regularly. That said, the information is accurate for `WSUS` in Windows Server 2022.

Configuring WSUS Update Synchronization

As you can see, WSUS can update hundreds of products, although many may not be useful in your organization. After you install WSUS and do the initial synchronization, you can configure WSUS to identify the specific products for which your organization requires product updates. You can also define the classifications of updates WSUS should download.

Once you define the updates to be obtained (and later provided to WSUS clients), you can configure WSUS to synchronize subsequent updates manually. You can also build an update schedule and have WSUS update automatically. In this way, you can have WSUS download only the updates for the product categories and the classifications you have selected at your chosen time. The first initial synchronization can take hours, depending on your selections. Subsequent synchronizations pull only the newest updates since the last synchronization.

Getting ready

You run this recipe on SRV1, a domain-joined Windows Server 2022 host. The recipe also uses the WSUS1 server, another 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. Creating a remote session on WSUS1

```
$Session = New-PSSession -ComputerName WSUS1
```

2. Locating versions of Windows Server supported by Windows Update

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusProduct |
    Where-Object -FilterScript {$_.product.title -match
                                '^Windows Server'} |
      Select-Object -ExpandProperty Product  |
        Format-Table Title, UpdateSource
}
```

3. Discovering updates for Windows 11

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusProduct -TitleIncludes 'Windows 11' |
    Select-Object -ExpandProperty Product |
      Format-Table -Property Title
}
```

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

```
Invoke-Command -Session $Session -ScriptBlock {
$Products =
   (Get-WsusProduct |
     Where-Object -FilterScript {$_.product.title -match
                                 '^Windows Server'}).Product.Title
  $Products += @('Microsoft SQL Server 2016','Windows 11')
  $Products
}
}
```

5. Assigning the desired products to include in Windows Update

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusProduct |
    Where-Object {$PSItem.Product.Title -in $Products} |
        Set-WsusProduct
}
```

6. Getting WSUS classification

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusClassification |
    Select-Object -ExpandProperty Classification |
      Format-Table -Property Title, Description  -Wrap
}
```

7. Building a list of desired update classifications

```
Invoke-Command -Session $Session -ScriptBlock {
  $UpdateList = @('Critical Updates',
                  'Definition Updates',
                  'Security Updates',
                  'Service Packs',
                  'Update Rollups',
                  'Updates')
}
```

8. Setting the list of desired update classifications in WSUS

```
Invoke-Command -Session $Session -ScriptBlock {
```

```
  Get-WsusClassification |
    Where-Object {$_.Classification.Title -in $UpdateList} |
      Set-WsusClassification
}
```

9. Getting synchronization details

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer = Get-WsusServer
  $WSUSSubscription = $WSUSServer.GetSubscription()
}
```

10. Starting synchronizing available updates

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSSubscription.StartSynchronization()
}
```

11. Looping and waiting for synchronization to complete

```
Invoke-Command -Session $Session -ScriptBlock {
  $IntervalSeconds = 15
  $NP = 'NotProcessing'
  Do {
    $WSUSSubscription.GetSynchronizationProgress()
    Start-Sleep -Seconds $IntervalSeconds
    } While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)
}
```

12. Synchronizing the updates which can take a long while to complete

```
Invoke-Command -Session $Session -ScriptBlock {
  $IntervalSeconds = 15
  $NP = 'NotProessing'
  #   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)
}
```

13. Checking the results of the synchronization

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSSubscription.GetLastSynchronizationInfo()
}
```

14. Configure automatic synchronization to run once per day

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSSubscription = $WSUSServer.GetSubscription()
  $WSUSSubscription.SynchronizeAutomatically = $true
  $WSUSSubscription.NumberOfSynchronizationsPerDay = 1
  $WSUSSubscription.Save()
}
```

How it works...

In *step 1*, you create a remoting session from `SRV1` to `WUS1`. This step produces no console output. Next, in *step 2*, you determine the different versions of Windows Server supported by `WSUS`, with output that looks like this:

```
PS C:\Foo> # 2. Locating versions of Windows Server supported by Windows Update
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             Get-WsusProduct |
               Where-Object -FilterScript {$_.product.title -match
                                           '^Windows Server'} |
                 Select-Object -ExpandProperty Product  |
                   Format-Table Title, UpdateSource
           }

Title                                                                  UpdateSource
-----                                                                  ------------
Windows Server 2003, Datacenter Edition                                MicrosoftUpdate
Windows Server 2003                                                    MicrosoftUpdate
Windows Server 2008 R2                                                 MicrosoftUpdate
Windows Server 2008 Server Manager Dynamic Installer                   MicrosoftUpdate
Windows Server 2008                                                    MicrosoftUpdate
Windows Server 2012 Language Packs                                     MicrosoftUpdate
Windows Server 2012 R2  and later drivers                              MicrosoftUpdate
Windows Server 2012 R2 Drivers                                         MicrosoftUpdate
Windows Server 2012 R2 Language Packs                                  MicrosoftUpdate
Windows Server 2012 R2                                                 MicrosoftUpdate
Windows Server 2012                                                    MicrosoftUpdate
Windows Server 2016 and Later Servicing Drivers                        MicrosoftUpdate
Windows Server 2016 for RS4                                            MicrosoftUpdate
Windows Server 2016                                                    MicrosoftUpdate
Windows Server 2016                                                    MicrosoftUpdate
Windows Server 2019 and later, Servicing Drivers                       MicrosoftUpdate
Windows Server 2019 and later, Upgrade & Servicing Drivers             MicrosoftUpdate
Windows Server 2019 Datacenter: Azure Edition Hotpatch                 MicrosoftUpdate
Windows Server 2019                                                    MicrosoftUpdate
Windows Server 2019                                                    MicrosoftUpdate
Windows Server Drivers                                                 MicrosoftUpdate
Windows Server Manager - Windows Server Update Services (WSUS) Dynamic Installer MicrosoftUpdate
Windows Server Solutions Best Practices Analyzer 1.0                   MicrosoftUpdate
Windows Server Technical Preview Language Packs                        MicrosoftUpdate
Windows Server, version 1903 and later                                 MicrosoftUpdate
Windows Server, version 1903 and later                                 MicrosoftUpdate
```

Figure 14.15: Reviewing product categories now available to WSUS

In *step 3*, you view the version(s) of Windows 11 that you can update using `WSUS` and Windows Update, like this:

```
PS C:\Foo> # 3. Discovering updates for for Windows 11
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             Get-WsusProduct -TitleIncludes 'Windows 11' |
               Select-Object -ExpandProperty Product |
                 Format-Table -Property Title
           }

Title
-----
Windows 11 Client S, version 22H2 and later, Servicing Drivers
Windows 11 Client S, version 22H2 and later, Upgrade & Servicing Drivers
Windows 11 Client, version 22H2 and later, Servicing Drivers
Windows 11 Client, version 22H2 and later, Upgrade & Servicing Drivers
Windows 11 Dynamic Update
Windows 11 GDR-DU
Windows 11
```

Figure 14.16: Versions of Windows 11 supported by WSUS

In most cases, you probably do not want or need to support all Microsoft products. Rather, you most likely want to get updates for a subset of products that exist in your environment. To achieve that, you begin, in *step 4*, 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 11, which looks like this:

```
PS C:\Foo> # 4. Create and view a list of software product titles to include
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $Products =
              (Get-WsusProduct |
                Where-Object -FilterScript {$_.product.title -match
                                            '^Windows Server'}).Product.Title
             $Products += @('Microsoft SQL Server 2016','Windows 11')
             $Products
           }
Windows Server 2003, Datacenter Edition
Windows Server 2003
Windows Server 2008 R2
Windows Server 2008 Server Manager Dynamic Installer
Windows Server 2008
Windows Server 2012 Language Packs
Windows Server 2012 R2  and later drivers
Windows Server 2012 R2 Drivers
Windows Server 2012 R2 Language Packs
Windows Server 2012 R2
Windows Server 2012
Windows Server 2016 and Later Servicing Drivers
Windows Server 2016 for RS4
Windows Server 2016
Windows Server 2016
Windows Server 2019 and later, Servicing Drivers
Windows Server 2019 and later, Upgrade & Servicing Drivers
Windows Server 2019 Datacenter: Azure Edition Hotpatch
Windows Server 2019
Windows Server 2019
Windows Server Drivers
Windows Server Manager - Windows Server Update Services (WSUS) Dynamic Installer
Windows Server Solutions Best Practices Analyzer 1.0
Windows Server Technical Preview Language Packs
Windows Server, version 1903 and later
Windows Server, version 1903 and later
Microsoft SQL Server 2016
Windows 11
```

Figure 14.17: Creating a list of products for WSUS to support

In *step 5*, you specify that your `WSUS` server should get updates for the products in the `$Products` array you created in the previous step. There is no output from this step.

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

```
PS C:\Foo> # 6. Getting WSUS classification
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             Get-WsusClassification |
               Select-Object -ExpandProperty Classification |
                 Format-Table -Property Title, Description  -Wrap
           }

Title              Description
-----              -----------
Applications       Products or line of business applications.
Critical Updates   A broadly released fix for a specific problem addressing a critical,
                   non-security related bug.
Definition Updates A broadly-released and frequent software update containing additions to a
                   product's definition database. Definition databases are often used to
                   detect objects with specific attributes, such as malicious code, phishing
                   Web sites, or junk e-mail.
Driver Sets        A package of software modules that is designed to support the hardware of
                   a specific model of computing device.
Drivers            A software component necessary to control or regulate another device.
Feature Packs      New product functionality that is first distributed outside the context
                   of a product release, and usually included in the next full product
                   release.
Security Updates   A broadly released fix for a product-specific security-related
                   vulnerability. Security vulnerabilities are rated based on their severity
                   which is indicated in the Microsoft® security bulletin as critical,
                   important, moderate, or low.
Service Packs      A tested, cumulative set of all hotfixes, security updates, critical
                   updates and updates, as well as additional fixes for problems found
                   internally since the release of the product. Service packs may also
                   contain a limited number of customer-requested design changes or features.
Tools              A utility or feature that aids in accomplishing a task or set of tasks.
Update Rollups     A tested, cumulative set of hotfixes, security updates, critical updates,
                   and updates packaged together for easy deployment. A rollup generally
                   targets a specific area, such as security, or a component of a product,
                   such as Internet Information Services "IIS".
Updates            A broadly released fix for a specific problem addressing a noncritical,
                   non-security-related bug.
Upgrades           A new product release bringing a device to the next version, containing
                   bug fixes, design changes and new features.
```

Figure 14.18: Reviewing WSUS update classifications

You may not want all these kinds of updates. For example, you may wish not to download driver updates, perhaps preferring manual updates for drivers if/when needed. To achieve this, in *step 7*, you build a list of the update classifications you wish `WSUS` to support. In *step 8*, you configure your `WSUS` server with this list. In *step 9*, you obtain the synchronization status of `WSUS1`, and in *step 10*, you initiate synchronization of update categories of `WSUS1` from Windows Update. Then in *step 11*, you wait for the synchronization to complete. These four steps produce no console output.

In *step 12*, you initiate a loop that gets the category synchronization status. If category updating is still processing, wait a bit longer. This synchronization should not take too long, but be patient. The console output looks like this:

```
PS C:\Foo> # 12. Synchronizing the updates which can take a long while to complete
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $IntervalSeconds = 15
             $NP = 'NotProcessing'
             Do {
               $WSUSSubscription.GetSynchronizationProgress()
               Start-Sleep -Seconds $IntervalSeconds
               } While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)
           }

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-alld-4cef89201968
TotalItems     : 31327
ProcessedItems : 420
Phase          : Updates

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-alld-4cef89201968
TotalItems     : 31327
ProcessedItems : 730
Phase          : Updates
PS C:\Foo>
```

Figure 14.19: Synchronizing update categories

With the update categories synchronized, you can now synchronize the updates available to your WSUS server – updates for specified properties with a specific update category. You do this in *step 13*. This step can take a long time. The (trimmed) output from this step looks like this:

```
PS C:\Foo> # 13. Synchronize the updates which can take a long while to compelete.
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $IntervalSeconds = 15
             $NP = 'NotProessing'
             #   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)
           }

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 3576
Phase          : Updates

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 24204
Phase          : Updates
PS C:\Foo>
```

Figure 14.20: Synchronizing updates

Once this synchronization is complete, in *step 14*, you view the `WSUS` synchronization status, which now looks like this:

```
PS C:\Foo> # 14. Checking the results of the synchronization
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSSubscription.GetLastSynchronizationInfo()
           }

PSComputerName  : WSUS1
RunspaceId      : 77ec8cff-d1f6-4c90-alld-4cef89201968
Id              : 10aee80a-2746-49af-b78e-27fdl2c6ec09
StartTime       : 04/12/2022 12:12:28
EndTime         : 04/12/2022 14:38:41
StartedManually : True
Result          : Succeeded
Error           : NotApplicable
ErrorText       :
UpdateErrors    : {}
```

Figure 14.21: Viewing synchronization status

In *step 14*, 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 2*, 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, you no longer use within your organization.

Inevitably, some organizations are still running Windows Server 2003, hopefully for good business reasons. It's comforting to know that you can still get updates even if you should have updated or replaced them years ago. That said, it would be highly unusual for Microsoft to issue further updates for such old and out-of-support versions of Windows.

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 can involve many updates you may never need.

In *step 13*, you see the synchronization status every 15 seconds. At each check, you can see how many updates have been downloaded. The initial update downloads can take a long time.

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. To configure Windows hosts to take updates from an internal `WSUS` server, you need to update the configuration of the built-in Windows Update Client in Windows.

Using Group Policy is the easiest method of configuring the Windows Update Client. 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 on a single site might use just one policy at the domain or site level. Large multinational organizations may have multiple `WSUS` servers around the globe and need multiple Windows Update policies applied throughout a large multi-forest network.

In this recipe, you configure SRV1 to get updates from the WSUS server WSUS1.

Getting ready

You run this recipe on SRV1 after you have installed and configured WSUS on the WSUS1 server.

How to do it...

1. Ensuring the GP management tools are available on SRV1

```
Install-WindowsFeature -Name GPMC -IncludeManagementTools | Out-Null
```

2. Creating a new policy and linking it to the domain

```
$PolicyName = 'Reskit WSUS Policy'
New-GPO -Name $PolicyName
```

3. Configuring SRV1 to use WSUS for updates

```
$WSUSKEY = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate\
AU'
$RVHT1 = @{
  Name       = $PolicyName
  Key        = $WSUSKEY
  ValueName  = 'UseWUServer'
  Type       = 'DWORD'
  Value      = 1
}
Set-GPRegistryValue @RVHT1 | Out-Null
```

4. Setting 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
```

5. Setting the WSUS server URL

```
$Session = New-PSSession -ComputerName WSUS1

$WSUSServer = Invoke-Command -Session $Session -ScriptBlock {
  Get-WSUSServer
}
$FS =  "http{2}://{0}:{1}"
$N  = $WSUSServer.Name
$P  = 8530 # default WSUS port
$WSUSURL = $FS -f $n, $p, ('','s')[$WSUSServer.UseSecureConnection]
$KEY3 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate'
$RVHT3 = @{
Name      = $PolicyName
Key       = $KEY3
ValueName = 'WUServer'
Type      = 'String'
Value     = $WSUSURL
}
Set-GPRegistryValue @RVHT3  | Out-Null
```

6. Setting the WU status server URL

```
$KEY4 = 'HKLM\Software\Policies\Microsoft\Windows\WindowsUpdate'
$RVHT4 = @{
Name       = $PolicyName
Key        = $KEY4
ValueName  = 'WUStatusServer'
Type       = 'String'
Value      = $WSUSURL
}
Set-GPRegistryValue @RVHT4 | Out-Null
```

7. Viewing a report on the GPO

```
$RHT = @{
Name       = $PolicyName
ReportType = 'Html'
Path       = 'C:\foo\out.html'
}
Get-GPOReport @RHT
Invoke-Item -Path $RHT.Path
```

How it works...

In *step 1*, you install the Group Policy Management Console and related tools (in case you have not previously installed the feature on this host in previous recipes). This step creates no console output.

In *step 2*, you create a new GPO in the `Reskit.Org` domain, generating output like this:

```
PS C:\Foo> # 2. Creating a new policy and linking it to the domain
PS C:\Foo> $PolicyName = 'Reskit WSUS Policy'
PS C:\Foo> New-GPO -Name $PolicyName

DisplayName      : Reskit WSUS Policy
DomainName       : Reskit.Org
Owner            : RESKIT\Domain Admins
Id               : 9c6e0c02-75ef-43c7-8148-366e23da186e
GpoStatus        : AllSettingsEnabled
Description      :
CreationTime     : 08/12/2022 17:08:43
ModificationTime : 08/12/2022 17:08:43
UserVersion      :
ComputerVersion  :
WmiFilter        :
```

Figure 14.22: Creating a new GPO

In *step 3*, you configure `SRV1` to accept updates from `WSUS` (versus Microsoft Update). In *step 4*, you set the options for Windows Update to download and install any approved updates. In *step 5*, you create and configure Windows Update with the URL to use to get `WSUS`-approved updates. In *step 6*, you specify the status server (the server to which Windows Update sends status reports on updates). These four steps create no output.

In the final step, *step 7*, you generate and view a management report showing the policy details. There is no console output, but this step opens a browser window with the report like this:

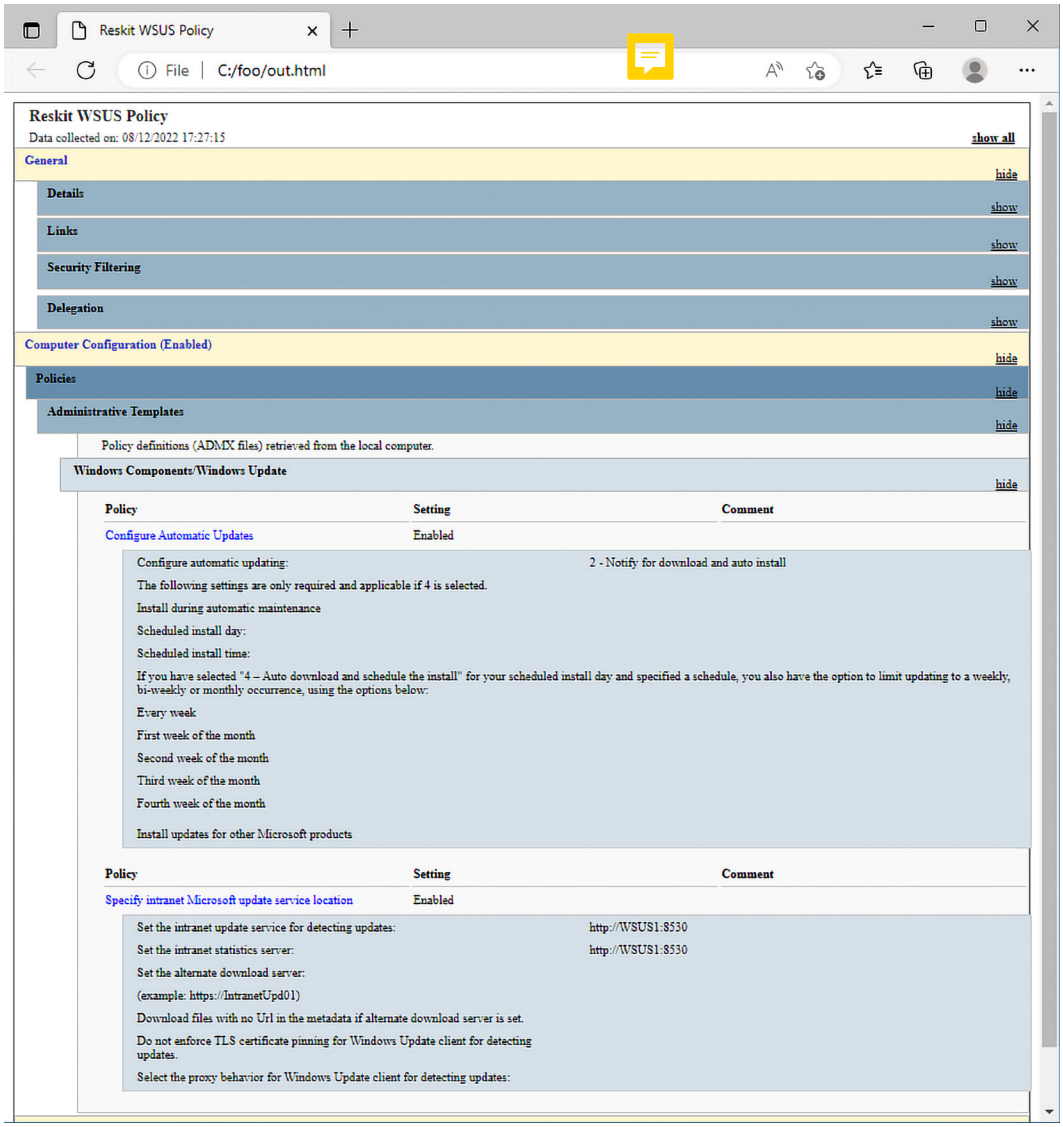

Figure 14.23: Viewing the GPO

There's more...

In *step 2*, you created the WSUS policy and linked it to the domain. For large organizations, separate policies may be appropriate, each connected to separate OUs or sites in your AD. This can facilitate distributed administration where different AD OUs or AD sites might need different settings, and for very large organizations, multiple independent WSUS implementations worldwide.

In this recipe, you configured the GPO object with four 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.

In *step 7*, you view the GPO report. This report shows what settings are included in the GPO.

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—the computers you plan to use when targeting WSUS updates.

In any organization, different groups of hosts can have other update requirements. Your Windows client hosts run software such as Microsoft Office that you do not normally see on a server.

Your mission-critical servers might require a separate testing and sign-off process for updates that you then approve for use. For efficient management of updates, you define target groups (for example, **domain controllers** (**DCs**), SQL servers, and so on) and then determine the computers in the target group.

In this recipe, you create a target group for domain servers, including SRV1 and SRV2.

Getting ready

You run this recipe on SRV1, with SRV1 and WSU1 online. You should have installed and configured WSUS on WSUS1. Additionally, you need at least one of the domain controllers in the Reskit.org domain up and running.

How to do it...

1. Creating a remoting session to WSUS1

```
$SessionHT = @{
    ConfigurationName = 'microsoft.powershell'
    ComputerName      = 'WSUS1'
```

```
    Name              = 'WSUS'
  }
$Session = New-PSSession @SessionHT
```

2. Creating a `WSUS` computer target group for servers

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer = Get-WsusServer  -Name WSUS1 -port 8530
  $WSUSServer.CreateComputerTargetGroup('Domain Servers')
}
```

3. Viewing all computer target groups on `WSUS1`

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer.GetComputerTargetGroups() |
    Format-Table -Property Name
}
```

4. Finding the servers whose name includes `SRV`

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusComputer -NameIncludes SRV |
    Format-Table -Property FullDomainName, OSDescription
}
```

5. Adding `SRV1`, `SRV2` to the `Domain Servers` target group

```
Invoke-Command -Session $Session -ScriptBlock {
  Get-WsusComputer -NameIncludes SRV |
    Where-Object FullDomainName -match '^SRV' |
      Add-WsusComputer -TargetGroupName 'Domain Servers'
}
```

6. Getting the `Domain Servers` computer target group

```
Invoke-Command -Session $Session -ScriptBlock {
  $SRVGroup = $WSUSServer.GetComputerTargetGroups() |
                Where-Object Name -eq 'Domain Servers'
}
```

7. Finding the computers in the group:

```
Invoke-Command -Session $Session -ScriptBlock {
```

```
  Get-WsusComputer |
    Where-Object ComputerTargetGroupIDs -Contains $SRvGroup.id |
      Sort-Object -Property FullDomainName |
          Format-Table -Property FullDomainName, ClientVersion,
                                 LastSyncTime
}
```

How it works...

In *step 1*, you create a remoting session to `WSUS1`, using a Windows PowerShell 5.1 endpoint. This step creates no console output.

In *step 2*, you create a new computer target group called `Domain Servers`, which looks like this:

```
PS C:\Foo> # 2.  Creating a WSUS computer target group for servers
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer = Get-WsusServer  -Name WSUS1 -port 8530
             $WSUSServer.CreateComputerTargetGroup('Domain Servers')
           }

PSComputerName    WSUS1
RunspaceId        c174bdae-e697-4fb4-8be2-3c9da9701920
UpdateServer      Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
Id                3a409534-50c7-4159-af22-caea674d5ff1
Name              Domain Servers
```

Figure 14.24: Creating a computer target group

In *step 3*, you use the `$WSUSServer` object to get and then display the current target groups, including the one you just created, which looks like this:

```
PS C:\Foo> # 3. Viewing all computer target groups on WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer.GetComputerTargetGroups() |
               Format-Table -Property Name
           }

Name
----
All Computers
Domain Servers
Unassigned Computers
```

Figure 14.25: Viewing computer target groups on `WSUS1`

In *step 4*, you retrieve the computers whose name contains SRV and that have registered with the WSUS server, which looks like this:

```
PS C:\Foo> # 4. Finding the Servers whose name includes SRV
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             Get-WsusComputer -NameIncludes SRV |
               Format-Table -Property FultDomainName, OSDescription
           }

FultDomainName  OSDescription
--------------  -------------
srv1.reskit.org Windows Server 2022 Datacenter
srv2.reskit.org Windows Server 2022 Datacenter
psrv.reskit.org Windows Server 2022 Datacenter
```

Figure 14.26: Viewing computer target groups on WSUS1

In *step 5*, which creates no output, you add just the two servers (SRV1 and SRV2) into the Domain Servers computer target group. In *step 6*, which also creates no console output, you instantiate the Domain Servers target group and store it in the SRVGroup variable.

In the final step, you view the servers in the Domain Servers target group with output like this:

```
PS C:\Foo> # 7. Finding the computers in the group:
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             Get-WsusComputer |
               Where-Object ComputerTargetGroupIDs -Contains $SRvGroup.id |
                 Sort-Object -Property FullDomainName |
                     Format-Table -Property FultDomainName, ClientVersion,
                                            LastSyncTime
           }

FullDomainName  ClientVersion   LastSyncTime
--------------  -------------   ------------
srv1.reskit.org 10.0.20348.1070 09/12/2022 10:03:01
srv1.reskit.org 10.0.20348.1070 09/12/2022 08:49:28
```

Figure 14.27: Viewing members of the Domain Servers computer target group

There's more...

In *step 2*, you create a target group. In WSUS, a target group is a collection of computer systems that an administrator defines and uses to specify the computers that should receive updates from the WSUS server. You can use the target groups that organize computers based on various criteria, such as their location, role, or hardware configuration. This allows you to configure which specific updates are needed by each target group.

In *step 7*, you display the computers in the Domain Servers computer target group. Once you create the GPO object (as in *step 2*), it can take 24 hours or longer to have all the computers in your domain begin working with WSUS for the computers in the Domain Servers target group. Since it can take a lot of time to set up a WSUS server and create and populate computer target groups, it may be a task you leave for a long weekend.

Configuring WSUS Automatic Approvals

Microsoft's Windows Update can produce many updates for you to manage (`inspect`, `accept/decline`, and `deploy`). Some update types, for 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.

Configuring automatic approvals can be a good thing in that you ask `WSUS` to push more urgent updates automatically. At the same time, automatically pushing an update can be problematic if, for some reason, the update has issues.

Getting ready

You run this recipe on `SRV1` after installing and configuring `WSUS` on `WSUS1`.

How to do it...

1. Creating a remoting session to `WSUS1`

```
$SessionHT = @{
        ConfigurationName = 'microsoft.powershell'
        ComputerName      = 'WSUS1'
        Name              = 'WSUS'
      }
$Session = New-PSSession @SessionHT
```

2. Creating the auto-approval rule

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer = Get-WsusServer
  $ApprovalRule =
    $WSUSServer.CreateInstallApprovalRule('Critical Updates')
}
```

3. Defining a deadline for the rule

```
Invoke-Command -Session $Session -ScriptBlock {
  $Type = 'Microsoft.UpdateServices.Administration.' +
          'AutomaticUpdateApprovalDeadline'
  $RuleDeadLine = New-Object -Typename $Type
  $RuleDeadLine.DayOffset = 3
  $RuleDeadLine.MinutesAfterMidnight = 180
  $ApprovalRule.Deadline = $RuleDeadLine
}
```

4. Adding update classifications to the rule

```
Invoke-Command -Session $Session -ScriptBlock {
  $UpdateClassifications = $ApprovalRule.GetUpdateClassifications()
  $CriticalUpdates = $WSUSServer.GetUpdateClassifications() |
    Where-Object -Property Title -eq 'Critical Updates'
  $UpdateClassifications.Add($CriticalUpdates) | Out-Null
  $Defs = $WSUSServer.GetUpdateClassifications() |
            Where-Object -Property Title -eq 'Definition Updates'
  $UpdateClassifications.Add($Defs) | Out-Null
  $ApprovalRule.SetUpdateClassifications($UpdateClassifications)
}
```

5. Assigning the rule to a computer target group

```
Invoke-Command -Session $Session -ScriptBlock {
  $Type = 'Microsoft.UpdateServices.Administration.'+
          'ComputerTargetGroupCollection'
  $TargetGroups = New-Object $Type
  $TargetGroups.Add(($WSUSServer.GetComputerTargetGroups() |
    Where-Object -Property Name -eq 'Domain Servers'))
  $ApprovalRule.SetComputerTargetGroups($TargetGroups) |
    Out-Null
}
```

6. Enabling the rule

```
Invoke-Command -Session $Session -ScriptBlock {
  $ApprovalRule.Enabled = $true
  $ApprovalRule.Save()
}
```

7. Getting a list of approval rules

```
Invoke-Command -Session $Session -ScriptBlock{
  $WSUSServer.GetInstallApprovalRules()   |
    Format-Table -Property Name, Enabled, Action
}
```

How it works...

In this recipe, you configure automatic approval for certain updates. This rule automatically approves updates that are either critical updates or definition updates. Updates of these two types you approve for use by clients.

In *step 1*, you create a PowerShell remoting session on the `WSUS1` server using a Windows PowerShell remoting endpoint. In *step 2*, you create an in-memory object for an approval rule. Next, in *step 3*, you define a deadline for the rule. In *step 4*, you add some update classifications to the rule. Then, in *step 5*, you assign the rule to a computer target group. In *step 5*, you enable this new approval rule and save it. These six steps produce no output.

In *step 7*, you get a list of the current approval rules on `WSUS1`, with output like this:

```
PS C:\Foo> # 7. Geting a list of approval rules
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock{
             $WSUSServer.GetInstallApprovalRules()   |
               Format-Table -Property Name, Enabled, Action
           }

Name                              Enabled  Action
----                              -------  ------
Default Automatic Approval Rule     False Install
Critical Updates                     True Install
```

Figure 14.28: Viewing a list of approval rules

There's more...

For many operations, the `WSUS` team did not provide cmdlets and also did not provide `GUI` support. Instead, you need to rely on the properties and particularly the methods of many `WSUS` objects. A key object is the `WSUS` server object. This object, which you instantiate using the `Get-WSUSServer cmdlet`, contains methods such as `GetInstallApprovalRules()`, which gets the installed approval rules. The `WSUS` team chose not to create a `Get-InstallApprovalRule cmdlet`.

How you write scripts to manage `WSUS` is similar to how you might have developed older-style Windows applications using the **Component Object Model (COM)**. PowerShell's built-in help system does not provide much assistance in discovering details about the methods or how to use them. There is little current, up-to-date documentation on the methods and objects either. This approach makes it harder to access all of the richness of `WSUS` simply through cmdlets.

Managing WSUS Updates

Microsoft produces a large number of updates and has ever since Microsoft first introduced `WSUS`. You can manage these updates via the `GUI` or PowerShell. As with other aspects of managing `WSUS`, managing updates via PowerShell means using the `WSUS` server object and its associated methods.

Getting ready

You run this recipe on `SRV1` after installing and configuring `WSUS` on `WSUS1`.

How to do it...

1. Creating a remoting session to `WSUS1`

```
$SessionHT = @{
  ConfigurationName = 'microsoft.powershell'
  ComputerName      = 'WSUS1'
  Name              = 'WSUS'
}
$Session = New-PSSession @SessionHT
```

2. Viewing the status of `WSUS1`

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer = Get-WsusServer
  $WSUSServer.GetStatus()
}
```

3. Viewing computer targets

```
Invoke-Command -Session $Session -ScriptBlock {
  $WSUSServer.GetComputerTargets() |
    Sort-Object -Property FullDomainName |
      Format-Table -Property FullDomainName, IPAddress, Last*
}
```

4. Searching for updates with titles containing `Windows Server 2022`

```
Invoke-Command -Session $Session -ScriptBlock {
  $Title   = 'Windows Server 2022'
  $Updates = 'Security Updates'
  $SecurityUpdates = $WSUSServer.SearchUpdates($Title)
}
```

5. Viewing the matching updates (first 10)

```
Invoke-Command -Session $Session -ScriptBlock {
  $SecurityUpdates |
    Sort-Object -Property Title |
      Select-Object -First 10 |
        Format-Table -Property Title, Description
}
```

6. Selecting one of the updates to approve based on the **Knowledge Based** (**KB**) article ID

```
Invoke-Command -Session $Session -ScriptBlock {
  $SelectedUpdate = $SecurityUpdates |
    Where-Object KnowledgebaseArticles -eq 5019080
}
```

7. Defining the computer target group

```
Invoke-Command -Session $Session -ScriptBlock {
  $SRVTargetGroup = $WSUSServer.GetComputerTargetGroups() |
    Where-Object -Property Name -eq 'Domain Servers'
}
```

8. Approving the update for installation in the target group

```
Invoke-Command -Session $Session -ScriptBlock {
  $SelectedUpdate.Approve('Install',$SRVTargetGroup)
}
```

9. Selecting one of the updates to decline based on a KB article ID

```
Invoke-Command -Session $Session -ScriptBlock {
$DeclinedUpdate = $SecurityUpdates |
  Where-Object -Property KnowledgebaseArticles -eq 5019080
}
```

10. Declining the update

```
Invoke-Command -Session $Session -ScriptBlock {
  $DeclinedUpdate.Decline($DCTargetGroup)
}
```

How it works...

In *step 1*, you create a PowerShell remoting session on the `WSUS1` server using a Windows PowerShell remoting endpoint. In *step 2*, you use the `Get-WsusServer cmdlet` to instantiate a `UpdateServer` object inside the persistent remoting session to `WSUS1`. 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:

```
PS C:\Foo> # 2. Viewing the status of WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer = Get-WsusServer
             $WSUSServer.GetStatus()
           }

PSComputerName                                 : WSUS1
RunspaceId                                     : aaeea23c-8aff-4460-abc8-5580828dfb01
UpdateCount                                    : 26471
DeclinedUpdateCount                            : 539
ApprovedUpdateCount                            : 18
NotApprovedUpdateCount                         : 25914
UpdatesWithStaleUpdateApprovalsCount           : 0
ExpiredUpdateCount                             : 0
CriticalOrSecurityUpdatesNotApprovedForInstallCount : 15474
WsusInfrastructureUpdatesNotApprovedForInstallCount : 0
UpdatesWithClientErrorsCount                   : 0
UpdatesWithServerErrorsCount                   : 0
UpdatesNeedingFilesCount                       : 0
UpdatesNeededByComputersCount                  : 46
UpdatesUpToDateCount                           : 25886
CustomComputerTargetGroupCount                 : 1
ComputerTargetCount                            : 13
ComputerTargetsNeedingUpdatesCount             : 12
ComputerTargetsWithUpdateErrorsCount           : 0
ComputersUpToDateCount                         : 0
UnrecognizedClientRequestedTargetGroupNames    : {}
ShouldDeleteUnneededRevisions                  : False
```

Figure 14.29: Viewing status of `WSUS1`

In *step 3*, you use the `GetComputerTargets()` method to retrieve the names of the host computers served by your `WSUS` server, which looks like this:

```
PS C:\Foo> # 3. Viewing computer targets
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $WSUSServer.GetComputerTargets() |
               Sort-Object -Property FullDomainName |
                 Format-Table -Property FullDomainName, IPAddress, Last*
           }

FullDomainName   IPAddress                   LastSyncTime        LastSyncResult LastReportedStatusTime LastReportedInventoryTime
--------------   ---------                   ------------        -------------- ---------------------- -------------------------
ch1.reskit.org   10.10.10.221                11/12/2022 13:58:36      Succeeded 11/12/2022 14:06:54    01/01/0001 00:00:00
dc1.reskit.org   10.10.10.10                 11/12/2022 15:56:15      Succeeded 11/12/2022 16:04:22    01/01/0001 00:00:00
dc2.reskit.org   10.10.10.11                 11/12/2022 14:28:41      Succeeded 11/12/2022 14:36:46    01/01/0001 00:00:00
fs1.reskit.org   10.10.10.101                11/12/2022 13:58:50      Succeeded 11/12/2022 14:06:59    01/01/0001 00:00:00
fs2.reskit.org   10.10.10.102                11/12/2022 13:58:26      Succeeded 11/12/2022 14:06:36    01/01/0001 00:00:00
hv1.reskit.org   10.10.10.201                11/12/2022 13:58:49      Succeeded 11/12/2022 14:06:59    01/01/0001 00:00:00
hv2.reskit.org   10.10.10.202                11/12/2022 13:58:51      Succeeded 11/12/2022 14:07:00    01/01/0001 00:00:00
psrv.reskit.org  10.10.10.60                 11/12/2022 14:59:07      Succeeded 11/12/2022 15:07:19    01/01/0001 00:00:00
smtp.reskit.org  10.10.10.49                 11/12/2022 10:54:59      Succeeded 11/12/2022 10:57:01    01/01/0001 00:00:00
srv1.reskit.org  10.10.10.50                 11/12/2022 16:23:24      Succeeded 11/12/2022 13:06:36    01/01/0001 00:00:00
srv2.reskit.org  10.10.10.52                 11/12/2022 11:52:24      Succeeded 11/12/2022 12:00:32    01/01/0001 00:00:00
ss1.reskit.org   10.10.10.111                11/12/2022 13:58:51      Succeeded 11/12/2022 14:06:59    01/01/0001 00:00:00
wsus1.reskit.org fe80::8672:8cc3:1e91:da0f%3 11/12/2022 06:00:52      Succeeded 11/12/2022 06:09:01    01/01/0001 00:00:00
```

Figure 14.30: Viewing WSUS computers

In *step 4*, you use the `SearchUpdates()` method to get the security updates for hosts running Windows Server 2022. This step produces no output. In *step 5*, you review the first 10 security updates for Windows Server 2022, which looks like this:

```
Title                                                                                                                                      Description
-----                                                                                                                                      -----------
2021-11 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5007386) Install this u[illegible] resol…
2021-12 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5008286) Install this update to resol…
2022-02 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5010456) Install this update to resol…
2022-03 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5011580) Install this update to resol…
2022-06 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5014677) Install this update to resol…
2022-09 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5017392) Install this update to resol…
2022-11 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5019080) Install this update to resol…
```

Figure 14.31: Viewing available updates

In *step 6*, you select a specific update based on a KB article number. In *step 7*, you define a target group to which to apply the selected update. These two steps produce no output.

In *step 8*, you approve this selected patch for installation for all members of the `Domain Servers` computer target groups. The output of this step looks like this:

```
PS C:\Foo> # 8. Approving the update for installation in the target group
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
             $SelectedUpdate.Approve('Install',$SRVTargetGroup)
           }

PSComputerName         : WSUS1
RunspaceId             : aaeea23c-8aff-4460-abc8-5580828dfb01
UpdateServer           : Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
Id                     : 678c6d2f-f575-4ade-95b1-9521942a8df6
CreationDate           : 11/12/2022 19:26:26
Action                 : Install
GoLiveTime             : 11/12/2022 19:26:26
Deadline               : 31/12/9999 23:59:59
IsOptional             : False
State                  : Pending
AdministratorName      : RESKIT\Administrator
UpdateId               : Microsoft.UpdateServices.Administration.UpdateRevisionId
ComputerTargetGroupId  : 3a409534-50c7-4159-af22-caea674d5ff1
IsAssigned             : True
```

Figure 14.32: Approving an update explicitly

In *step 9*, select an update you don't wish to install. This step produces no output. In *step 10*, you decline that update for the `Domain Servers` computer target group, which also creates no console output.

There's more...

In *step 4*, you examined the security updates for Windows Server 2022. You could also have looked for any updates or critical updates. You can also update this step to search for different targets, such as Windows 10 or Office.

In *step 6*, you selected a specific update based on a KB article ID that you want to approve explicitly. Suppose you are an IT pro responsible for Windows Update Services inside your organization. In that case, you must keep up to date on critical updates and deploy urgent patches as quickly as possible.

In *step 9*, you declined a specific update for one computer target group, also based on the KB article number. As you administer `WSUS`, you may discover certain updates that you can decline, since they do not impact certain target groups, or experience says they are not good to deploy in your organization. Keeping on top of which patches to approve or decline can be a lot of work, but it is vital to ensure that your systems are updated promptly.

Join our community on Discord

Join our community's Discord space for discussions with the author and other readers:

`https://packt.link/SecNet`