10

Managing Shared Data

In this chapter, we cover the following recipes:

* Managing NTFS files and folders
* Setting up and Securing an SMB File Server
* Creating and Securing SMB Shares
* Accessing SMB Shares
* Creating an iSCSI Target
* Using an ISCSI Target
* Implementing FSRM Quotas
* Implementing FSRM Reporting
* Implementing FSRM File Screening
* Implementing DFS Name Space
* Implementing DFS Replication

# Introduction

Sharing data with other users on your network has been a feature of computer operating systems from the very earliest days of networking. This chapter looks at Windows Server 2022 features that enable you to share files and folders and to use the data that you've shared.

Microsoft's LAN Manager was the company's first network offering. It enabled client computers to create, manage, and share files securely. LAN Manager's protocol to provide this client/server functionality was an early version of the Server Message Block (SMB) protocol.

SMB is a file-level storage protocol running over TCP/IP. SMB enables you to share files and folders securely and reliably. To increase reliability for SMB servers, you can install a cluster and cluster the file server role. A simple cluster solution is an active-passive solution – you have one cluster member sitting by if the other member fails. This solution works great as long as the underlying data is accessible. The Scale-Out File Server (SOFS) is a clustering-based solution that is active-active. Both nodes of the cluster can serve cluster clients.

This chapter shows you how to implement and leverage the features of sharing data between systems in Windows Server 2022.

In the first recipe, Managing NTFS File and Folder Permissions, you use the NTFS Security 3rd party module to set ACLs and ACL inheritance for files held in NTFS from SRV1. In the following recipe, Setting up and Securing an SMB file server, you deploy a hardened SMB file server. You run that recipe SRV2.

iSCSI is a popular Storage Area Networking (SAN) technology. Many SAN vendors provide iSCSI as a way to access data stored in a SAN. There are two aspects to iSCSI: the server (the iSCSI target) and the client (the iSCSI initiator). With the Creating an iSCSI Target recipe, you create an iSCSI target on the SS1 server, while in the Using an iSCSI target recipe, you make use of that shared iSCSI disk from SRV1 and SRV2.

File System Resource Manager (FSRM) is a Windows Server feature designed to help you manage file servers.

Several servers are involved in the recipes in this chapter—each recipe describes the specific server(s) you use for that recipe. As with other chapters, all the servers are members of the Reskit.Org domain on which you have loaded PowerShell 7 and VS Code. You can install them by using the Reskit.Org setup scripts on GitHub.

# Managing NTFS File and Folder Permissions

Every file and folder in an NTFS filesystem has an Access Control List (ACL). The ACL contains a set of Access Control Entries (ACEs). Each ACE defines permission to a file or folder for an account. For example, you could give the Sales AD global group full control of a file.

NTFS also allows a file or folder to inherit permission from its parent folder. If you create a new folder and then create a file within that new folder, the new file inherits the parent folder's permissions. You can manage the ACL list to add or remove permissions, and you can modify inheritance.

There's limited PowerShell support for managing NTFS permissions. PowerShell does have the Get-ACL and Set-ACL cmdlets, but creating the individual ACEs and managing inheritance requires using the .NET Framework (by default). A more straightforward approach is to use a third-party module, NTFSSecurity, which makes managing ACEs and ACLs, including dealing with inheritance, a lot easier.

## Getting Ready

This recipe uses SRV2, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. You also need DC1 online. In the storage chapter, you added several virtual disks to SRV1 and SRV2 and setup Storage Replica. You created an F: drive on both servers. Storage Replication should be replicating from disk 1 (the F: partition) in SRV2 to Disk 1 on SRV1 (effectively in a hidden partition). If your Hyper-V host is low on physical resources, consider removing the replication partnership.

## How to do it...

1. Downloading NTFSSecurity module from PSGallery

Install-Module NTFSSecurity -Force

1. Getting commands in the module

Get-Command -Module NTFSSecurity

1. Creating a new folder and a file in the folder

New-Item -Path F:\Secure1 -ItemType Directory |

  Out-Null

"Secure" | Out-File -FilePath F:\Secure1\Secure.Txt

Get-ChildItem -Path F:\Secure1

1. Viewing ACL of the folder

Get-NTFSAccess -Path F:\Secure1 |

  Format-Table -AutoSize

1. Viewing ACL of the file

Get-NTFSAccess F:\Secure1\Secure.Txt |

  Format-Table -AutoSize

1. Creating the Sales group in AD if it does not exist

$SB = {

  try {

    Get-ADGroup -Identity 'Sales' -ErrorAction Stop

  }

  catch {

    New-ADGroup -Name Sales -GroupScope Global |

      Out-Null

  }

}

Invoke-Command -ComputerName DC1 -ScriptBlock $SB

1. Displaying Sales AD Group

Invoke-Command -ComputerName DC1 -ScriptBlock {

                                   Get-ADGroup -Identity Sales}

1. Addding explicit full control for DomainAdmins

$AHT1 = @{

  Path         = 'F:\Secure1'

  Account      = 'Reskit\Domain Admins'

  AccessRights = 'FullControl'

}

Add-NTFSAccess @AHT1

1. Removing builtin\users access from secure.txt file

$AHT2 = @{

  Path         = 'F:\Secure1\Secure.Txt'

  Account      = 'Builtin\Users'

  AccessRights = 'FullControl'

}

Remove-NTFSAccess @AHT2

1. Removing inherited rights for the folder

$IRHT1 = @{

  Path                       = 'F:\Secure1'

  RemoveInheritedAccessRules = $True

}

Disable-NTFSAccessInheritance @IRHT1

1. Adding Sales group access to the folder

$AHT3 = @{

  Path         = 'F:\Secure1\'

  Account      = 'Reskit\Sales'

  AccessRights = 'FullControl'

}

Add-NTFSAccess @AHT3

1. Getting ACL on path

Get-NTFSAccess -Path F:\Secure1 |

  Format-Table -AutoSize

1. Getting resulting ACL on the file

Get-NTFSAccess -Path F:\Secure1\Secure.Txt |

  Format-Table -AutoSize

## How it works...

In step 1, you download and install the NTFSSecurity module from the PowerShell Gallery. This step creates no console output. In step 2, you take a look at the commands provided by the NTFSSecurity module, with output like this:

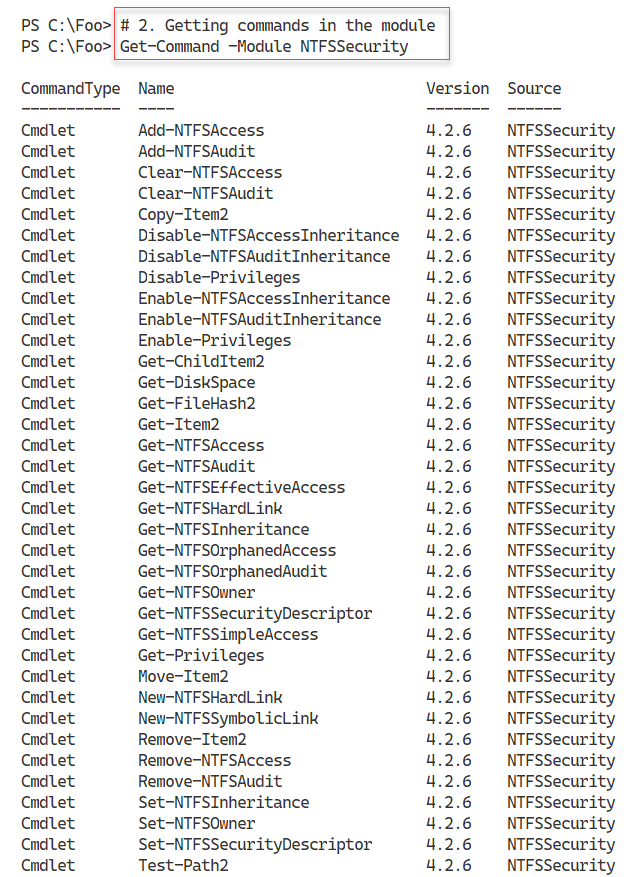
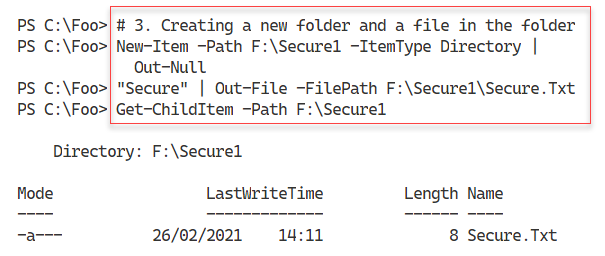


Figure 10.1: Viewing commands in the NTFSSecurity module

**Insert image B42024\_10\_01.png**

In step 3, you create a folder and a file within that folder, with output like this:



In step 4, you use the Get-NTFSAccess cmdlet to return the ACL of the folder, with output like this:

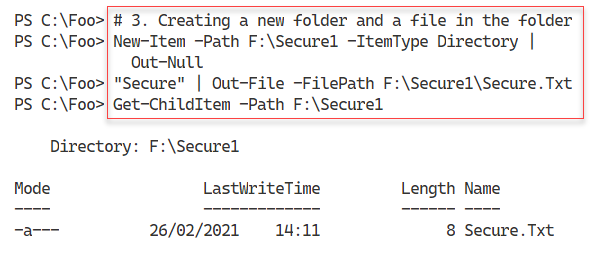


Figure 10.3: Viewing ACL of F:\Secure1 folder

**Insert image B42024\_10\_03.png**

In step 5, you view the ACL of the F:\Secure1\Secure.Txt with output like this:

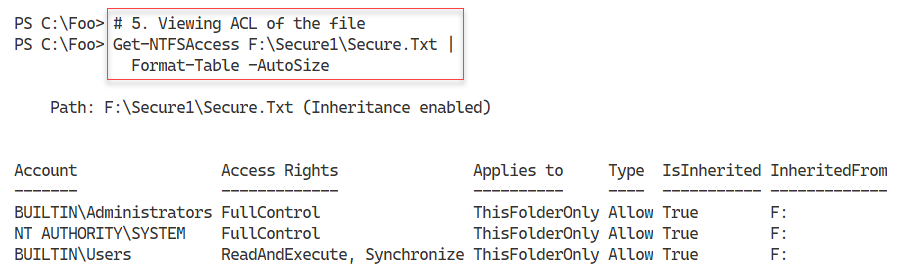


Figure 10.4: Viewing ACL of F:\Secure1\Secure.txt folder

**Insert image B42024\_10\_04.png**

In step 6, you create a global group, Sales, in the Reskit.Org domain if it does not already exist. This step creates no output. In step 7, you view the Sales group from the AD, with output like this:

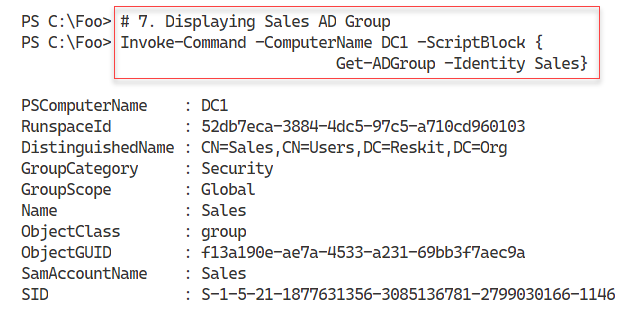


Figure 10.5: Viewing the Sales global group

**Insert image B42024\_10\_05.png**

In step 8, you add an explicit Access Control Entry (ACE) for the Domain Admins group for full control of the F:\Secure1 folder. Next, in step 9, you remove the builtin\users group from access to the Secure.txt file. And then, in step 10, you remove all inherited access from F:\Secure 1. Finally, in step 11, you add explicit access to the F:\Secure1 folder to the Sales group. These four steps produce no output.

In step 12, you examine the updated ACL for the F:\Secure folder, with output like this:

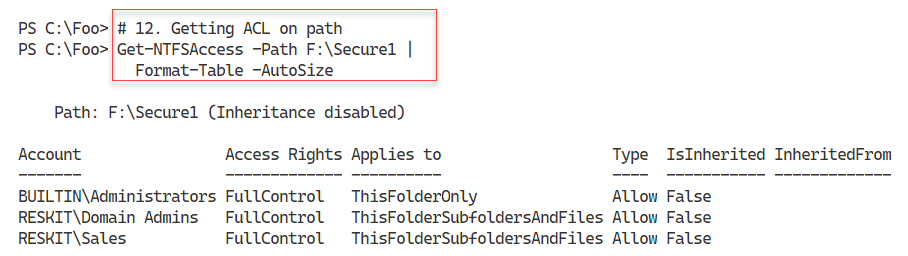


Figure 10.6: Viewing updated ACL for the F:\Secure folder

**Insert image B42024\_10\_06.png**

In step 13, you look at the updated ACL for F:\Secure\Secure1.txt file, with output like this:

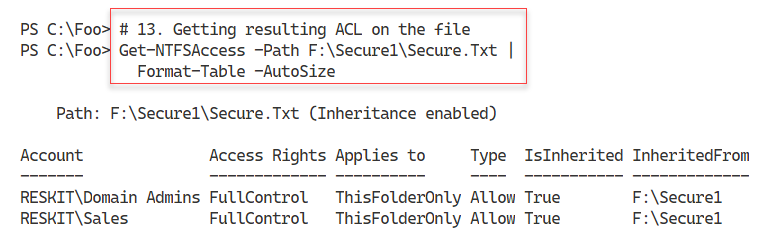


Figure 10.7: Viewing the updated ACL for F:\Secure1\Secure.Txt

**Insert image B42024\_10\_07.png**

## There's more...

As you can see in step 2, there are several cmdlets in the NTFSSecurity module. You can use these cmdlets to set up the ACL on a file or folder and the System ACL list (SAC) that enables you to audit file or folder access. There are also some improved cmdlets, such as Get‑ChildItem2 and Get-Item2, that you may find helpful.

# Setting up and Securing an SMB File Server

The next step in creating a file server is to install the necessary features to the server, then harden it. You use the Add-WindowsFeature cmdlet to add the features necessary for a file server. You can then use the Set-SmbServerConfiguration cmdlet to improve the configuration.

Since your file server can contain sensitive information, you must take reasonable steps to avoid some of the expected attack mechanisms and adopt best security practices. Security is a good thing but, as always, be careful! By locking down your SMB file server too hard, you can lock some users out of the server.

SMB 1.0 has many weaknesses and, in general, should be removed. By default Windows, Server 2022 installs with SMV 1.0 turned off. Remember that if you disable SMB 1.0, you may find that older computers (for example, those running Windows XP) lose the ability to access shared data. Before you lock down any of the server configurations, be sure to test your changes carefully.

## Getting Ready

This recipe uses SRV2, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. You also use SRV2 and should have DC1 online.

## How to do it...

1. Adding File Server features to SRV2

$Features = 'FileAndStorage-Services',

            'File-Services',

            'FS-FileServer',

            'RSAT-File-Services'

Add-WindowsFeature -Name $Features

1. Viewing the SMB server settings

Get-SmbServerConfiguration

1. Turning off SMB1

$CHT = @{

  EnableSMB1Protocol = $false

  Confirm            = $false

}

Set-SmbServerConfiguration @CHT

1. Turning on SMB signing and encryption

$SHT1 = @{

    RequireSecuritySignature = $true

    EnableSecuritySignature  = $true

    EncryptData              = $true

    Confirm                  = $false

}

Set-SmbServerConfiguration @SHT1

1. Turning off default server and workstations shares

$SHT2 = @{

    AutoShareServer       = $false

    AutoShareWorkstation  = $false

    Confirm               = $false

}

Set-SmbServerConfiguration @SHT2

1. Turning off server announcements

$SHT3 = @{

    ServerHidden   = $true

    AnnounceServer = $false

    Confirm        = $false

}

Set-SmbServerConfiguration @SHT3

1. Restarting SMB Server service with the new configuration

Restart-Service lanmanserver -Force

## How it works...

In step 1, you add the file server features to SRV2, with output like this:

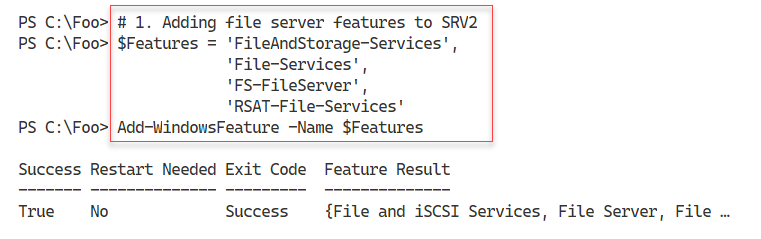


Figure 10.8: Adding file server features to SRV2

**Insert image B42024\_10\_08.png**

In step 2, you use the Get-SmbServerConfiguration cmdlet to return the SMB server settings for SRV2, which looks like this:

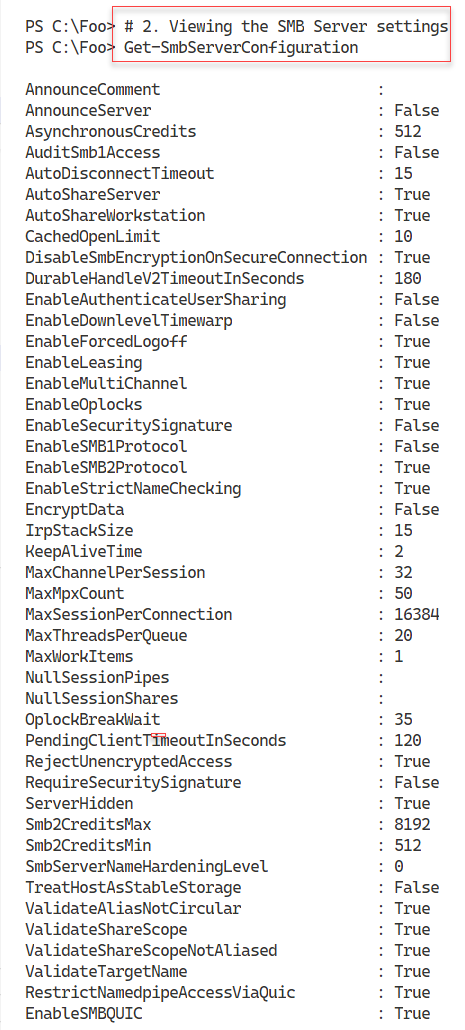


Figure 10.9: Viewing SMB Server settings on SRV2

**Insert image B42024\_10\_09.png**

In step 3, you turn off SMB 1.0 explicitly. In step 4, you turn on digital signing and encrypting of all SMB related data packets. With step 5, you turn off the default server and workstation shares, and with step 6, you turn off SMB Server announcements to improve security. These 4 steps produce no output.

In step 7, which also produces no output, you restart the LanManServer service, which is the Windows service that provides the SMB file sharing.

## There's more...

In step 7, you restart the LanManWorkstation service. You restart this service to implement the changes to the File Server configuration. This step produces no output.

# Creating and Securing SMB Shares

With your file server service set up, the next step in deploying a file server is to create SMB shares and secure them. For decades, administrators have used the net.exe command to set up shared folders and more. These continue to work, but you may find the new cmdlets easier to use, particularly if you're automating large-scale SMB server deployments.

This recipe looks at creating and securing shares on a Server 2022 platform using the PowerShell SMBServer module. You also use cmdlets from the NTFSSecurity module (a third-party module you download from the PS Gallery).

You run this recipe on the file server (SRV2) that you set up and hardened to secure an SMB File Serverrecipe. In this recipe, you share a folder (C:\Foo) on the file server. You created this folder previously. Then, you create a file in the C:\Foo folder you just shared and set the ACL for the files to be the same for the share. You use the Set-SMBPathAcl cmdlet to do this. You then review the ACL for both the folder and the file.

This recipe uses a global security group, Sales, which you create in the Reskit.Org domain. See the introduction section in Chapter 7, Managing Printers, for the script snippet you can use to create the groups, users, and group memberships used by this recipe. In this recipe, you use the Get‑NTFSAccess cmdlet from NTFSSecurity, a third-party module that you downloaded from the PowerShell Gallery. See the Managing NTFS File and Folder Permissions recipe for more details about this module and instructions on downloading it.

## Getting Ready

This recipe uses SRV2, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. You also use SRV2 and should have DC1 online.

## How to do it...

1. Discovering existing shares and access rights

Get-SmbShare -Name \* |

  Get-SmbShareAccess |

    Format-Table -GroupBy Name

1. Sharing a new folder

New-Item -Path F: -Name ITShare -ItemType Directory |

Out-Null

New-SmbShare -Name ITShare -Path C:\ITShare

1. Updating the share to have a description

$CHT = @{Confirm=$False}

Set-SmbShare -Name Foo -Description 'File Share for IT' @CHT

1. Setting folder enumeration mode

$CHT = @{Confirm = $false}

Set-SMBShare -Name ITShare -FolderEnumerationMode AccessBased @CHT

1. Setting encryption on for ITShare share

Set-SmbShare –Name ITShare -EncryptData $true @CHT

1. Removing all access to ITShare share for the Everyone group

$AHT1 = @{

  Name        =  'ITShare'

  AccountName = 'Everyone'

  Confirm     =  $false

}

Revoke-SmbShareAccess @AHT1 | Out-Null

1. Adding Reskit\Administrators to have read permissions

$AHT2 = @{

    Name         = 'ITShare'

    AccessRight  = 'Read'

    AccountName  = 'Reskit\ADMINISTRATOR'

    ConFirm      =  $false

}

Grant-SmbShareAccess @AHT2 | Out-Null

1. Adding system full access

$AHT3 = @{

    Name          = 'ITShare'

    AccessRight   = 'Full'

    AccountName   = 'NT Authority\SYSTEM'

    Confirm       = $False

}

Grant-SmbShareAccess  @AHT3 | Out-Null

1. Setting Creator/Owner to Full Access

$AHT4 = @{

    Name         = 'foo'

    AccessRight  = 'Full'

    AccountName  = 'CREATOR OWNER'

    Confirm      = $False

}

Grant-SmbShareAccess @AHT4  | Out-Null

1. Granting Sales group read access, SalesAdmins has Full access

$AHT5 = @{

    Name        = 'Foo'

    AccessRight = 'Read'

    AccountName = 'Sales'

    Confirm     = $false

}

Grant-SmbShareAccess @AHT5 | Out-Null

1. Reviewing share access

Get-SmbShareAccess -Name Foo |

  Sort-Object AccessRight

1. Setting file ACL to be same as share ACL

Set-SmbPathAcl -ShareName 'Foo'

1. Creating a file in c:\Foo

'File Contents' | Out-File -FilePath C:\Foo\Foo.Txt

1. Setting the file ACL to be the same as share ACL

Set-SmbPathAcl -ShareName 'Foo'

1. Viewing folder ACL using Get-NTFSAccess

Get-NTFSAccess -Path C:\Foo |

  Format-Table -AutoSize

1. Viewing the file ACL

Get-NTFSAccess -Path C:\Foo\Foo.Txt |

  Format-Table -AutoSize

## How it works...

In step 1, you use Get-SmbShare to discover the current SMB Shares on SRV2 and which accounts have access to those shares. The output looks like this:

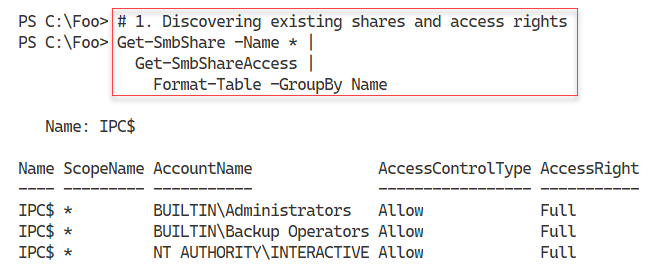


Figure 10.10: Viewing SMB Server settings on SRV2

**Insert image B42024\_10\_10.png**

In step 2, you create a new folder on the F:\ drive and share the folder as \\SRV2\ITShare. The output from this step looks like this:

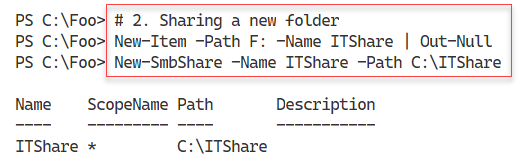


Figure 10.11: Creating and sharing F:\ITShare as \\SRV2\ITShare

**Insert image B42024\_10\_11.png**

Having created the share, you next configure share access details. In step 3, you modify the share to have a description. With step 4, you set access-based enumeration on the share. Then, in step 5, you ensure Windows encrypts all data transferred via the share. Next, with step 6, you remove access to the ITShare for the Everyone group. In step 7, you grant the Reskit\Administrator group read permission on the ITShare. With step 8, you give the OS full access to the share. Finally, in step 9, you grant the creator or owner of any file/folder full access to the file. These 7 configuration steps produce no output.

In step 11, you review the access to the share, which produces output like this:

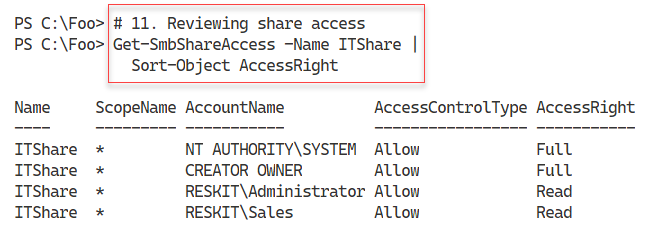


Figure 10.12: Viewing access to ITShare

**Insert image B42024\_10\_12.png**

Now that you have access to the share configured as needed, in step 12, you use the Set-SMBPathAcl command to make the NTFS permissions match the SMB share permissions. In step 13, you create a new file in the folder shared as ITShare and then ensure, in step 14, that the file itself has the same ACL as the share. These three steps produce no output.

In step 15, you view the file, F:\ITShare\File.txt, which produces output like this:

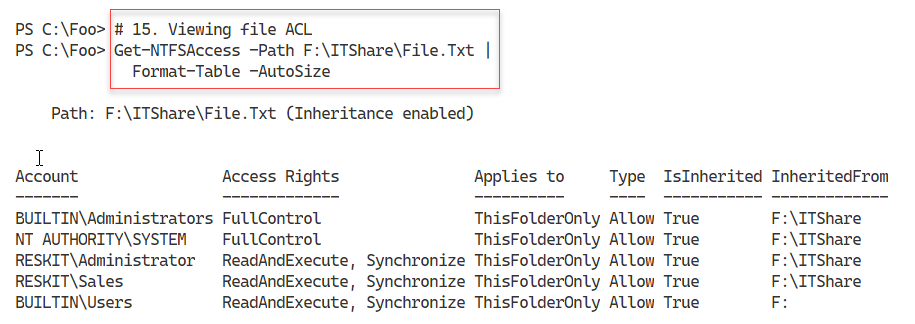


Figure 10.13: Viewing access to the file

**Insert image B42024\_10\_13.png**

## There's more...

In step 1, you examine the shares available on SRV2. In Setting up and Securing an SMB File Server recipe, you configured the SMB service to remove the default shares on SRV2. The only share you see in step 1 is the IPC$ share, which Windows uses for named pipes communication mechanism. For more details about this share, see https://docs.microsoft.com/troubleshoot/windows-server/networking/inter-process-communication-share-null-session.

In step 4, you set access-based enumeration for the ITShare share. This setting means that any user viewing files or folders within the share only see objects to which they have access. This setting improves security and minimises the administrative questions, “What is this file/folder, and why can’t I have access to this file/folder”.

In step 5, you set encryption on for the ITShare share. This step ensures that Windows performs data encryption on any data transferred across this share. You can set this by default at the server level or, as in this case, at the share level.

# Accessing SMB Shares

## Getting Ready

This recipe uses SRV1, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. You also use SRV2 and should have DC1 online. You previously created SMB shares on SRV2, which you use in this recipe.

## How to do it...

1. Examining the SMB client's configuration on SRV2

Get-SmbClientConfiguration

1. Setting signing of SMB packets

$CHT = @{Confirm=$false}

Set-SmbClientConfiguration -RequireSecuritySignature $True @CHT

1. Examine SMB client's network interface

Get-SmbClientNetworkInterface |

    Format-Table

1. Examining the shares provided by SRV2

net view \\SRV2

1. Creating a drive mapping, mapping the r: to the share on server SRV2

New-SmbMapping -LocalPath R: -RemotePath \\SRV2\ITShare

1. Viewing the shared folder mapping

Get-SmbMapping

1. View the shared folder contents

Get-ChildItem -Path R:

1. Viewing existing connections

Get-SmbConnection

1. Viewing open files on SRV2

Get-SmbOpenFile

## How it works...

In step 1, you examine the SRV2’s SMB client configuration, with output like this:

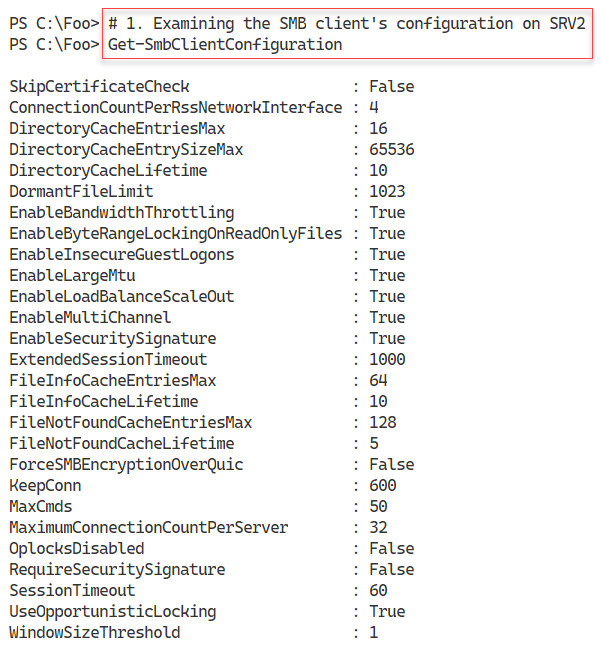


Figure 10.14: Viewing access to the file

**Insert image B42024\_10\_14.png**

In step 2, you ensure that SRV1 requires signed SMB Packets, irrespective of settings on the SMB Server (SRV2). There is no output from this step.

In step 3, you examine the client NIC on SRV1, with output that looks like this:

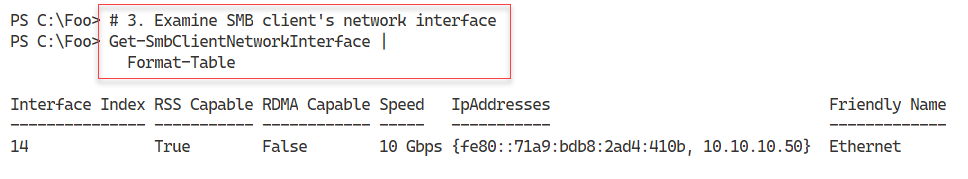


Figure 10.15: Viewing NIC on SRV1

**Insert image B42024\_10\_15.png**

In step 4, you use the net.exe command to view the shares provided by the SRV2 host. The output from this step looks like this:

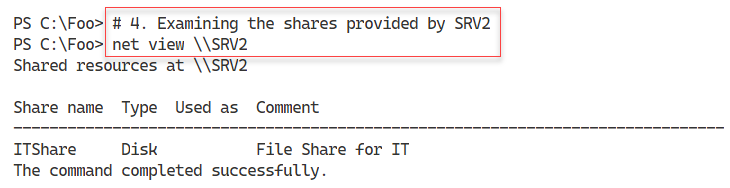


Figure 10.16: Viewing shares offered by SRV2

**Insert image B42024\_10\_16.png**

In step 5, you create a new drive mapping on SRV1, mapping the R: drive to \\SRV2\ITShare, which creates output that looks like this:

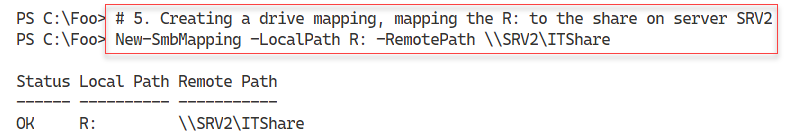


Figure 10.17: Creating a new drive mapping on SRV1

**Insert image B42024\_10\_17.png**

In step 6, you view the SMB drive mappings on SRV1, which looks like this:

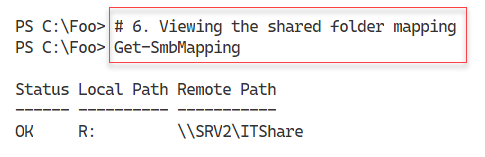


Figure 10.18: Viewing SMB shares provided by SRV1

**Insert image B42024\_10\_18.png**

In step 7, you view the contents of the share to reveal the file you created in “Creating and Securing SMB Shares”, with output like this:

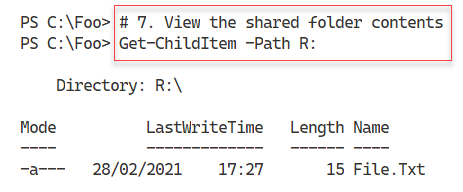


Figure 10.19: Viewing the contents of the shared folder

**Insert image B42024\_10\_19.png**

In step 8, you view all existing SMB connections from SRV1. This step produces the following output:

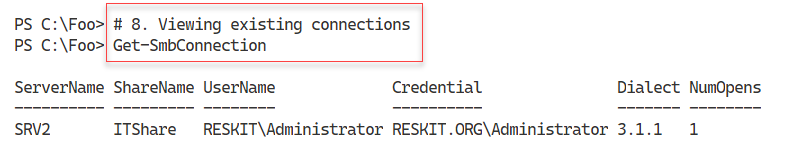


Figure 10.20: Viewing existing SMV connections from SRV1

**Insert image B42024\_10\_20.png**

## There's more...

In step 4, you use the net.exe command to view the shares offered by SRV2 (from SRV1). The SMBShare module does not provide a cmdlet that views shares offered by a remote host.

In step 8, you note that there are no open files on SRV2, which is what you might expect.

# Creating an iSCSI Target

iSCSI is an industry-standard protocol that implements block storage over a TCP/IP network. With iSCSI, the server, or target, provides a volume shared via iSCSI to an iSCSI client, also known as the initiator.

With iSCSI, you giver each shared volumes an iSCSI logical unit numbers. The iSCSI client then sees that remote disk (LUN) as locally attached. From the iSCSI client, you can manage the disk just like locally-attached storage. Windows Server 2022 includes both iSCSI target (server) and iSCSI initiator (client) features.

You set up an iSCSI target on a server and then use an iSCSI initiator on another server (or client) system to access the iSCSI target. You can use both Microsoft and third-party initiators and targets, although if you mix and match, you need to test very carefully that the combination works in your environment.

With iSCSI, a target is a single disk that the client accesses using the iSCSI Client. An iSCSI target server hosts one or more targets, where each iSCSI target is equivalent to a LUN on a Fiber Channel SAN.

You could use iSCSI in a cluster of Hyper-V servers. The servers in the cluster can use the iSCSI initiator to access an iSCSI target. Used via the Cluster Shared Volume, the shared iSCSI target is shared between nodes in a failover cluster that enables the VMs in that cluster to be highly available.

## Getting Ready

This recipe uses SS1, domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code.

## How to do it...

1. Installing the iSCSI target feature on SS1

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature FS-iSCSITarget-Server

1. Exploring iSCSI target server settings

Get-IscsiTargetServerSetting

1. Creating a folder on SS1 to hold the iSCSI virtual disk

$NIHT = @{

  Path        = 'C:\iSCSI'

  ItemType    = 'Directory'

  ErrorAction = 'SilentlyContinue'

}

New-Item @NIHT | Out-Null

1. Creating an iSCSI virtual disk (that is a LUN):

$LP = 'C:\iSCSI\ITData.Vhdx'

$LN = 'SalesTarget'

$VDHT = @{

   Path        = $LP

   Description = 'LUN For IT Group'

   SizeBytes   = 500MB

 }

New-IscsiVirtualDisk @VDHT

1. Setting the iSCSI target, specifying who can initiate an iSCSI connection.

$THT = @{

  TargetName   = $LN

  InitiatorIds = 'DNSNAME:SRV1.Reskit.Org',

                 'DNSNAME:SRV2.Reskit.Org'

}

New-IscsiServerTarget @THT

1. Creating iSCSI disk target mapping LUN name to a local path

Add-IscsiVirtualDiskTargetMapping -TargetName $LN -Path $LP

## How it works...

In step 1, you install the iSCSI target feature on the SS1 server, with output like this:

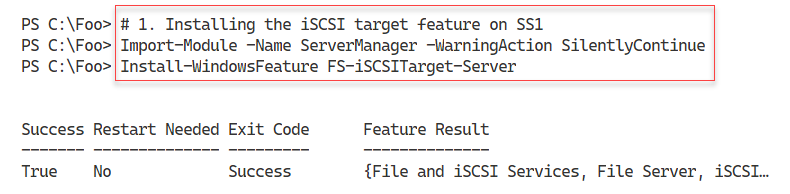


Figure 10.21: Installing the iSCSI target feature to SS1

**Insert image B42024\_10\_21.png**

In step 2, you examine the iSCSI target server settings, with output that looks like this:

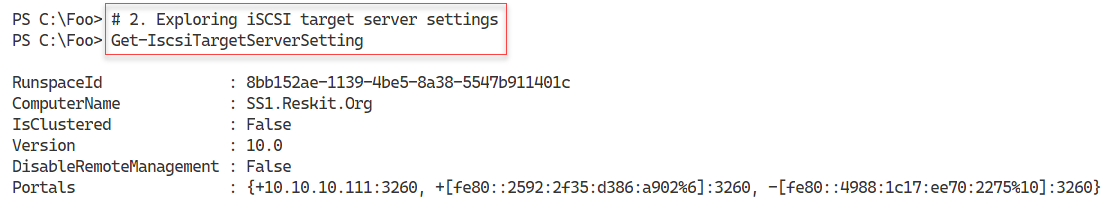


Figure 10.22 Examining the iSCSI target server settings

**Insert image B42024\_10\_22.png**

In step 3, you create a folder on SS1 to hole the iSCSI virtual disk, which creates no output. In step 4, you create an iSCSI virtual disk (essentially a LUN) with output that looks like this:

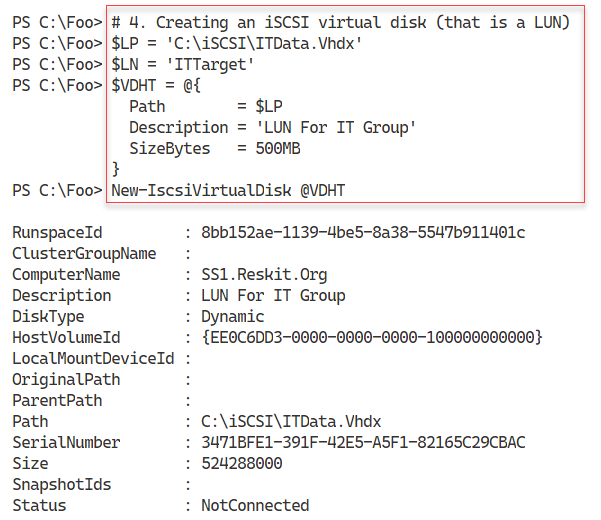


Figure 10.23 Creating an iSCSI virtual disk on SS1

**Insert image B42024\_10\_23.png**

In step 5, you specify which computers can use the virtual iSCSI target, with output like this:

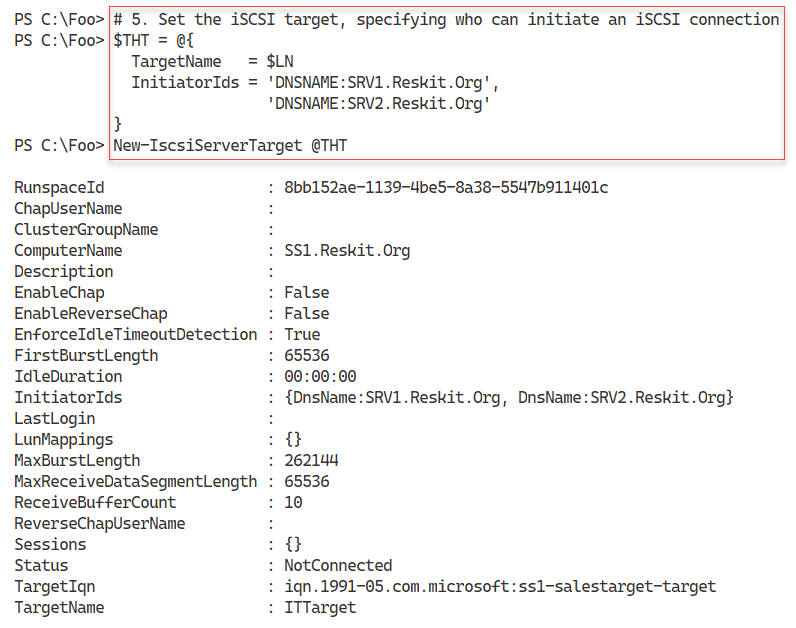


Figure 10.24 Specifying which hosts can access the iSCSI virtual disk

**Insert image B42024\_10\_24.png**

In the final step, step 6, you specify the disk target mapping. This step creates a mapping between an iSCSI target name (ITTarget) and the local path where you stored the virtual iSCSI hard disk.

## There's more...

By default, Windows does not install the iSCSI target feature, but as you can see in step 1, you use Install-WindowsFeature to add the feature to this storage server.

When you create an iSCSI target, you create the target name and the target virtual hard drive separately, and then, in step 6, you map the iSCSI target name to the file location. In production, you would use a separate set of (fault-tolerant) disks to hold the iSCSI information, possibly using Storage Spaces to create fault-tolerant virtual disks.

# Using an ISCSI Target

## Getting Ready

This recipe uses SRV1, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. You also use SS1 and should have DC1 online. You previously created an iSCSI target (on SS1), and now you use the built-in iSCSI initiator to access the iSCSI disk. You run this recipe on SRV1.

## How to do it...

1. Adjusting the iSCSI service to auto start, then start the service

Set-Service MSiSCSI -StartupType 'Automatic'

Start-Service MSiSCSI

1. Setting up the portal to SS1

$PHT = @{

  TargetPortalAddress     = 'SS1.Reskit.Org'

  TargetPortalPortNumber  = 3260

}

New-IscsiTargetPortal @PHT

1. Finding and viewing the ITTarget on the portal

$Target  = Get-IscsiTarget |

               Where-Object NodeAddress -Match 'ITTarget'

$Target

1. Connecting to the target on SS1

$CHT = @{

  TargetPortalAddress = 'SS1.Reskit.Org'

  NodeAddress         = $Target.NodeAddress

}

Connect-IscsiTarget  @CHT

1. Viewing the ISCSI disk from SRV1 on SS1

$ISD =  Get-Disk |

  Where-Object BusType -eq 'iscsi'

$ISD |

  Format-Table -AutoSize

1. Turning disk online and making disk R/W

$ISD |

  Set-Disk -IsOffline  $False

$ISD |

  Set-Disk -Isreadonly $False

1. Formating the volume on SS1

$NVHT = @{

  FriendlyName = 'ITData'

  FileSystem   = 'NTFS'

  DriveLetter  = 'I'

}

$ISD |

  New-Volume @NVHT

1. Using the drive as a local drive

Set-Location -Path I:

New-Item -Path I:\  -Name ITData -ItemType Directory |

  Out-Null

'Testing 1-2-3' |

  Out-File -FilePath I:\ITData\Test.Txt

Get-ChildItem I:\ITData

## How it works...

In step 1, you set the iSCSI service to start when SRV1 starts automatically, and then you explicitly start the iSCSI service. This step creates no console output.

In step 2, you set up the iSCSI portal to SS1, which looks like this:

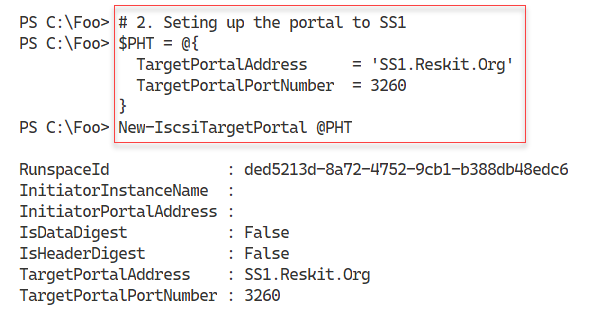


Figure 10.25: Setting up the iSCSI portal to SS1

**Insert image B42024\_10\_25.png**

In step 3, you find and view the ITTarget LUN from the SS1. The output looks like this:

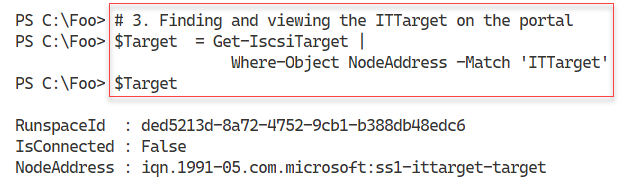


Figure 10.26: Viewing iTTarget LUN

**Insert image B42024\_10\_26.png**

In step 4, you connect from SRV1 to the iSCSI target on SS1, which looks like this:

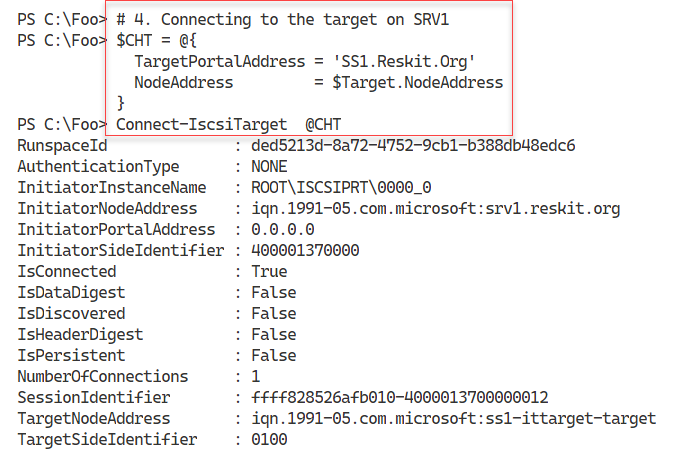


Figure 10.27: Connecting to the iSCSI target on SS1

**Insert image B42024\_10\_27.png**

In step 5, you use the Get-Disk to view the iSCSI disk from SRV1, which looks like this:

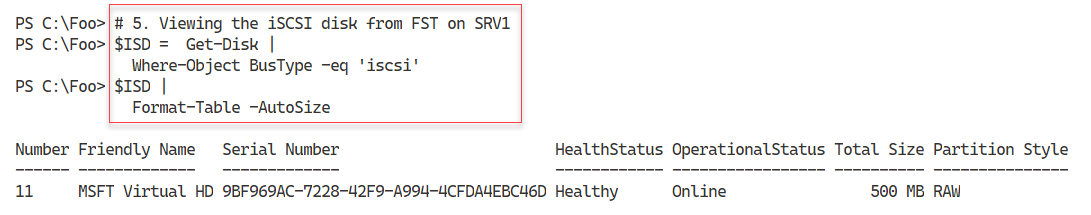


Figure 10.28: Viewing the iSCSI disk

**Insert image B42024\_10\_28.png**

In step 6, you ensure the iSCSI disk is online and Read/Write – a step that generates no output. In step 7, you create a new volume on the iSCSI disks, which looks like this:

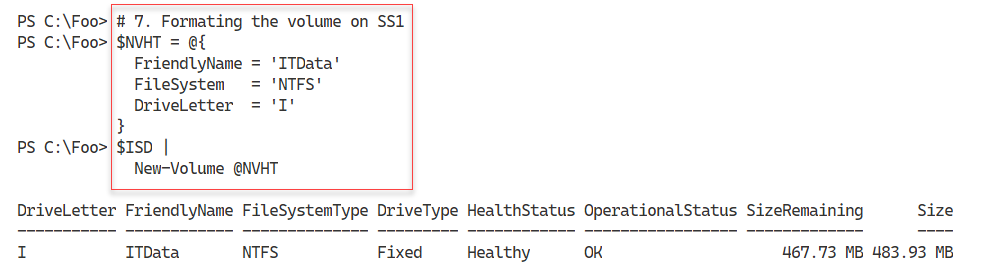


Figure 10.29: Formatting the iSCSI disk

**Insert image B42024\_10\_29.png**

In the final step in this recipe, step 8, you create a folder in the iSCSI disk. Then you create a file and view the file, which looks like this:

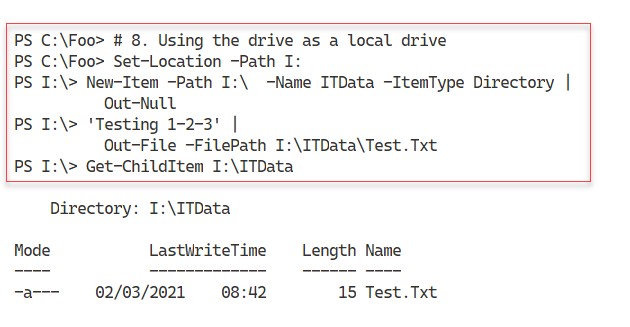


Figure 10.30: Using the iSCSI disk

**Insert image B42024\_10\_30.png**

## There's more...

# Implementing FSRM Quotas

The FSRM is a feature of the Windows server that assists you in managing file servers. FSRM  
has three key features:

* Quota management: With FSRM, you can set soft or hard quotas on volumes and folders. A soft quota allows a user to exceed an allowance, while hard quotas stop a user from exceeding an allowance. You can configure a quota with thresholds and threshold actions. If a user exceeds 65% of the quota allowance, FSRM can send an email, while at 90%, you log an event in the event log or run a program. You have different actions for different quota levels. This recipe shows how to use quotas.
* File screening: You can set up a file screen and stop a user from saving screened files. For example, you could screen for .MP3 or FLAC files—should a user then attempt to save a file (say, jg75-02-28D1T1.flac), the file screen rejects the request and doesn't allow the user to save the file.
* Reporting: FSRM enables you to create a wealth of storage reports that can be highly useful for management purposes

In this recipe, you install FSRM, perform some general configuration, and then work with soft and hard quotas.

## Getting Ready

This recipe uses SRV1, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code.

This recipe makes use of an email server so that FSRM can send an email to the administrator. To test this recipe's email-related components, you need to have an SMTP server or an email forwarder. This recipe sends mail to a host (SMTP.Reskit.Org), then forwards the mail to a free email service at https://www.sendgrid.com. For more details on how to set up SendGrid to forward the email, see <https://tfl09.blogspot.com/2020/04/setting-up-smtp-relay-using-sendgrid.html>. If you have an SMTP server that accepts mail from FSRM, then change this recipe to use that server.

Note that Microsoft may remove the SMTP relay service Windows Server 2022 before shipping the OS.

## And How to do it...

1. Installing FS Resource Manager feature on SRV1

Import-Module -Name ServerManager -WarningAction 'SilentlyContinue'

$IHT = @{

  Name                   = 'FS-Resource-Manager'

  IncludeManagementTools = $True

  WarningAction          = 'SilentlyContinue'

}

Install-WindowsFeature @IHT

1. Setting SMTP settings in FSRM

$MHT = @{

  SmtpServer        = 'SMTP.Reskit.Org'

  FromEmailAddress  = 'FSRM@Reskit.Org'

  AdminEmailAddress = 'Doctordns@Gmail.Com'

}

Set-FsrmSetting @MHT

1. Sending and viewing a test email to check the setup

$MHT = @{

  ToEmailAddress = 'DoctorDNS@gmail.com'

  Confirm        = $false

}

Send-FsrmTestEmail @MHT

1. Creating a new FSRM quota template for a 10MB hard limit

$QHT1 = @{

  Name        = '10 MB Reskit Quota'

  Description = 'Filestore Quota (10mb)'

  Size        = 10MB

}

New-FsrmQuotaTemplate @QHT1

1. Viewing available FSRM quota templates

Get-FsrmQuotaTemplate |

  Format-Table -Property Name, Description, Size, SoftLimit

1. Creating a new folder on which to apply a quota

If (-Not (Test-Path C:\Quota)) {

  New-Item -Path C:\Quota -ItemType Directory  |

    Out-Null

}

1. Building an FSRM action

$Body = @'

User [Source Io Owner] has exceeded the [Quota Threshold]% quota

threshold for the quota on [Quota Path] on server [Server].

The quota limit is [Quota Limit MB] MB, and [Quota Used MB] MB

currently is in use ([Quota Used Percent]% of limit).

'@

$NAHT = @{

  Type      = 'Email'

  MailTo    = 'Doctordns@gmail.Com'

  Subject   = 'FSRM Over limit [Source Io Owner]'

  Body      = $Body

}

$Action1 = New-FsrmAction @NAHT

1. Creating an FSRM threshold

$Thresh = New-FsrmQuotaThreshold -Percentage 85 -Action $Action1

1. Building a quota for the C:\Quota folder

$NQHT1 = @{

  Path      = 'C:\Quota'

  Template  = '10 MB Reskit Quota.'

  Threshold = $Thresh

}

New-FsrmQuota @NQHT1

1. Testing the 85% soft quota limit on C:\Quota

Get-ChildItem -Path C:\Quota -Recurse |

  Remove-Item -Force     # for testing purposes!

$S = '+'.PadRight(8MB)

# Make a first file - under the soft quota

$S | Out-File -FilePath C:\Quota\Demo1.Txt

$S2 = '+'.PadRight(.66MB)

# Now create a second file to take the user over the soft quota

$S2 | Out-File -FilePath C:\Quota\Demo2.Txt

1. Testing hard limit quota

$S | Out-File -FilePath C:\Quota\Demo3.Txt

1. Viewing folder contents

Get-ChildItem -Path C:\Quota

## How it works...

In step 1, you use the Install-WindowsFeature cmdlet to add the FS-ResourceManager feature to SRV1, which looks like this:

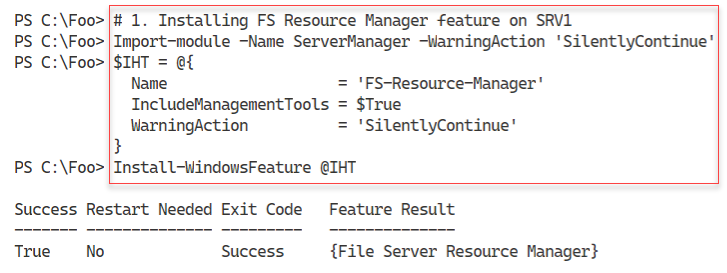


Figure 10.31: Installing FSRM Feature to SRV1

**Insert image B42024\_10\_31.png**

In step 2, you set SMTP details, including the SMPT server name, the from and admin addresses. This step produces no output. In step 3, you use the Send-FsrmTestEmail cmdlet to test SMTP email handling. This step produces no console output but does generate an email, which looks like this:

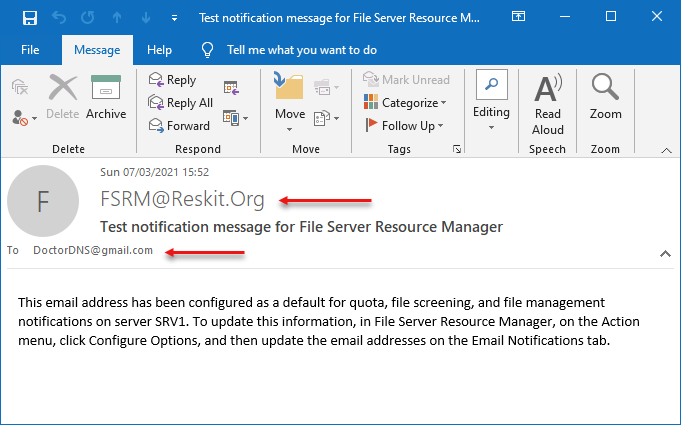


Figure 10.32: Test email received from FSRM

**Insert image B42024\_10\_32.png**

In step 4, you create a new FSRM quota template for a 10MB hard quota limit. The output from this step looks like this:

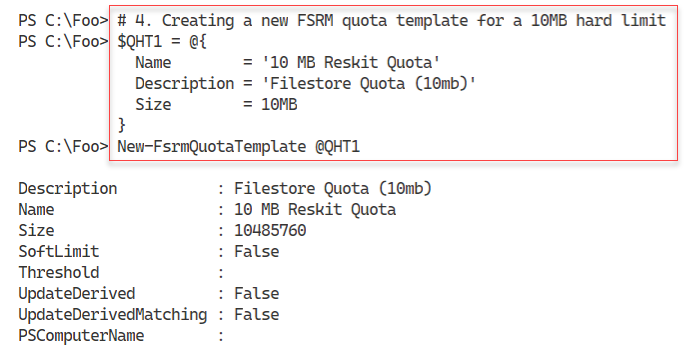


Figure 10.33: Creating an FSRM quota template

**Insert image B42024\_10\_33.png**

In step 5, you view all available FSRM quota templates, with output like this:

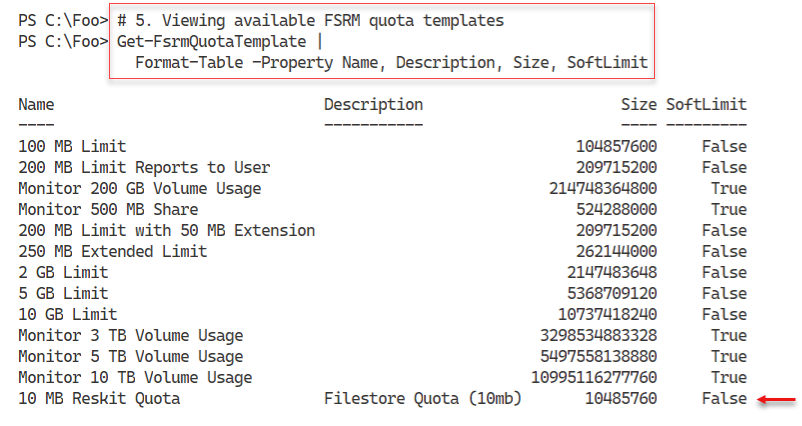


Figure 10.34: Viewing available FSRM quota templates

**Insert image B42024\_10\_34.png**

In step 6, you create a new folder, C:\Quota. In step 7, you build an FSRM action that sends an email whenever a user exceeds the quota. In step 8, you create an FSRM threshold (how much of the soft quota limit a user can use before triggering a quota violation). These three steps produce no console output.

In step 9, you build a quota for the C:\Quota folder, with output that looks like this:

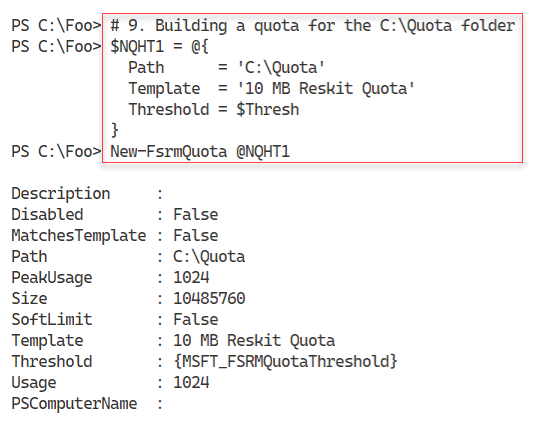


Figure 10.35: Building a quota for C:\Quota folder

**Insert image B42024\_10\_35.png**

In step 10, you test the 85% soft quota limit. First, you create a new file (C:\Quota\Demo1.Txt) that is under the soft quota limit. Then, you create a second file (C:\Quota\Demo1.Txt) that uses up more than the soft quota limit. There is no console output from this step, but FSRM detects you have exceeded the soft limit quota for this folder and generates an email message that looks like this:

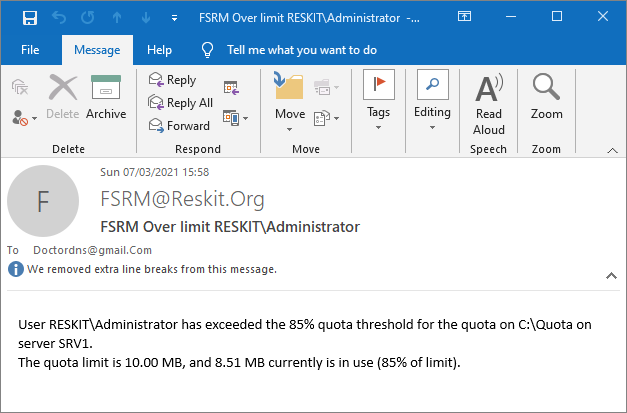


Figure 10.36: Exceeding soft quota limit email

**Insert image B42024\_10\_36.png**

In step 11, you attempt to create an additional file, C:\Quota\Demo3.Txt, which exceeds the hard-quota limit. You see the following output:

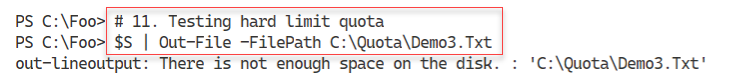


Figure 10.37: Testing hard limit quota

**Insert image B42024\_10\_37.png**

In step 12, you examine the files in the C:\Quota folder, which looks like this:

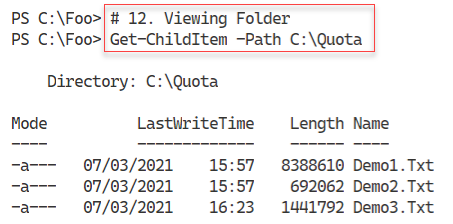


Figure 10.38: Viewing C:\Quota folder

**Insert image B42024\_10\_38.png**

## There's more...

In this recipe, you set up and tested both a soft and a hard FSRM quota. With the soft quota, you configured FSRM to send an email to inform the recipient that they exceeded a quota while the hard quota stops the user from saving more than the allowed quota. While the soft quota means a user can exceed the quota, the user can only save up to the quota with a hard quota.

The quotas set in this recipe are small and would probably not be of much use in production. But a simple change from, say, 10 MB to 10 GB would be simple to make. Also, for the soft quota, the quota exceeded the action results in FSRM sending an email, while for the hard quota, FSRM writes Application event-log entries and stops the user from saving excess data. You might want to send an email to either an administrator or a user who has exceeded the quota thresholds.

In step 4, you create a new FSRM quota template. You can see this new template in the output generated by step 5. Note that this quota template is for a hard, not a soft quota limit.

In step 12, you examine the C:\Quota folder. Notice that with the third file (which you attempted to create in step 11), Windows has not saved the entire file. If you are planning on imposing hard quotas, you must ensure users understand the implications of exceeding any hard quota limits.

# Implementing FSRM Reporting

## Getting Ready

This recipe uses SRV1, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. In the previous recipe, “Implementing FSRM Quotas”, you have installed FRSRM on SRV1.

## How to do it...

1. Creating a new FSRM storage report for large files on C:\ on SRV1

$NRHT = @{

  Name             = 'Large Files on SRV1'

  NameSpace        = 'C:\'

  ReportType       = 'LargeFiles'

  LargeFileMinimum = 10MB

  Interactive      = $true

}

New-FsrmStorageReport @NRHT

1. Getting existing FSRM reports

Get-FsrmStorageReport \* |

  Format-Table -Property Name, NameSpace,

                         ReportType, ReportFormat

1. Viewing interactive reports available on SRV1

$Path = 'C:\StorageReports\Interactive'

Get-ChildItem -Path $Path

1. Viewing the report

$Rep = Get-ChildItem -Path $Path\\*.html

Invoke-Item -Path $Rep

1. Extracting key information from the FSRM XML output

$XF   = Get-ChildItem -Path $Path\\*.xml

$XML  = [XML] (Get-Content -Path $XF)

$Files = $XML.StorageReport.ReportData.Item

$Files | Where-Object Path -NotMatch '^Windows|^Program|^Users'|

  Format-Table -Property name, path,

    @{ Name ='Sizemb'

       Expression = {(([int]$\_.size)/1mb).tostring('N2')}},

       DaysSinceLastAccessed -AutoSize

1. Creating a monthly task in the task scheduler

$Date = Get-Date '04:20'

$NTHT = @{

  Time    = $Date

  Monthly = 1

}

$Task = New-FsrmScheduledTask @NTHT

$NRHT = @{

  Name             = 'Monthly Files by files group report'

  Namespace        = 'C:\'

  Schedule         = $Task

  ReportType       = 'FilesbyFileGroup'

  FileGroupINclude = 'Text Files'

  LargeFileMinimum = 25MB

}

New-FsrmStorageReport @NRHT | Out-Null

1. Getting details of the task

Get-ScheduledTask |

  Where-Object TaskName -Match 'Monthly' |

    Format-Table -AutoSize

1. Running the task now

Get-ScheduledTask -TaskName '\*Monthly\*' |

  Start-ScheduledTask

Get-ScheduledTask -TaskName '\*Monthly\*'

1. Viewing the report in the C:\StorageReports folder

$Path = 'C:\StorageReports\Scheduled'

$Rep = Get-ChildItem -Path $path\\*.html

$Rep

1. Viewing the report

Invoke-item -Path $Rep

## How it works...

In step 1, you create a new FSRM report to discover large files (over 10 MB in size) on the C:\ drive. The output from this step looks like this:

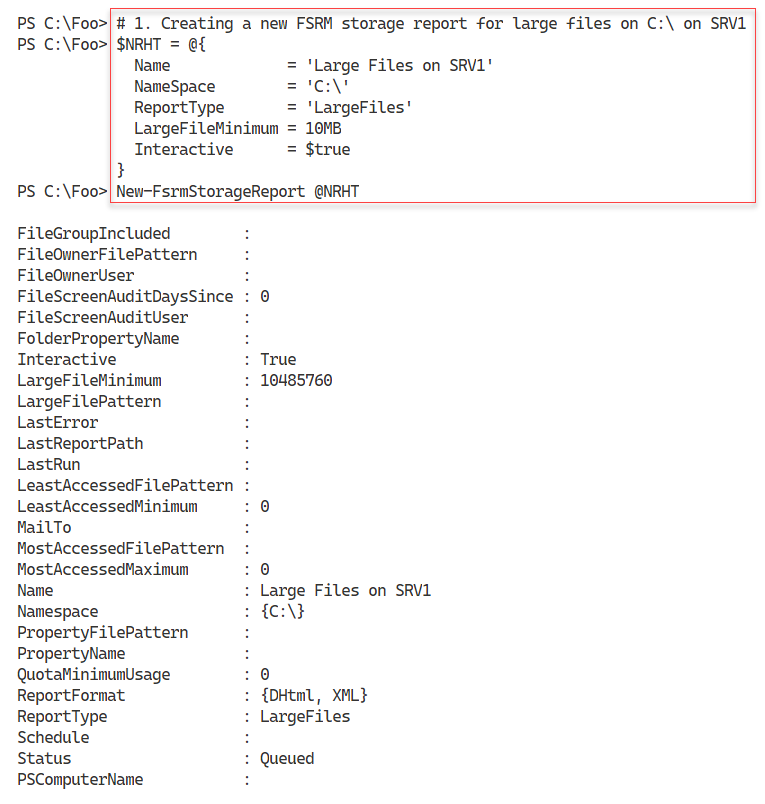


Figure 10.39: Creating a new FSRM storage report

**Insert image B42024\_10\_39.png**

In step 2, you view the available FSRM reports, with output like this:

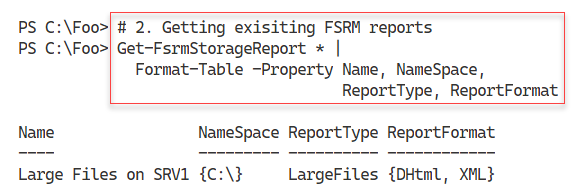


Figure 10.40: Viewing available FSRM storage reports

**Insert image B42024\_10\_40.png**

In step 3, you examine the reports that have completed and have output in the C:\StorageReports folder. The output looks li9ke this:

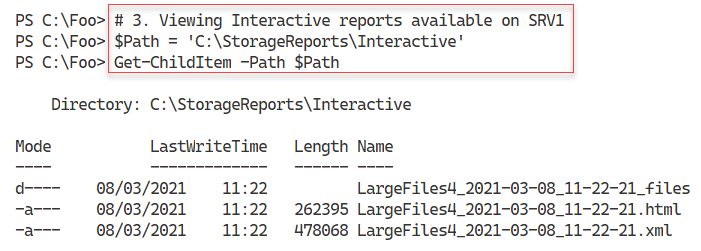


Figure 10.41: Viewing completed FSRM storage reports

**Insert image B42024\_10\_41.png**

In step 4, you examine the large file report in the browser, which looks like this:

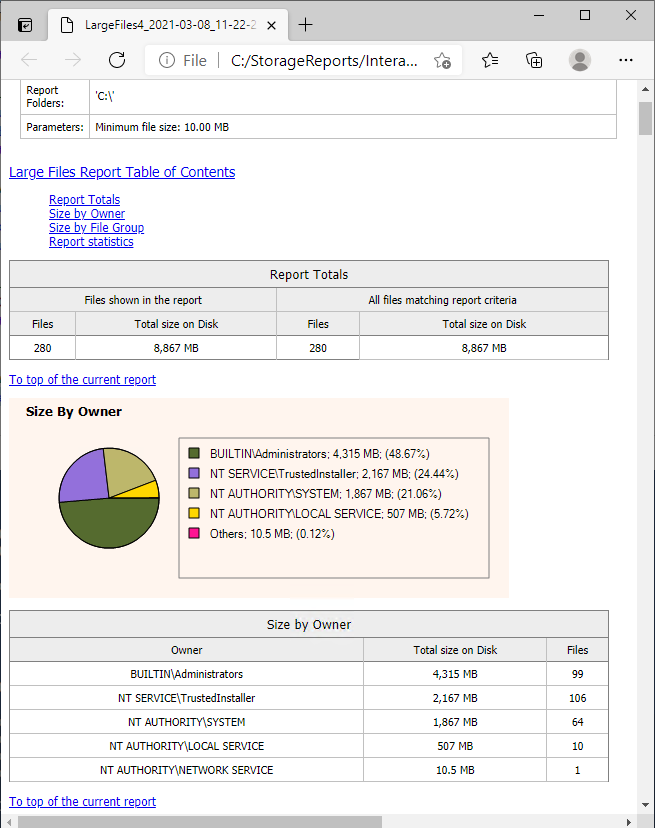


Figure 10.42: Viewing large file report from SRV1

**Insert image B42024\_10\_42.png**

In step 5, you extract the critical information from the report XML file and output it to the console. The output looks like this:

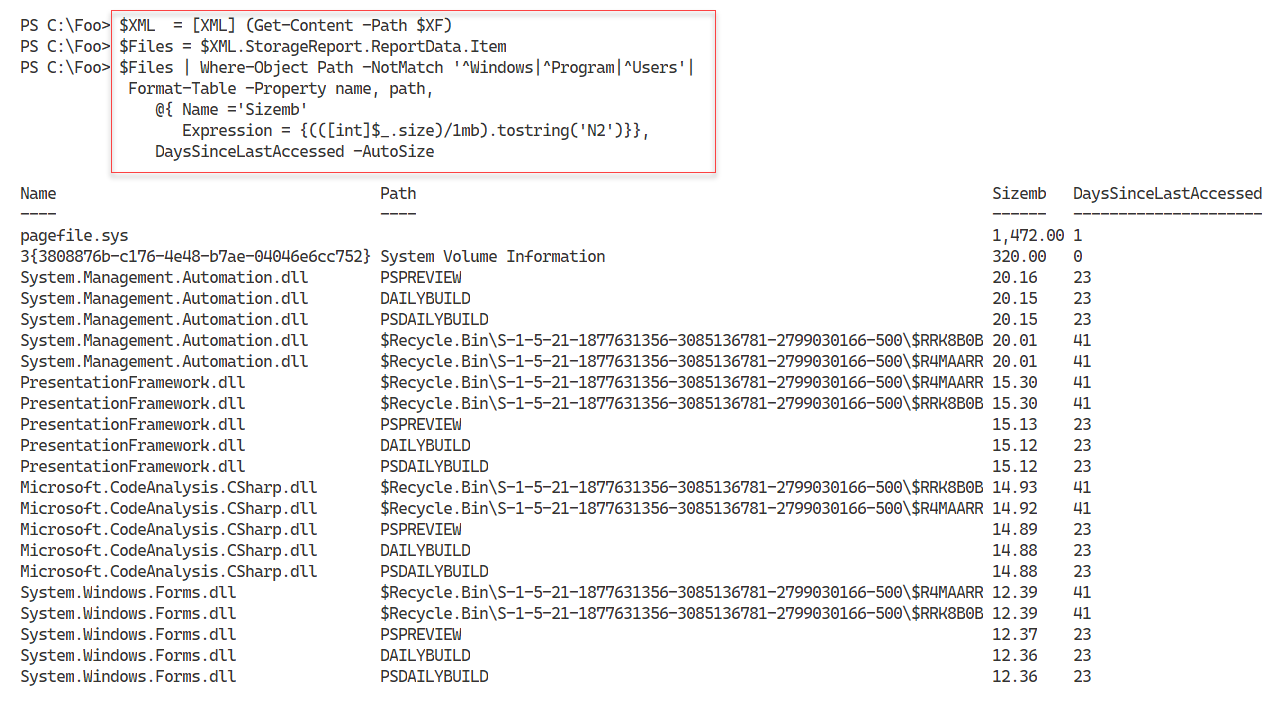


Figure 10.43: Viewing large file information

**Insert image B42024\_10\_43.png**

In step 6, you create a new scheduled task to run monthly. The task runs the FilesbyFileGroup report. This step creates no output.

In step 7, you examine the details of the scheduled task, with output like this:

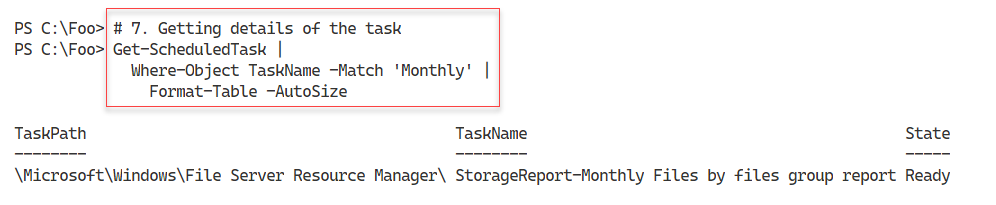


Figure 10.44: Viewing large file information

**Insert image B42024\_10\_44.png**

In step 8, you execute the scheduled task immediately, with output like this:

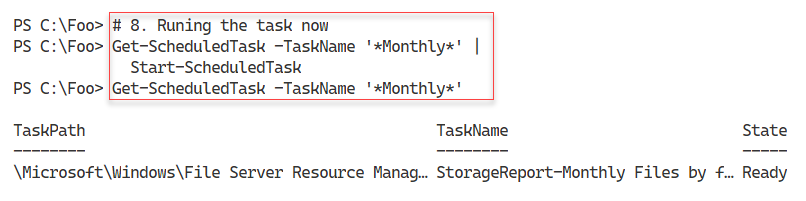


Figure 10.45: Executing the scheduled FSRM task

**Insert image B42024\_10\_45.png**

In step 9, after FSRM completes running the report, you view the report output like this:

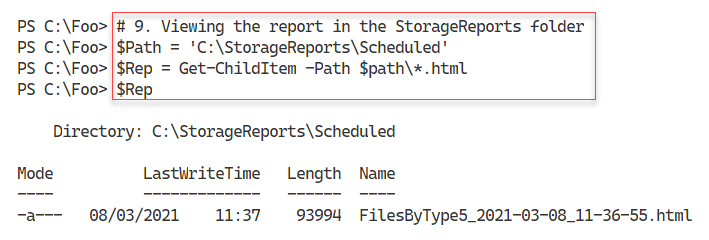


Figure 10.46: Viewing the report data

**Insert image B42024\_10\_46.png**

In the final step in this recipe, step 10, you view the report in the browser, with output like this:

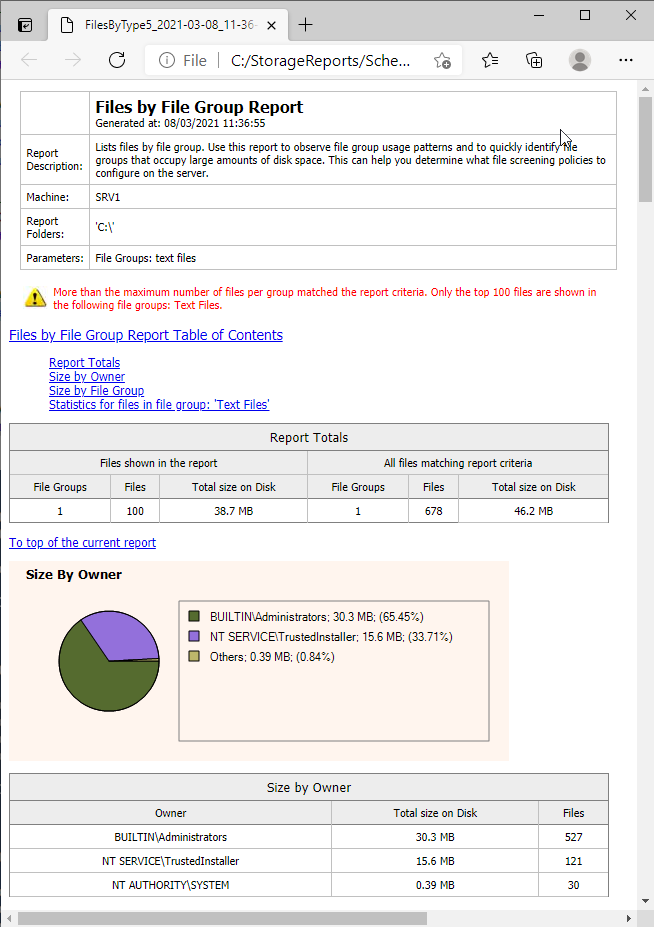


Figure 10.47: Viewing the report

**Insert image B42024\_10\_47.png**

## There's more...

In step 1, you create a new FSRM interactive report. FSRM starts running this command immediately. If you are not quick enough, you might see no output when you attempt to view the FSRM reports in step 2. This lack of output typically indicates that the report has completed, and you can view the report output, as you do in step 3.

As you can see from this recipe, FSRM creates report output both as HTML and XML. You can not change the HTML format – but it probably good enough for most uses. If you want a specific format or just some of the data, you get the same information from the XML and format it to suit your needs.

# Implementing FSRM File Screening

FSRM has a file screening option. This feature allows you to control the types of files you allow to be stored on your file server. You could, for example, define a file screen to prohibit a user from saving music files (files with the .MP3 or .FLAC extension) to be saved to your file server. With file screening, if a user attempts to save a file such as GD71-02-18.T09.FLAC, FSRM prevents the saving of the file.

To configure FSRM file screening, you need to specify the folder FSRM should protect and a file screen template that describes the characteristics of files that FSRM should block. FSRM comes with five built-in file screen templates. You can create additional templates to suit your requirements.

Each file screen template contains a set of file groups. Each file group defines a set of file extensions that FSRM can block. FSRM comes with 11 built-in file groups that cover common content types and can be updated and extended.

One built-in FSRM file group is Audio and Video Files. This group, for example, includes a wide variety of audio and video file extensions, including.AAC, .MP3, .FLAC, and more. Interestingly, this built-in file group does not block .SHN (Shorten) files. Shorten is another older lossless compression algorithm that some (legal) music trading circles still use. You could easily add this extension to the relevant file group, should you wish.

Note that file screening works solely based on file extensions. FSRM, for example, might block you from saving a file such as GD71-02-18.T09.FLAC. However, if you tried to sore this file as GD71-02-18.T09.CALF, FSRM would allow the file to be stored. The FSRM file screening does not examine the file to ascertain the actual file type. In most cases, file screening stops the more obvious policy infractions.

## Getting Ready

This recipe uses SRV1, a domain-joined host in the Reskit.Org domain, on which you have installed PowerShell 7 and VS Code. In the previous recipe, “Implementing FSRM Quotas”, you have installed FRSRM on SRV1.

## How to do it...

1. Examining the existing file groups

Get-FsrmFileGroup |

  Format-Table -Property Name, IncludePattern

1. Examining existing File Screen templates

Get-FsrmFileScreenTemplate |

  Format-Table -Property Name, IncludeGroup, Active

1. Creating a new folder

$Path = 'C:\FileScreen'

If (-Not (Test-Path $Path)) {

  New-Item -Path $Path -ItemType Directory  |

    Out-Null

}

1. Creating a new file screen

$FSHT =  @{

  Path         = $Path

  Description  = 'Block Executable Files'

  IncludeGroup = 'Executable Files'

}

New-FsrmFileScreen @FSHT

1. Testing file screen by copying notepad.exe

$FSTHT = @{

  Path        = "$Env:windir\notepad.exe"

  Destination = 'C:\FileScreen\notepad.exe'

}

Copy-Item  @FSTHT

1. Setting up Active Email Notification

$Body =

"[Source Io Owner] attempted to save an executable program to

[File Screen Path].

This is not allowed!

"

$FSRMA = @{

  Type             = 'Email'

  MailTo           = 'DoctorDNS@Gmail.Com'

  Subject          = 'Warning: attempted to save an executable file'

  Body             = $Body

  RunLimitInterval = 60

}

$Notification = New-FsrmAction @FSRMA

$FSFS = @{

  Path         = $Path

  Notification = $Notification

  IncludeGroup = 'Executable Files'

  Description  = 'Block any executable file'

  Active       = $true

}

Set-FsrmFileScreen @FSFS

1. Get-FSRM Notification Limits

Get-FsrmSetting |

  Format-List -Property "\*NotificationLimit"

1. Changing FSRM notification limits

$FSRMSHT = @{

  CommandNotificationLimit = 1

  EmailNotificationLimit   = 1

  EventNotificationLimit   = 1

  ReportNotificationLimit  = 1

}

Set-FsrmSetting @FSRMSHT

1. Re-testing the file screen to check the action

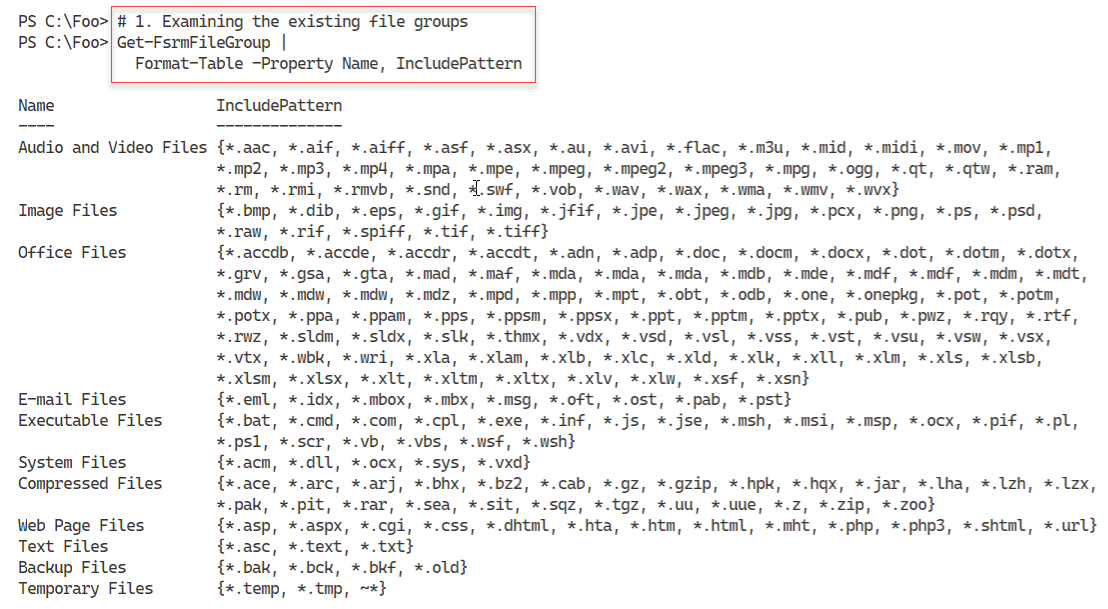
Copy-Item @FSTHT

1. Viewing file screen email

Use your email client to view the mail created by FSRM.

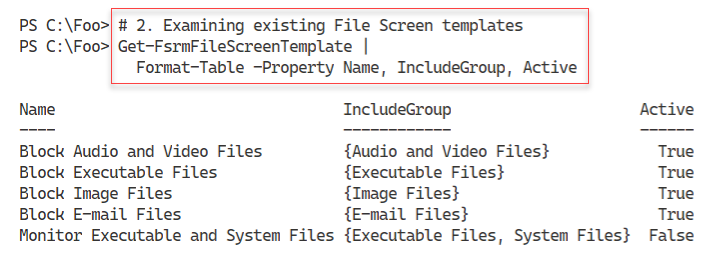
## How it works...

In step 1, you examine the initial set of FSRM file groups. The output looks like this:



1. Figure 10.48: Viewing the report
2. **Insert image B42024\_10\_48.png**

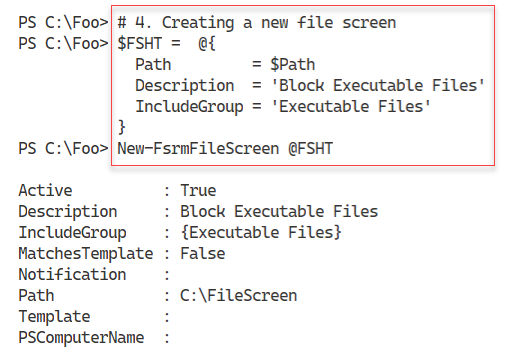
In step 2, you examine the built-in FSRM file screening templates. The output from this step is:



1. Figure 10.49: Viewing file screen templates

**Insert image B42024\_10\_49.png**

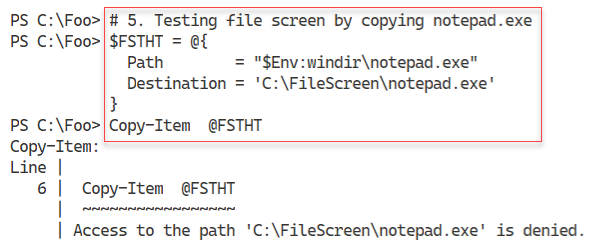
In step 3, you create a new folder for testing FSRM file screening, which produces no output. In step 4, you create a new FSRM file screen. The output from this step looks like this:



1. Figure 10.50: Creating a new file screen

**Insert image B42024\_10\_50.png**

To test the file screen, in step 5, you copy notepad.exe from the Windows folder to the file screen folder, with output like this:

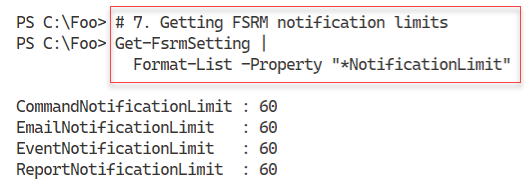


1. Figure 10.51: Testing a file screen

**Insert image B42024\_10\_51.png**

In step 6, you set up an active email notification to notify you any time a user attempts to save an executable file to the screened folder. This step creates no output.

In step 7, you examine the FSRM notification limits with output like this:

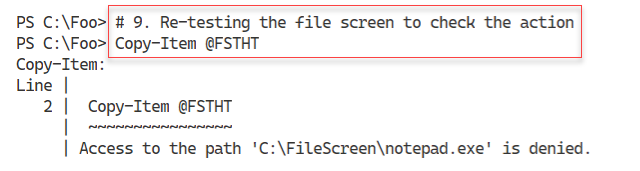


1. Figure 10.52: Examining FSRM notification limits

**Insert image B42024\_10\_52.png**

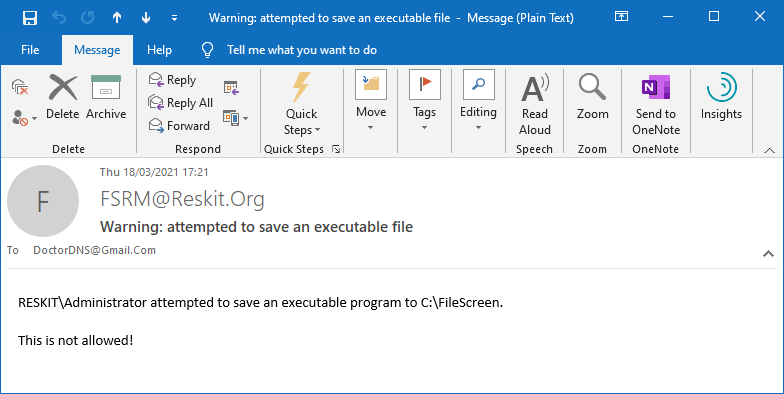
To speed up the creation of email notifications, in step 8, you reduce the email notification limits to one second. This step creates no console output.

In step 9, you test the updated file screen by re-attempting to save an executable to the screened folder, as before you get the following output:

1. 
2. Figure 10.53: Re-testing the file screen

**Insert image B42024\_10\_53.png**

Having set up an email notification for the file screen, in step 10, you view the generated email, which looks like this:



1. Figure 10.54: Re-testing the file screen

**Insert image B42024\_10\_54.png**

## There's more...