



9



Managing Printing

This chapter covers the following recipes:

- Installing and Sharing Printers
- Publishing a Printer to Active Directory
- Changing the Spooler Directory
- Changing Printer Drivers
- Printing a Test Page
- Managing Printer Security
- Creating a Printer Pool

Introduction

Printing is a feature that Microsoft has incorporated into various versions of the Windows operating system and has evolved over the years. Printer configuration and management in Windows Server 2022 hasn't changed much from earlier versions. Windows Server 2022 provides you with a printing platform for your organization and the ability to create print servers that you can share with users in your organization.

When printing in Windows, the physical device that renders output onto paper is known as a print device. A printer is a queue of documents to be printed on a print device. A print server can support multiple printers. Each printing device has an associated printer driver that converts your documents to a printed form on a given print device. Most printer drivers are vendor and printer model-dependent.

Some drivers come with Windows; others you need to obtain from a printer vendor. In some cases, these drivers are downloadable from the internet; in other cases, you may need to download and run a driver installation program to add the correct drivers to your print server.

Printers print to the print device by using a printer port (such as USB, parallel, or a network address). You define the printer port before creating a Windows printer for network printers. Microsoft hasn't changed the basic print architecture with Windows Server 2022. Windows Server 2012 introduced a new driver architecture that Windows Server 2022 supports. This driver model enables you to make use of two different driver types: printer class drivers and model-specific drivers. The former provides a single driver for various specific printing device models, whereas you use the model-specific drivers for just a single model. Increasingly, print device manufacturers are implementing more generic drivers that can simplify the organizational rollout of shared printing.

Another change in Windows Server 2012, carried into Windows Server 2022, is that you no longer have to use the print server to distribute printer drivers (which is especially relevant for network printers). You can use tools, such as the Microsoft Endpoint Manager or Group Policies, to distribute print drivers to clients in such cases.

This chapter covers installing, managing, and updating printers, print drivers, and printer ports on a Windows Server 2022 server. You may find that some of the administration tools used in this chapter aren't available on Windows Server Core systems. You must have the full Windows Server GUI (including the Desktop Experience) for any GUI utilities to enable full management.

In the *Installing and Sharing Printers* recipe, you install a printer and share it. In the *Publishing a Printer to Active Directory* recipe, you'll publish the printer to **Active Directory (AD)**, enabling users in the domain to search AD to find the printer. When you create a print server (by adding printer ports, printers, and so on), you may find that the default spool folder (underneath C:\Windows) may not be in an ideal location. When you print a document, Windows uses this spool folder to hold the temporary files required for the print device to render the document. Occasionally, this folder can get quite large, such as when a physical print device runs out of paper, and no one fixes the problem. In production, you might move the spool folder to a separate physical storage device, which can also improve the performance of a busy print server. In the *Changing the Spooler Directory* recipe, you change the default location for the printer spool.

In the *Changing Printer Drivers* recipe, you change the driver for the printer you created earlier. If you need to replace a specific print device with a different model, you may need to use other printer drivers. The printer vendor could also release updated drivers that you can deploy.

When you install a printer, a good step is to print a test page, as you can see in the *Printing a Test Page* recipe.

A Windows Printer can have an **Access Control List (ACL)** like a file or folder. You can also modify the ACL, as shown in the *Managing Printer Security* recipe. In most organizations, print devices deployed via a print server are shared resources. You see how to deploy a shared printer in the *Installing and Sharing Printers* recipe.

In Windows, a printer pool is a printer that has two or more associated printing devices. This means having two or more physical printers (print devices on separate ports) that users see as just a single printer. A printer pool could be useful when users create large numbers of printed documents. You can deploy multiple physical printers but expose these as a single Windows printer. In the *Creating a Printer Pool* recipe, you see how you can automate the creation of a printer pool using `rundll32.exe`.

The systems used in the chapter

This chapter uses a domain-joined server, PSRV, a member server in the `Reskit.org` domain on which you have installed PowerShell 7 (and VS Code). You also need the domain controller for the domain online as well, DC1, as follows:

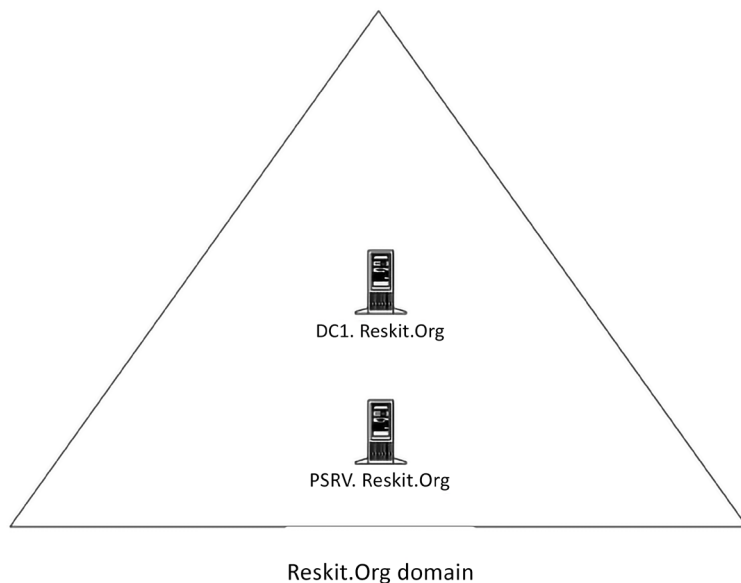


Figure 9.1: Hosts in use for this chapter

Installing and Sharing Printers

The first step in creating a print server for your organization involves installing the print server feature, printer drivers, and printer ports to the print server. Once you do that, you can create and share printers for others to access. In this recipe, you download and install two Xerox printer drivers. This recipe uses one of the drivers, while you use the second in the *Changing Printer Drivers* recipe. This internet download comes as a .zip archive that you must extract before using the drivers.

Note: if you're using this recipe to support other printer makes and models, you may need to make some changes. In some cases, such as with some Hewlett-Packard printers, the manufacturer designed the printer drivers to install them by running a downloadable executable. You must run the downloaded executable on each print server to add the necessary drivers. Thus, this recipe may not apply to all printing devices.

Getting ready

You run this recipe on PSRV after you have installed PowerShell 7.

How to do it...

1. Installing the Print-Server features on PSRV

```
Install-WindowsFeature -Name Print-Server,  
                        Print-Services,  
                        RSAT-Print-Services
```

2. Creating a folder for the Xerox printer drivers

```
$NewItemHT = @{  
    Path      = 'C:\Foo\Xerox'  
    ItemType  = 'Directory'  
    Force     = $true  
    ErrorAction = "Silentlycontinue"  
}  
New-Item @NewItemHT | Out-Null
```

3. Downloading printer drivers for the Xerox printers

```
$URL='http://download.support.xerox.com/pub/drivers/6510/' +  
    'drivers/win10x64/ar/6510_5.617.7.0_PCL6_x64.zip'  
$Target='C:\Foo\Xerox\Xdrivers.zip'
```

```
Start-BitsTransfer -Source $URL -Destination $Target
```

4. Expanding the .zip file

```
$Drivers = 'C:\Foo\Xerox\Drivers'  
Expand-Archive -Path $Target -DestinationPath $Drivers
```

5. Installing the drivers

```
$Model1 = 'Xerox Phaser 6510 PCL6'  
$P = 'C:\Foo\Xerox\Drivers\6510_5.617.7.0_PCL6_x64_Driver.inf\' +  
    'x3NSURX.inf'  
rundll32.exe printui.dll,PrintUIEntry /ia /m "$Model1" /f "$P"  
$Model2 = 'Xerox WorkCentre 6515 PCL6'  
rundll32.exe printui.dll,PrintUIEntry /ia /m "$Model2" /f "$P"
```

6. Adding a PrinterPort for a new printer

```
$PrintPortHT = @{  
    Name = 'SalesPP'  
    PrinterHostAddress = '10.10.10.61'  
}  
Add-PrinterPort @PrintPortHT
```

7. Adding the printer to PSRV

```
$PrinterHT = @{  
    Name = 'SalesPrinter1'  
    DriverName = $Model1  
    PortName = 'SalesPP'  
}  
Add-Printer @PrinterHT
```

8. Sharing the printer

```
Set-Printer -Name SalesPrinter1 -Shared $True
```

9. Reviewing what you have done

```
Get-PrinterPort -Name SalesPP |  
    Format-Table -Autosize -Property Name, Description,  
        PrinterHostAddress, PortNumber  
Get-PrinterDriver -Name xerox* |
```

```
Format-Table -Property Name, Manufacturer,
                                DriverVersion, PrinterEnvironment
Get-Printer -ComputerName PSRV -Name SalesPrinter1 |
Format-Table -Property Name, ComputerName,
                                Type, PortName, Location, Shared
```

10. Checking the status of the shared printer

```
net view \\PSRV
```

How it works...

In *step 1*, you install both the Print Server Windows feature and the printing RSAT, with output like this:

PS C:\Foo> # 1. Installing the Print-Server features on PSRV

PS C:\Foo> Install-WindowsFeature -Name Print-Server,

Print-Services,

RSAT-Print-Services

Success	Restart	Needed	Exit Code	Feature	Result
True	No		Success	{Print Server, Print and Document Services, ...	

Figure 9.2: Installing printer features to PSRV

In *step 2*, you create a new folder to hold the printer driver download. In *step 3*, you download the drivers (as a compressed .zip file) into the folder you just created. In *step 4*, you expand the .ZIP files, and in *step 5*, you install these printer drivers. In *step 6*, you add a new printer port to PSRV, and in *step 7*, you add a new printer using the printer port you just created and the Xerox printer drivers you downloaded. Finally, in *step 8*, you share the printer so other users can print to this printer. These seven steps produce no output.

In *step 9*, you examine the printer ports, printer drivers, and printers available on PSRV, with output like this:

```

PS C:\Foo> # 9. Reviewing what you have done
PS C:\Foo> Get-PrinterPort -Name SalesPP |
Format-Table -AutoSize -Property Name, Description,
PrinterHostAddress, PortNumber

```

Name	Description	PrinterHostAddress	PortNumber
SalesPP	Standard TCP/IP Port	10.10.10.61	9100

```

PS C:\Foo> Get-PrinterDriver -Name xerox* |
Format-Table -Property Name, Manufacturer,
DriverVersion, PrinterEnvironment

```

Name	Manufacturer	DriverVersion	PrinterEnvironment
Xerox Phaser 6510 PCL6	Xerox	1581047950660861952	Windows x64
Xerox WorkCentre 6515 PCL6	Xerox	1581047950660861952	Windows x64

```

PS C:\Foo> Get-Printer -ComputerName PSRV -Name SalesPrinter1 |
Format-Table -Property Name, ComputerName,
Type, PortName, Location, Shared

```

Name	ComputerName	Type	PortName	Location	Shared
SalesPrinter1	PSRV	Local	SalesPP		True

Figure 9.3: Viewing printer components on PSRV

In *step 10*, you use the `net.exe` command to view the shared printers that SRV1 provides, with output like this:

```

PS C:\Foo> # 10. Checking the status of the shared printer
PS C:\Foo> net view \\PSRV
Shared resources at \\PSRV

```

Share name	Type	Used as	Comment
SalesPrinter1	Print		SalesPrinter1

The command completed successfully.

Figure 9.4: Viewing the shared printer on PSRV

There's more...

In this recipe, you create a printer on PSRV based on Xerox printers. The fact that you may not have this printer model in your environment means you can't physically print to such a print device, but you can set up the printer as shown in the recipe.

In *step 5*, you use `PrintUI.DLL` and `RunDLL32.EXE`. The former, `PrintUI.DLL`, is a library of printer management functionalities. If you use the Printer Management GUI tool to manage printers, the GUI calls this DLL to perform your chosen action. Since Microsoft designed and built this DLL to support the Windows printer GUI, the DLL can create additional dialog boxes, which are not useful for automation. You can rely on the help information generated to resolve any problems.

In *step 10*, you use the `net .exe` command to view the shared printer on PSRV. There is currently no cmdlet that provides this information.

Publishing a Printer to Active Directory

After you create and share a printer (as shown in the previous recipe), you can publish it to Active Directory. You publish the printer you created in the last recipe and examine the results in this recipe. When you publish a printer, you can also specify a physical location for it. Your users can search for published printers based on location and capabilities (such as color printers).

Getting ready

Before running this recipe, you must set up the PSRV printer server (you did this in the *Installing and Sharing Printers* recipe). Additionally, you need `SalesPrinter1` created.

How to do it...

1. Getting the printer to publish and store the returned object in `$Printer`

```
$Printer = Get-Printer -Name SalesPrinter1
```

2. Viewing the printer details

```
$Printer | Format-Table -Property Name, Published
```

3. Publishing and sharing the printer to AD

```
$Printer | Set-Printer -Location '10th floor 10E4'  
$Printer | Set-Printer -Shared $true -Published $true
```

4. Viewing the updated publication status

```
Get-Printer -Name SalesPrinter1 |  
Format-Table -Property Name, Location, Drivename, Published
```


How it works...

In *step 1*, you obtain details of the printer you wish to share and store this into the variable `$Printer`, producing no output. In *step 2*, you examine the printer details contained in `$Printer` with output like this:

```
PS C:\Foo> # 2. Viewing the printer details
PS C:\Foo> $Printer | Format-Table -Property Name, Published

Name                Published
-----
SalesPrinter1       False
```

Figure 9.5: Viewing the SalesPrinter1 printer on PSRV

In *step 3*, you publish the printer to the Reskit.Org AD, and you share the printer explicitly. This step produces no output. In *step 4*, you use `Get-Printer` to review the publication status of the printer, with output like this:

```
PS C:\Foo> # 4. Viewing the updated publication status
PS C:\Foo> Get-Printer -Name SalesPrinter1 |
             Format-Table -Property Name, Location, Drivename, Published

Name                Location          Drivename          Published
-----
SalesPrinter1 10th floor 10E4 Xerox Phaser 6510 PCL6      True
```

Figure 9.6: Viewing the SalesPrinter1 printer details

There’s more...

Publishing a printer to AD allows users to locate printers near them using the **Add Printer** dialog to search for published printers. For example, if you log on to any computer in the domain, you can get to this dialog box by clicking **Start | Settings | Devices | Printers & scanners** to bring up the **Add printers & scanners** dialog. From this dialog box, click **Add a printer or scanner**.

Wait until the search is complete, then click on **The printer that I want isn't listed**, which brings up the **Add Printer** dialog, like this:

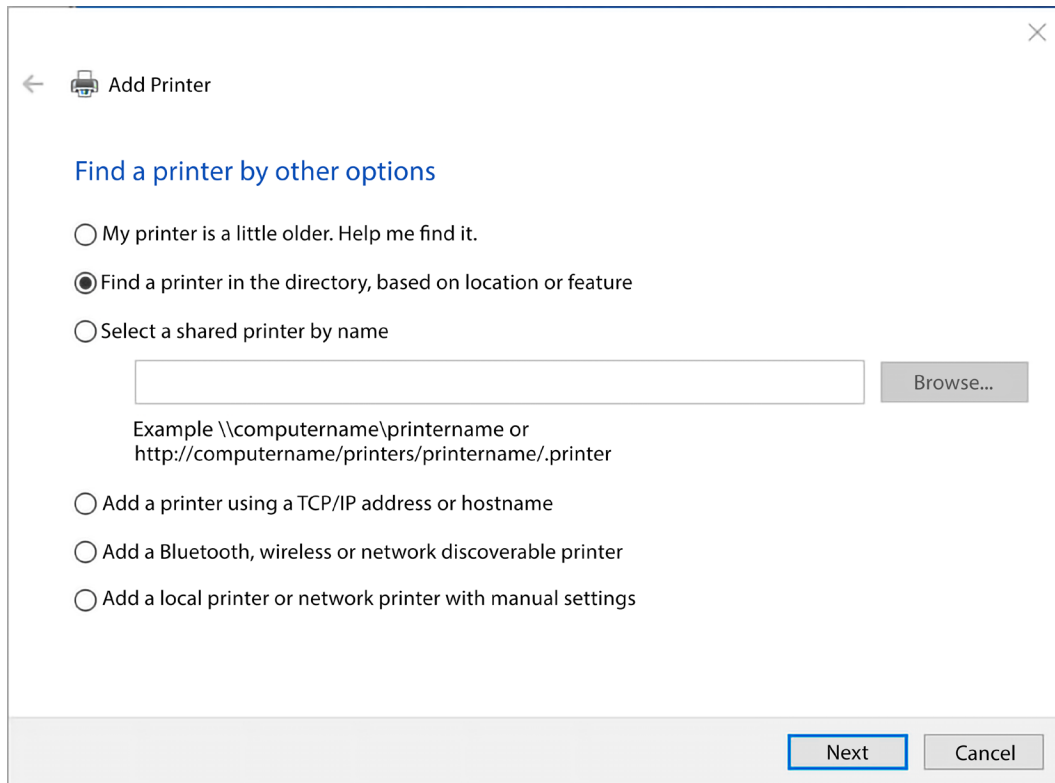


Figure 9.7: Using the Add Printer dialog

From this dialog box, click on **Next** to bring up the **Find Printers** dialog, which looks like this:

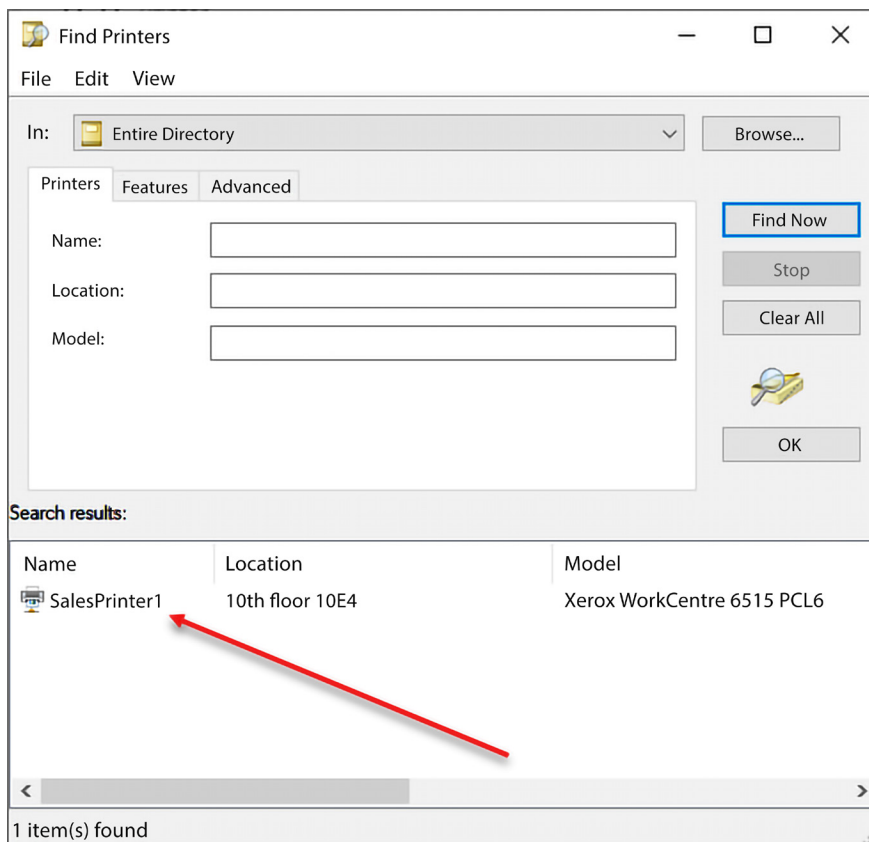


Figure 9.8: Using the Find Printers dialog

In larger organizations, publishing printers to the AD can be very useful in helping users find the corporate printers available.

Changing the Spooler Directory

During the printing process, the Windows printer spooler in Windows uses an on-disk folder to hold the temporary files the printing process creates. If multiple users each print large documents to a single printer, the print queue, and the temporary folder can get quite large. By default, this folder is `C:\Windows\System32\spool\PRINTERS`. You may wish to change the default spool folder on a separate physical storage device for a busy print server with multiple printers. Also, consider ensuring the volume is fail-safe (e.g. using hardware or software RAID).

Getting ready

Before running this recipe, you must set up the PSRV printer server (you did this in the *Installing and Sharing Printers* recipe). Additionally, you need SalesPrinter1 created.

How to do it...

1. Loading the System.Printing namespace and classes

```
Add-Type -AssemblyName System.Printing
```

2. Defining the required permissions

```
$Permissions =  
[System.Printing.PrintSystemDesiredAccess]::AdministrateServer
```

3. Creating a PrintServer object with the required permissions

```
$NewObjHT = @{  
    TypeName      = 'System.Printing.PrintServer'  
    ArgumentList = $Permissions  
}  
$PrintServer = New-Object @NewObjHT
```

4. Displaying print server properties

```
$PrintServer
```

5. Observing the default spool folder

```
"The default spool folder is: [{0}]" -f $PrintServer.  
DefaultSpoolDirectory
```

6. Creating a new spool folder

```
$NewItemHT = @{  
    Path          = 'C:\SpoolPath'  
    ItemType      = 'Directory'  
    Force         = $true  
    ErrorAction   = 'SilentlyContinue'  
}  
New-Item @NewItemHT | Out-Null
```

7. Updating the default spool folder path

```
$NewPath = 'C:\SpoolPath'
$PrintServer.DefaultSpoolDirectory = $NewPath
```

8. Committing the change

```
$PrintServer.Commit()
```

9. Restarting the spooler to accept the new folder

```
Restart-Service -Name Spooler
```

10. Verifying the new spooler folder

```
New-Object -TypeName System.Printing.PrintServer |
    Format-Table -Property Name,
                    DefaultSpoolDirectory
```

Another way to set the spooler directory is by directly editing the registry as follows:

11. Stopping the spooler service

```
Stop-Service -Name Spooler
```

12. Creating a new spool directory

```
$SpoolFolder2 = 'C:\SpoolViaRegistry'
$NewItemHT2 = @{
    Path      = $SpoolFolder2
    Itemtype   = 'Directory'
    ErrorAction = 'SilentlyContinue'
}
New-Item @NewItemHT2 | Out-Null
```

13. Creating the spooler folder and configuring it in the registry

```
$RegistryPath = 'HKLM:\SYSTEM\CurrentControlSet\Control\' +
                'Print\Printers'
$ItemPropHT = @{
    Path    = $RegistryPath
    Name    = 'DefaultSpoolDirectory'
    Value   = $SPL
}
Set-ItemProperty @ItemPropHT
```

14. Restarting the spooler service

```
Start-Service -Name Spooler
```

15. Viewing the results

```
New-Object -TypeName System.Printing.PrintServer |  
Format-Table -Property Name, DefaultSpoolDirectory
```

How it works...

In *step 1*, you load the System.Printing namespace, which produces no output. In *step 2*, you create a variable holding the desired access to the printer. In *step 3*, you create a PrintServer object with the appropriate permissions. Then, in *step 4*, you examine the default spooler folder for a newly installed Windows Server 2022 host, which produces output like this:

```
PS C:\Foo> # 4. Displaying print server properties  
PS C:\Foo> $PrintServer  
  
Name : \\COOKHAM216  
SubSystemVersion : 0  
RestartJobOnPoolEnabled : True  
RestartJobOnPoolTimeout : 600  
MinorVersion : 0  
MajorVersion : 3  
EventLog : LogPrintingErrorEvents  
NetPopup : False  
BeepEnabled : False  
DefaultSchedulerPriority : Normal  
SchedulerPriority : Normal  
DefaultPortThreadPriority : Normal  
PortThreadPriority : Normal  
DefaultSpoolDirectory : C:\Windows\system32\spool\PRINTERS  
Parent :  
PropertiesCollection : {EventLog, RestartJobOnPoolTimeout, SchedulerPriority,  
                        MinorVersion, Name, MajorVersion, DefaultSpoolDirectory,  
                        DefaultPortThreadPriority, DefaultSchedulerPriority,  
                        PortThreadPriority, BeepEnabled, RestartJobOnPoolEnabled, NetPopup}
```

Figure 9.9: Examining the default spool folder

In *step 5*, you create a new folder on PSRV to serve as your print server's spool folder. In *step 6*, you update the printer spool folder path. In *step 7*, you commit this change. These three steps produce no console output. In *step 8*, you restart the spooler service. Depending on the speed of your system, you may see output as follows:

```
PS C:\Foo> # 9. Restarting the Spooler to accept the new folder  
PS C:\Foo> Restart-Service -Name Spooler  
WARNING: Waiting for service 'Print Spooler (Spooler)' to start...
```

Figure 9.10: Restarting the print spooler service

In *step 10*, you view the printer details with output like this:

```
PS C:\Foo> # 10. Verifying the new spooler folder
PS C:\Foo> New-Object -TypeName System.Printing.PrintServer |
            Format-Table -Property Name,
                        DefaultSpoolDirectory

Name      DefaultSpoolDirectory
-----
\\PSRV C:\SpoolPath
```

Figure 9.11: Examining the default spool folder

From *step 1* through to *step 9*, you set and validate the spool folder using .NET. There is another separate way to change the spool folder that involves using WMI, which we will use next.

In *step 11*, you stop the spooler service. In *step 12*, you create a new folder, and in *step 13*, you configure the necessary registry settings to contain the path to the new spool folder. Then in *step 14*, you restart the spooler. These four steps produce no console output. With the changes made to the spool folder within the registry, in *step 15*, you can view the updated spool folder with output like this:

```
PS C:\Foo> # 15. Viewing the results
PS C:\Foo> New-Object -TypeName System.Printing.PrintServer |
            Format-Table -Property Name, DefaultSpoolDirectory

Name      DefaultSpoolDirectory
-----
\\PSRV C:\WINDOWS\system32\spool\PRINTERS
```

Figure 9.12: Examining the updated spool folder

There's more...

For most organizations with a larger number of shared printers, configuring the print server to use another folder for spooling is a good idea. If possible, use a large separate storage volume to avoid the risk of the spool folder filling up. And ensure that the disk volume is fault tolerant (i.e., using software or hardware RAID).

You used two mechanisms in this recipe to change the spooler folder. One uses a .NET object, while the other involves directly editing the registry.

Many of the steps in this recipe produce no output. This lack of output is normal when dealing directly with .NET classes and methods and editing the registry.

In *step 1*, you add the `.NET System.Printing` namespace DLL into the current PowerShell session. This namespace DLL contains the classes involved with managing printing. When PowerShell loads, it does not load this DLL, so you have to load it, as shown in this step, before you can use the classes, types, etc. inside the namespace. For details on what is inside this namespace, see <https://docs.microsoft.com/dotnet/api/system.printing>.

Changing Printer Drivers

It may be necessary to change the printer driver for a Windows printer. For example, you might replace an existing print device with a new or different model, or the printer vendor may have released a new driver for your printer. In these cases, you want the printer's name to remain the same, but you need to update the printer's print driver. In the *Installing and Sharing Printers* recipe, you downloaded and installed two Xerox printer drivers. You used the first driver, Xerox Phaser 6510 PCL6, when you defined the `SalesPrinter1` printer.

In this recipe, you change the driver for the printer and use the other previously installed driver, Xerox Phaser 6515 PCL6.

Keeping the printer name (and port) unchanged makes swapping a printer relatively straightforward. In this recipe, you change the printer driver while leaving the printer's name and the printer's port (including the printer's IP address and port number) unchanged. This scenario might occur if you replace the physical printing device with a newer model, or even change the printer type/manufacturer.

Getting ready

You run this recipe on PSRV after you have installed PowerShell 7.

How to do it...

1. Adding the print driver for the new printing device

```
$Model2 = 'Xerox WorkCentre 6515 PCL6'  
Add-PrinterDriver -Name $Model2
```

2. Viewing loaded printer drivers

```
Get-PrinterDriver
```

3. Getting the Sales Group printer object and store it in `$Printer`

```
$PrinterName = 'SalesPrinter1'  
$Printer = Get-Printer -Name $PrinterName
```


- 4. Updating the driver using the Set-Printer cmdlet

```
$Printer | Set-Printer -DriverName $Model2
```

- 5. Observing the updated printer driver

```
Get-Printer -Name $PrinterName |  
Format-Table -Property Name, DriverName, PortName,  
Published, Shared
```

How it works...

In *step 1*, you use the Add-PrinterDriver cmdlet to add the printer driver for the printer. In *step 2*, you view the loaded print drivers without output like this:

PS C:\Foo> # 2. Viewing loaded printer drivers:
PS C:\Foo> Get-PrinterDriver

Name	PrinterEnvironment	MajorVersion	Manufacturer
Microsoft XPS Document Writer v4	Windows x64	4	Microsoft
Microsoft Software Printer Driver	Windows x64	4	Microsoft
Microsoft Print To PDF	Windows x64	4	Microsoft
Xerox WorkCentre 6515 PCL6	Windows x64	3	Xerox
Xerox Phaser 6510 PCL6	Windows x64	3	Xerox
Remote Desktop Easy Print	Windows x64	3	Microsoft
Microsoft Shared Fax Driver	Windows x64	3	Microsoft
Microsoft enhanced Point and Print...	Windows x64	3	Microsoft
Microsoft enhanced Point and Print...	Windows NT x86	3	Microsoft

Figure 9.13: Viewing loaded printer drivers

In *step 3*, you obtain the printer details, and in *step 4*, you update the printer to use the updated driver. These steps produce no console output.

In *step 5*, you use the Get-Printer cmdlet to observe that you have installed the updated driver for the SalesPrinter1 printer, which looks like this:

PS C:\Foo> Get-Printer -Name \$PrinterName |
Format-Table -Property Name, DriverName, PortName,
Published, Shared

Name	DriverName	PortName	Published	Shared
SalesPrinter1	Xerox WorkCentre 6515 PCL6	SalesPP	True	True

Figure 9.14: Viewing the updated printer driver for SalesPrinter1

As you see in this recipe, changing a printer driver is straightforward – after you install a new printer driver, you use `Set-Printer` to inform Windows which driver it should use when printing to the printer.

Printing a Test Page

There are occasions when you may wish to print a test page on a printer – for example, after you change the toner or printer ink on a physical printer or after changing the print driver (as shown in the *Changing Printer Drivers* recipe). In those cases, the test page helps you to ensure that the printer is working properly.

Getting ready

This recipe uses the PSRV print server that you set up in the *Installing and Sharing Printers* recipe.

How to do it...

1. Getting the printer objects from WMI

```
$Printers = Get-CimInstance -ClassName Win32_Printer
```

2. Displaying the number of printers defined on PSRV

```
'{0} Printers defined on this system' -f $Printers.Count
```

3. Getting the Sales Group printer WMI object

```
$Printer = $Printers |  
    Where-Object Name -eq 'SalesPrinter1'
```

4. Displaying the printer's details

```
$Printer | Format-Table -AutoSize
```

5. Printing a test page

```
Invoke-CimMethod -InputObject $Printer -MethodName PrintTestPage
```

6. Checking on test page print job

```
Get-PrintJob -PrinterName SalesPrinter1
```

How it works...

In *step 1*, you get details of the printers installed on PSRV using WMI and store them in the variable `$Printers`, producing no output. In *step 2*, you display how many printers you have are defined on your printer server, with output like this:

```
PS C:\Foo> # 2. Displaying the number of printers defined on PSRV
PS C:\Foo> '{0} Printers defined on this system' -f $Printers.Count
7 Printers defined on this system
```

Figure 9.15: Viewing printers on PSRV

In *step 3*, you get the specific WMI instance for the `SalesPrinter1` printer, creating no console output. In *step 4*, you view the details of this printer, with output like this:

```
PS C:\Foo> # 4. Displaying the printer's details
PS C:\Foo> $Printer | Format-Table -AutoSize
```

Name	ShareName	SystemName	PrinterState	PrinterStatus	Location
SalesPrinter1	SalesPrinter1	PSRV	0	3	10th floor 10E4

Figure 9.16: Viewing details of `SalesPrinter1`

In *step 5*, you use `Invoke-CimMethod` to run the `PrintTestPage()` method on the printer, generating output like this:

```
PS C:\Foo> # 5. Printing a test page
PS C:\Foo> Invoke-CimMethod -InputObject $Printer -MethodName PrintTestPage
```

ReturnValue	PSComputerName
0	

Figure 9.17: Invoking the `PrintTestPage` method

In the final step in this recipe, *step 6*, you view the print jobs on the `SalesPrinter1` printer, where you can see your test page output generated in the previous step. The output from this step looks like this:

```
PS C:\Foo> # 6. Checking on test page print job
PS C:\Foo> Get-PrintJob -PrinterName SalesPrinter1
```

Id	ComputerName	PrinterName	DocumentName	SubmittedTime	JobStatus
2		SalesPrinter1	Test Page	03/09/2022 13:16:18	Printing, Reta...

Figure 9.18: Viewing printer jobs to observe the test page

There's more...

In this recipe, you used WMI to create a test page on the SalesPrinter1 printer. As you saw in *step 6*, the printer's queue has the print job. In theory, the document should appear on the print device immediately. If you do not get any physical output, you must troubleshoot your printer infrastructure: is the printer turned on, is the network cable plugged in and working, does the printer have the correct IP address, etc.?

Managing Printer Security

Every Windows printer has a discretionary **ACL**. The ACL contains one or more **Access Control Entries (ACEs)**. Each ACE defines a specific permission for some particular group or user. You could define a group (such as SalesAdmins) and permit that group to manage documents while you give another group (such as Sales) access to print to the printer.

By default, when you create a printer, Windows adds some ACEs to the printer's ACL. The default ACL includes permitting the Everyone group to print to the printer. For some printers, this may not be appropriate. For this reason, you may need to adjust the ACL, as shown in this recipe.

The PrintManagement module contains several cmdlets that help you manage the printers. However, there are no cmdlets for managing ACLs on printers. You can always use .NET directly to manage the ACL or use third-party scripts that do the job for you. But this code can be complex (and easy to mess up). Make sure you test any recipes that modify printer ACLs very carefully. Always have a way to reset any ACL back to defaults should you make a mistake and need to start again to define the printer ACL. And as ever, you can always manage the ACL using the GUI if you need to!

Getting ready

Run this recipe on the PSRV printer server after you have installed and configured the SalesPrinter1 printer. This recipe uses the SalesPrinter1 printer and creates two new groups, Sales and SalesAdmin, in the AD.

How to do it...

1. Setting up AD for this recipe

```
$ScriptBlock = {  
    # 1.1 Creating Sales OU  
    $OUHT = @(  
        Name = 'Sales'
```

```

        Path = 'DC=Reskit,DC=Org'
    }
    New-ADOrganizationalUnit @OUHT
    # 1.2 Creating Sales Group
    $G1HT = @{
        Name          = 'SalesGroup'
        GroupScope    = 'Universal'
        Path          = 'OU=Sales,DC=Reskit,DC=Org'
    }
    New-ADGroup @G1HT
    # 1.3 Creating SalesAdmin Group
    $G2HT = @{
        Name          = 'SalesAdmins'
        GroupScope    = 'Universal'
        Path          = 'OU=Sales,DC=Reskit,DC=Org'
    }
    New-ADGroup @G2HT
}
# 1.4 Running Script block on DC1
Invoke-Command -ComputerName DC1 -ScriptBlock $ScripBlock

```

2. Getting the group to allow access

```

$GroupHT1 = @{
    Typename      = 'Security.Principal.NTAccount'
    Argumentlist  = 'SalesGroup'
}
$SalesGroup = New-Object @GroupHT1
$GroupHT2 = @{
    Typename      = 'Security.Principal.NTAccount'
    Argumentlist  = 'SalesAdmins'
}
$SalesAdminGroup = New-Object @GroupHT2

```

3. Getting the group SIDs

```

$SalesGroupSid =
    $SalesGroup.Translate([Security.Principal.SecurityIdentifier]).
    Value

```

```
$SalesAdminGroupSid =  
    $SalesAdminGroup.Translate(  
        [Security.Principal.SecurityIdentifier]).Value
```

4. Defining the SDDL for this printer

```
$SDDL = 'O:BAG:DUD:PAI(A;OICI;FA;;;DA)' +  
        "(A;OICI;0x3D8F8;;; $SalesGroupSid)" +  
        "(A;;LCSWSDRCWDWO;;; $SalesAdminGroupSid)"
```

5. Getting the Sales Group printer object

```
$SGPrinter = Get-Printer -Name SalesPrinter1 -Full
```

6. Setting the permissions

```
$SGPrinter | Set-Printer -Permission $SDDL
```

7. Viewing the permissions from the GUI.

Check the printer's security settings from the **Settings** applet in Windows.

How it works...

In *step 1*, you create a new OU in the Reskit.Org domain (Sales), then create two new Universal security groups (SalesGroup and SalesAdmins).

In *step 2*, you create two `Security.Principal.NTAccount` objects with the properties for the two security groups. In *step 3*, you use these two new objects to retrieve the **Security IDs (SIDs)** for each of the groups. In *step 4*, you create a **Security Descriptor Description Language (SDDL)** permission set.

In *step 5*, you use the `Get-Printer` to return a WMI object that describes the printer. Then in *step 6*, you use `Set-Printer` to set the ACL for your printer. These first six steps produce no console output.

In *step 7*, you observe the ACL by using the Windows Printer GUI. The output looks like this:

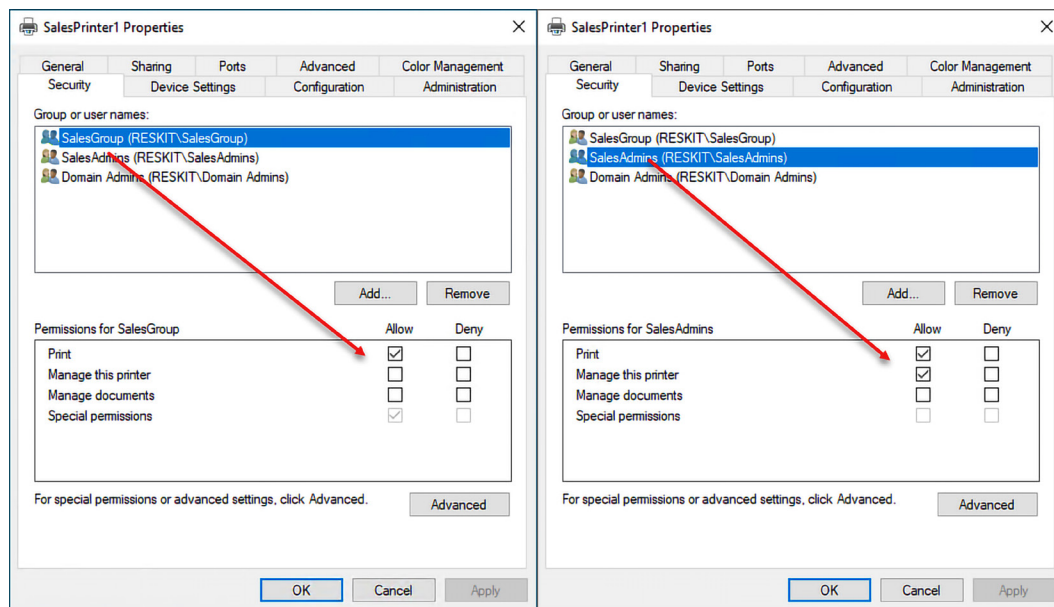


Figure 9.19: Viewing printer security settings

There's more...

In this recipe, you update the ACL for the Sales Group printer SalesPrinter1. The recipe uses a .NET object to obtain the SIDs for two security groups. Then you hand-construct the SDDL and apply it to the printer.

Unlike NTFS, there are no third-party printer ACL management tools readily available to simplify the setting of ACLs. SDDL is the default mechanism, but it is not always straightforward. For some details on SDDL, see <http://3.95.189.156.xip.io/blog/an-sddl-primer/>.

Creating a Printer Pool

Windows allows you to create a *printer pool*, a printer with two or more print devices (each with a separate printer port). With a printer pool, Windows sends a given print job to any of the pool's printers. This feature is helpful in environments where users print large numbers of documents and need the speed additional printers can provide, without asking the user to choose the specific print device.

There are no PowerShell cmdlets to enable you to create a printer pool. Also, WMI does not provide a mechanism to create a printer pool. As with other recipes in this chapter, you use PrintUI.dll and RunDLL32 to deploy your printer pool. This recipe is another example of utilizing older console applications to achieve your objective.

Getting ready

You run this recipe on PSRV, on which you have set up a new printer, SalesPrinter1.

How to do it...

1. Adding a port for the printer

```
$P = 'SalesPP2' # new printer port name
Add-PrinterPort -Name $P -PrinterHostAddress 10.10.10.62
```

2. Creating the printer pool for SalesPrinter1

```
$Printer = 'SalesPrinter1'
$P1      = 'SalesPP'   # First printer port
$P2      = 'SalesPP2'  # Second printer port
rundll32.exe printui.dll,PrintUIEntry /Xs /n $Printer Portname
"$P1,$P2"
```

3. Viewing the printer pool

```
Get-Printer $Printer |
Format-Table -Property Name, Type, DriverName, PortName -AutoSize
```

How it works...

In *step 1*, you add a new printer port, SalesPP2. In *step 2*, you create a new printer pool by using printui.dll and setting two printer ports. In the final step, *step 3*, you view the output to confirm you have set up a printer pool of two print devices/ports. The output of this final step looks like this:

```
PS C:\Foo> Get-Printer $Printer |
            Format-Table -Property Name, Type, DriverName, PortName -AutoSize
```

Name	Type	DriverName	PortName
SalesPrinter1	Local	Xerox WorkCentre 6515 PCL6	SalesPP,SalesPP2

Figure 9.20: Viewing the printer pool

There's more...

In *step 3*, you use the `Get-Printer` command to retrieve details about the printer to verify you have set up a printer pool. You can also view this pool using the printer GUI, which looks like this:

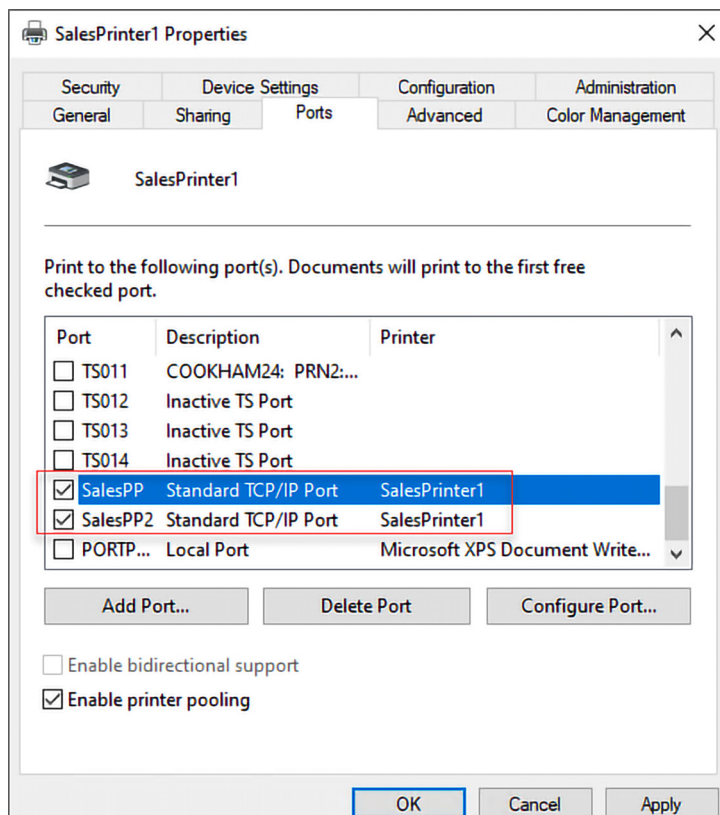


Figure 9.21: Viewing the printer pool from the GUI

Join our community on Discord

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

<https://packt.link/SecNet>

