11

Managing Printing

In this chapter, we cover the following recipes:

* Installing and sharing printers
* Changing the spooler directory
* Changing printer drivers
* Printing a test page
* Managing printer security
* Creating a printer pool

# Introduction

Printing is a feature that has been incorporated into various versions of the Windows operating system, and has evolved over the years. Printer configuration and management in Windows Server 2019 hasn't changed much from earlier versions, and provides you with 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 for a print device. A print server can support multiple printers. Each printing device has an associated printer driver that converts your documents to the printed form on a given print device. Some drivers come with Windows—others you need to obtain from the 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 network). For network printers, you need to define the port before you can create a Windows printer. Microsoft hasn't changed the basic print architecture with Windows Server 2019. Windows Server 2012 introduced a new driver architecture that Windows Server 2019 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 a variety of specific printing device models, whereas the latter is used 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 use the print server to distribute printer drivers (which is especially relevant for network printers). You can use tools, such as the System Center Configuration 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. To enable full management, you need to have the full GUI (including the Desktop Experience) for any GUI utilities.

In the *Installing and sharing printers* recipe, you install a printer and share it for others to use. In the *Publishing a printer* 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 (adding printer ports, printers, and so on), the default spool folder (underneath C:\Windows) may not be in an ideal location. In the *Changing the spool directory* recipe, you change the default location for the printer spool.  
Sometimes, a printer can have an associated print device swapped for a different printer model. In the *Changing printer drivers* recipe, you change the driver for the printer you created earlier. A useful troubleshooting step when working with printers is to print a test page, as you can see in the *Printing a test page* recipe.

Printers, like files, can have Access Control Lists (ACL) to specify who can use the printer. In the *Reporting on printer security* recipe, you report on the access enabled for a printer. You can also modify the ACL, as shown in the *Modifying printer* security recipe. In many organizations, print devices are a shared resource. In the *Deploying shared printers* recipe, you'll see how to deploy a shared printer.

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. This could be useful in situations where users create large numbers of printed documents. In the *Creating a printer pool* recipe, you see how you can automate the creation of a printer pool, using RunDLL32.Exe.

This chapter makes use of a sales team inside Reskit.Org, which has a number of users  
and groups contained in an organizational unit (OU). You create these AD resources as follows:  
# Create-SaleTeam.ps1  
# Creates the OU, groups, users and group memberships  
# used in Reskit.Org  
# Create Sales OU  
$OUPath = 'DC=Reskit,DC=Org'  
New-ADOrganizationalUnit -Name Sales -Path $OUPath  
# Setup for creating users for sales  
$OUPath = 'OU=Sales,DC=Reskit,DC=Org'  
$Password = 'Pa$$w0rd'  
$PHT = @{  
String = $Password  
AsPlainText = $true  
Force = $true  
}  
$PasswordSS = ConvertTo-SecureString @PHT  
$NewUserHT = @{  
AccountPassword = $PasswordSS;  
Enabled = $true;  
PasswordNeverExpires = $true;  
ChangePasswordAtLogon = $false  
Path = $OUPath  
}  
# Create Sales users Nigel, Samantha, Pippa, Jeremy  
$NewUserHT.SamAccountName = 'Nigel'  
$NewUserHT.UserPrincipalName = 'Nigel@Reskit.Org'  
$NewuserHT.Name = 'Nigel'  
$NewUserHT.DisplayName = 'Nigel Hwathorn-Smyth'  
New-ADUser @NewUserHT  
$NewUserHT.SamAccountName = 'Samantha'  
$NewUserHT.UserPrincipalName = 'Samantha@Reskit.Org'  
$NewuserHT.Name = 'Samantha'  
$NewUserHT.DisplayName = 'Samantha Rhees-Jenkins'  
Managing Printing  
218  
New-ADUser @NewUserHT  
$NewUserHT.SamAccountName = 'Pippa'  
$NewUserHT.UserPrincipalName = 'Pippa@Reskit.Org'  
$NewuserHT.Name = 'Pippa'  
$NewUserHT.DisplayName = 'Pippa van Spergel'  
New-ADUser @NewUserHT  
$NewUserHT.SamAccountName = 'Jeremy'  
$NewUserHT.UserPrincipalName = 'Jeremy@Reskit.Org'  
$NewuserHT.Name = 'Jeremy'  
$NewUserHT.DisplayName = 'Jeremy Cadwalender'  
New-ADUser @NewUserHT  
# Create Sales Groups  
$GSHT = @{GroupScope = 'Global'}  
$OUPath = 'OU=Sales,DC=Reskit,DC=Org'  
New-ADGroup -Name Sales -Path $OUPath @GSHT  
New-ADGroup -Name SalesAdmins -Path $OUPath @GSHT  
New-ADGroup -Name SalesPrinterUsers -Path $OUPath @GSHT  
# Add users to the groups  
Add-ADGroupMember -Identity Sales -Members Nigel, Samantha  
Add-ADGroupMember -Identity Sales -Members Pippa, Jeremy  
Add-ADGroupMember -Identity SalesAdmins -Members Nigel, Samantha  
Add-AdgroupMember -Identity SalesPrinterUsers -Members Sales  
Add-AdgroupMember -Identity SalesPrinterUsers -Members ThomasL  
You can find this script in the GitHub repository for this book. The script name is CreateSalesTeam.ps1. You can find this script at https://github.com/doctordns/  
PowerShellCookBook2019/blob/master/Chapter%2007%20-%20Managing%20  
Printers/Create-SalesTeam.ps1

# 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. With those installed, you can create and share a printer for others to access.

In this recipe, you download and install two Xerox printer drivers. One of the drivers is used in this recipe; the other is used in the *Changing printer drivers* recipe. This download comes as a .ZIP archive that you need to extract before you can use the drivers.

Note: if you're using this recipe to suport 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 so you install them via a downloadable executable. You would need run the downloaded executable and then run it on your print server to add the drivers. Thus, this recipe may not be applicable to all printing devices.

## Getting Ready

This recipe uses a Windows Server host PSRV. This is a domain joined Windows Server 2022 host on which you have installed PowerShell 7 and VS Code.

## How to do it...

1. Installing the Print-Server feature on PSRV

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

1. Creating a folder for the Xerox printer drivers:

$NIHT = @{

  Path        = 'C:\Foo\Xerox'

  ItemType    = 'Directory'

  Force       = $true

  ErrorAction = "Silentlycontinue"

}

New-Item @NIHT | Out-Null

1. Downloading printer drivers for 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

1. Expanding the ZIP file

$Drivers = 'C:\Foo\Xerox\Drivers'

Expand-Archive -Path $Target -DestinationPath $Drivers

1. Installing the drivers

$M1 = '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 "$M1"  /f "$P"

$M2 = 'Xerox WorkCentre 6515 PCL6'

rundll32.exe printui.dll,PrintUIEntry /ia /m "$M2"  /f "$P"

1. Adding a PrinterPort for a new printer

$PPHT = @{

  Name               = 'SalesPP'

  PrinterHostAddress = '10.10.10.61'

}

Add-PrinterPort @PPHT

1. Adding the printer to PSRV

$PRHT = @{

  Name = 'SalesPrinter1'

  DriverName = $m1

  PortName   = 'SalesPP'

}

Add-Printer @PRHT

1. Sharing the printer

Set-Printer -Name SalesPrinter1 -Shared $True

1. 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

## How it works...

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

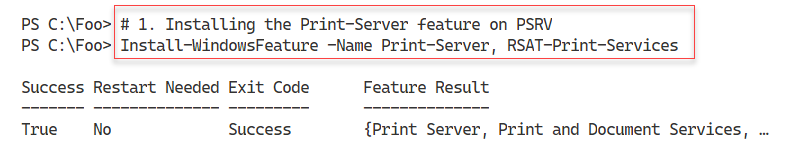


Figure 11.1: Viewing commands in the NTFSSecurity module

**Insert image B42024\_11\_01.png**

In step 2, you create a new folder that you use 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 creaed and making use of the Xerox printer drivers you have downloaded. Finally, in step 8, you share the printer so othyer users can print to this printer. These 7 steps produce no output.

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

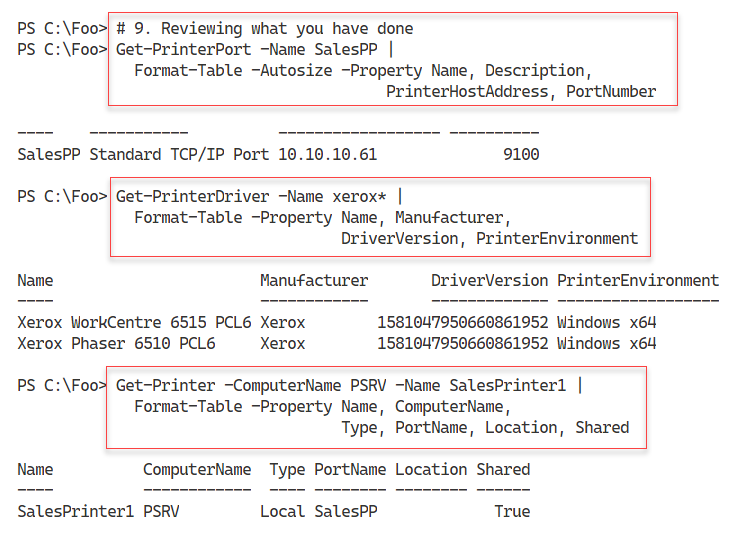


Figure 11.2: Viewing printer components on PSRV

**Insert image B42024\_11\_02.png**

## There's more...

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

In step 5, you use PrintUI.DLL and RunDLL32.EXE. 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. That can often result in displaying another Windows form (as opposed to generating text output).

RunDLL32.EXE allows you to run the same functions within PrintUI.DLL. In many cases, using PrintUI.DLL generates a pop-up form for you to use, which is useful, but not so helpful in terms of automation.

In practice, sometimes you may see curious errors when using PrintUI.DLL. The solution is to reboot the server.

A further downside to using PrintUI.DLL is that the error messages aren't very actionable, should you make errors with the complex syntax involved. You can get help information by opening a PowerShell console window and running the following command:

rundll32 printui.dll PrintUIEntry

In this recipe, you downloaded and installed two drivers, although you only used one to create the SalesPrinter1 printer. You use the second driver in the Changing printer drivers recipe later in this chapter.

# Publishing a printer

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

## Getting Ready

Before running this recipe, you need to have the PSRV printer server set up (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

1. Viewing the printer details

$Printer | Format-Table -Property Name, Published

1. Publishing and sharing the printer to AD

$Printer | Set-Printer -Location '10th floor 10E4'

$Printer | Set-Printer -Shared $true -Published $true

1. Viewing the updated publication status

Get-Printer -Name SalesPrinter1 |

  Format-Table -Property Name, Location, DriverName, Published

## How it works...

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

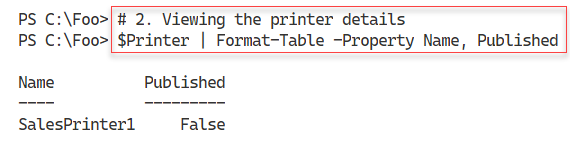


Figure 11.3: Viewing the SalesPrinter1 printer on PSRV

**Insert image B42024\_11\_03.png**

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:

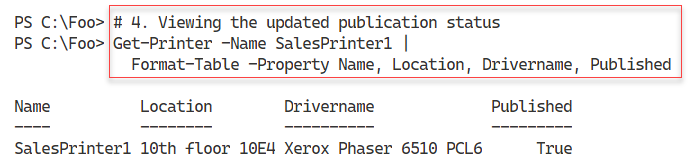


Figure 11.4: Viewing the SalesPrinter1 printer details

**Insert image B42024\_11\_04.png**

## 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 onto any computer in the domain, you can get to this dialog 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:

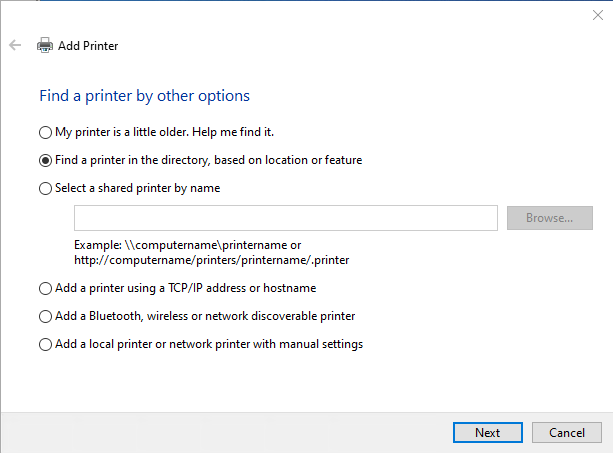


Figure 11.5: Using the Add Printer dialog

**Insert image B42024\_11\_05.png**

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

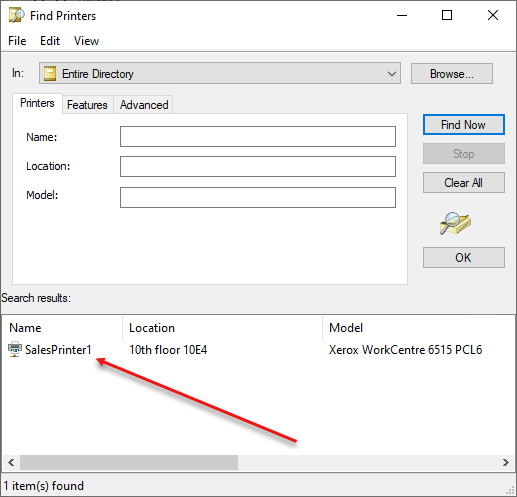


Figure 11.6: Using the Find Printers dialog

**Insert image B42024\_11\_06.png**

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

# 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. For a busy print server with multiple printers, you may wish to change the default spool folder.

## Getting Ready

This recipe uses the PSRV printer server that was set up as per the Installing and sharing printers recipe.

## How to do it...

1. Loading the System.Printing namespace and classes

Add-Type -AssemblyName System.Printing

1. Defining the required permissions

$Permissions =

   [System.Printing.PrintSystemDesiredAccess]::AdministrateServer

1. Creating a PrintServer object with the required permissions

$NOHT = @{

  TypeName     = 'System.Printing.PrintServer'

  ArgumentList = $Permissions

}

$PS = New-Object @NOHT

1. Observing the default spool folder

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

1. Creating a new spool path

$NIHT = @{

  Path        = 'C:\SpoolPath'

  ItemType    = 'Directory'

  Force       = $true

  ErrorAction = 'SilentlyContinue'

}

New-Item @NIHT | Out-Null

1. Updating the default spool folder path

$Newpath = 'C:\SpoolPath'

$PS.DefaultSpoolDirectory = $Newpath

1. Committing the change

$Ps.Commit()

1. Restarting the Spooler to accept the new folder

Restart-Service -Name Spooler

1. Checking the result

New-Object -TypeName System.Printing.PrintServer |

  Format-Table -Property Name,

                DefaultSpoolDirectory

1. Stopping the Spooler service

Stop-Service -Name Spooler

1. Creating a new spool directory

$SPL = 'C:\SpoolViaRegistry'

$NIHT2 = @{

  Path        = $SPL

  Itemtype    = 'Directory'

  ErrorAction = 'SilentlyContinue'

}

New-Item  @NIHT2 | Out-Null

1. Creating the spooler folder and configuring in the registry

$RPath = 'HKLM:\SYSTEM\CurrentControlSet\Control\' +

         'Print\Printers'

$IP = @{

  Path    = $RPath

  Name    = 'DefaultSpoolDirectory'

  Value   = $SPL

}

Set-ItemProperty @IP

1. Restarting the spooler

Start-Service -Name Spooler

1. 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 Print Server 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:

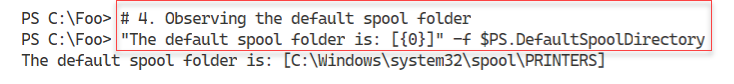


Figure 11.7: Examining default spool folder

**Insert image B42024\_11\_07.png**

In step 5, you create a new folder on PSRV to serve as your print servers spool folder. In step 6, You update the printer spool folder path. In step 7, you commit this change and in step 8, you restart the spooler service. These four steps produce no console output.

In step 9, you re-view the printer details with output like this:

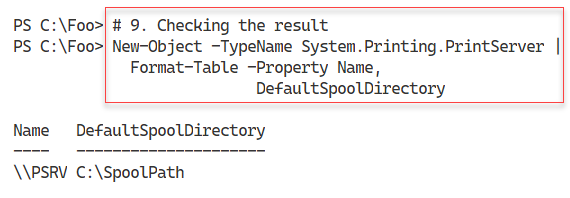


Figure 11.8: Examining default spool folder

**Insert image B42024\_11\_08.png**

In step 1 through step 9, you set and validate the spool folder using a .NET Object. There is another separate way to change the spool folder which involves using WMI.

In step 10, you stop the spooler service. In step 11, you create a new folder and in step 12, you configure the necessary registry settings to contain the path to the new spool folder. Then in step 13, you restart the spooler. These four steps produce no console output. With the changes made to the spool folder within the registry, in step 14, you can view the updated spool folder with outout like this:

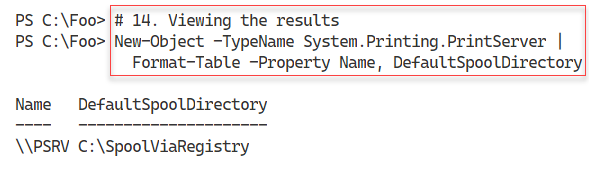


Figure 11.9 Examining updated spool folder

**Insert image B42024\_11\_09.png**

## There's more...

For most organizations with larger number of shared printers, configuring ther print server to use another folder for spooling is a good idea. If possible, use a separate disk drive if you can to avoid the risk of the spool folder filling up.

In this recipe, you used two different mechanisms to change the spooler folder. One uses a .NET object (which isn't loaded by default), while the other involves directly editing the registry. Needless to say, if you're rolling out printers using scripts, particularly ones that edit the registry, careful testing is vital.

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

# Changing printer drivers

On occasion, it may be necessary to change the printer driver for a printer. For example, you might be replacing an existing print device with a new or different model. In this case, you want the printer name to remain the same, but you need to update the printer’s print driver. In the Installing and sharing a printer 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, the Xerox Phaser 6515 PCL6.

The assumption behind this recipe is that the printer name and printer port (including the printer's IP address and port number) don't change, only the driver.

## Getting Ready

Run this recipe on the PSRV host, set up as per the Installing and sharing printers recipe.

## How to do it...

1. Adding the print driver for the new printing device

$M2 = 'Xerox WorkCentre 6515 PCL6'

Add-PrinterDriver -Name $M2

1. Getting the Sales group printer object and store it in $Printer

$Printern = 'SalesPrinter1'

$Printer  = Get-Printer -Name $Printern

1. Updating the driver using the Set-Printer cmdlet

$Printer | Set-Printer -DriverName $M2

1. Observing the result

Get-Printer -Name $Printern |

  Format-Table -Property Name, DriverName, PortName,

                Published, Shared

## How it works...

In step 1, you use the Add-PrinterDriver cmdlet to add the pirhnter driver for the printer. In step 2, you obtain the printer details and in step 3, you update the printer to use the updated driver. These steps produce no console output.

In step 4, you use Get-Printer to observe that you have installed the updated driver for the SalesPrinter1 printer, which looks like this:

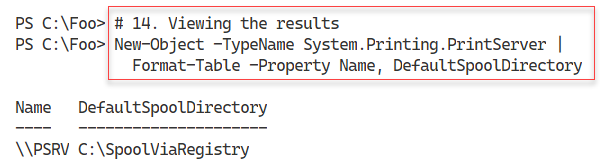


Figure 11.10: Viewing updated printer drivere for SalesPrinter1

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

# 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

1. Displaying the number of printers defined on PSRV

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

1. Getting the Sales Group printer WMI object

$Printer = $Printers |

  Where-Object Name -eq 'SalesPrinter1'

1. Displying the printer's details

$Printer | Format-Table -AutoSize

1. Printing a test page

Invoke-CimMethod -InputObject $Printer -MethodName PrintTestPage

1. Checking on 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 are defined on your printer server with output like this:

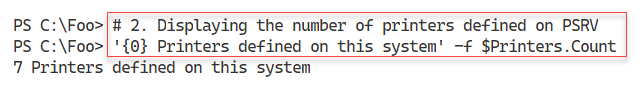


Figure 11.11: Viewing printers on PSRV

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

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

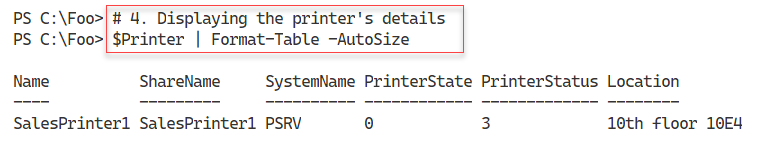


Figure 11.12: Viewing printers on PSRV

**Insert image B42024\_11\_12.png**

In step 5, you use Invokje-CimMethod to run the PrintTestPage method on the printer, This step generates a simple printer test page, with console output like this:

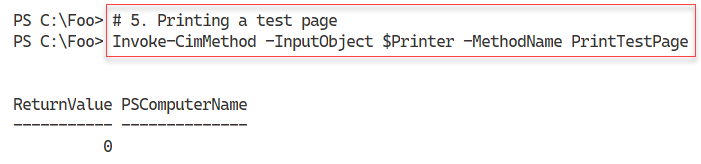


Figure 11.3 Printing a test page

**Insert image B42024\_11\_13.png**

In the final step in this recipe, yhou 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:

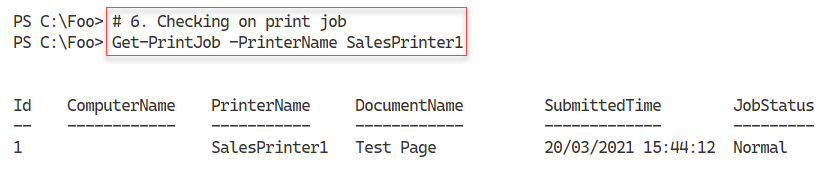


Figure 11.14: Viewing printer jobs to observe test page

**Insert image B42024\_11\_14.png**

## 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 pretty much straight away. If you do not get any physical output, you need to carry out some routing printer trouble shooting: Is the printer turned on, is the network cable pluged on and working, etc.

# Modifying printer security

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

By default, when you create a printer, Windows adds some ACEs to the printer's ACL. This includes giving the Everyone group the permission 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 a number of 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 you can use third-party scripts that does the job for you. But the code for can be complex (and easy to mess up). Make sure you test any recpies that modify printer ACLs very carefully. An always have a way to resetting 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 manager the ACL using the GUI if you really need to!

## Getting Ready

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

## How to do it...

1. Setting up AD for this recipe

$SB = {

 # 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 $SB

1. Creating an NTAccount object

$GHT1 = @{

    Typename     = 'Security.Principal.NTAccount'

    Argumentlist = 'SalesGroup'

}

$SalesGroup = New-Object @GHT1

$GHT2 = @{

    Typename     = 'Security.Principal.NTAccount'

    Argumentlist = 'SalesAdmins'

}

$SalesAdminGroup = New-Object @GHT2

1. Getting the group SIDs

$SalesGroupSid =

  $SalesGroup.Translate([Security.Principal.Securityidentifier]).Value

$SalesAdminGroupSid =

  $SalesAdminGroup.Translate([Security.Principal.Securityidentifier]).Value

1. Defining the SDDL for this printer

$SDDL = 'O:BAG:DUD:PAI(A;OICI;FA;;;DA)' +

        "(A;OICI;0x3D8F8;;;$SalesGroupSid)" +

        "(A;;LCSWSDRCWDWO;;;$SalesAdminGroupSid)"

1. Getting the Sales group's printer object

$SGPrinter = Get-Printer -Name SalesPrinter1 -Full

1. Setting the permissions

$SGPrinter | Set-Printer -Permission $SDDL

1. Viewing the permissions from the GUI

View Settings in the windows Printer GUI

## 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.Principlal.NTAccounts for the two security groups. In step 3, you use these two new object to retrieve the Security IDs (SIDs) for each of the group. In step 4, you cratate a Security Descriptor Description Language (SDDL) permission set.

In step 5, you use the Get-Printer to return a WMI object that describesd 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:

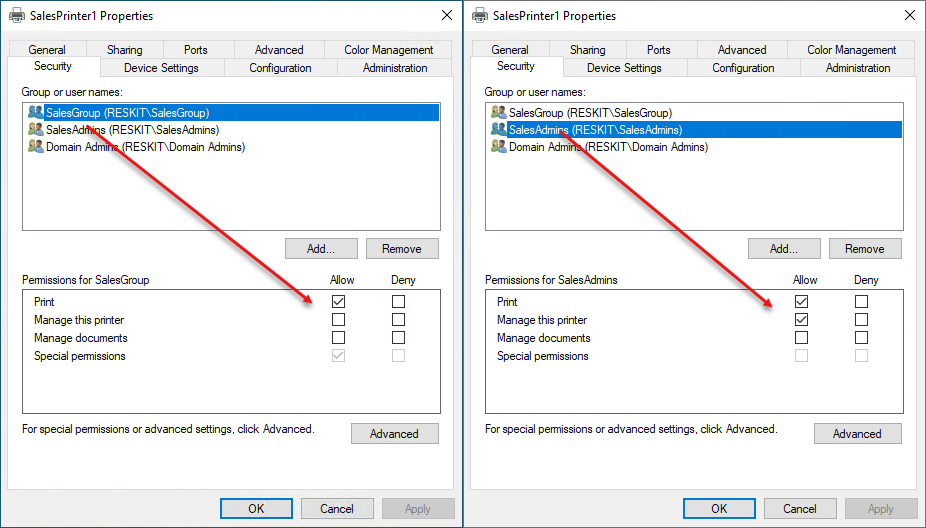


Figure 11.15: Viewing printer security

**Insert image B42024\_11\_15.png**

## There's more...

In this recipe, you create an ACL for the Sales Group printer SalesPrinter1. The recipe makes use of a .NET object to obtain the SIDs for two security group. 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 setting of ACLs. SDDL is the default mechanism, but is not always all that 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 a lot documewnts and need the speed that additional printers can provide without asking the user to choose the specific print device to use.

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 an excellent example of utilizing older console applications to achieve your objective. The Windows Print UI uses a DLL (PrintUI.DLL) to perform the actions generated via the GUI. With the use of RunDLL32, you can invoke those same functions via PowerShell.

## 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

1. 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)

1. Viewing resultant details

Get-Printer $Printer |

   Format-Table -Property Name, Type, DriverName, PortName

## 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 outout to confirm you have setup a printer pool of two print devices/ports. The output of this final step looks like this:

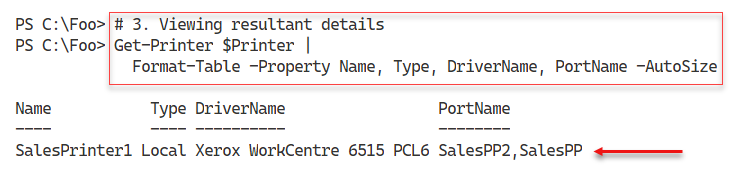


Figure 11.16: Viewing printer pool

**Insert image B42024\_11\_16.png**

## There's more...

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

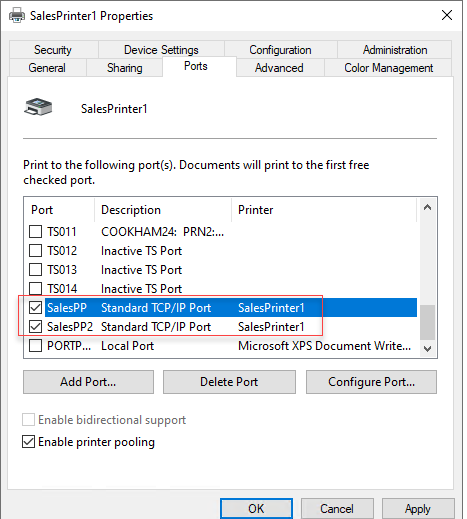


Figure 11.17: Viewing printer pool from the GUI

**Insert image B42024\_11\_17.png**