7

Managing Printing

In this chapter, we cover the following recipes:

* Installing and sharing printers
* Publishing a printer
* Changing the spool 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 2019, 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 2019 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'

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 Create-SalesTeam.ps1. You can find this script at <https://github.com/doctordns/PowerShellCookBook2019/blob/master/Chapter%2007%20-%20Managing%20Printers/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. You can also, as you see in the Deploying shared printers recipe, push out the printer details to client computers.

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 adapting this recipe to use other printer makes and models, this recipe may have to be adapted. In some cases (such as with Hewlett Packard printers), the printer drivers are installed via a downloadable executable that wasn't designed to be automated. You need run the downloaded executable and then run it on your print server to add the drivers.

## Getting ready

In this recipe, you install the Print-Server feature to set up a print server on PSRV, and then set up a TCP/IP printer. PSRV is a domain-joined Windows 2019 server with only the default features installed. This server is used throughout this chapter.

## How to do it...

1. Install the Print-Server feature on PSRV, along with the Print Management RSAT tools:

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

1. Create a folder on PSRV for the Xerox printer drivers:

$NIHT = @{

Path = 'C:\Foo\Xerox'

ItemType = 'Directory'

Force = $true

ErrorAction = "SilentlyContinue"

}

New-Item @NIHT | Out-Null

1. Download the 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. Expand the archive into the C:\Foo\Xerox\Drivers folder:

$Drivers = 'C:\Foo\Xerox\Drivers'

Expand-Archive -Path $Target -DestinationPath $Drivers

1. Install the two Xerox printer 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. Add an IP address-based PrinterPort for a new printer:

$PPHT = @{

Name = 'SalesPP'

PrinterHostAddress = '10.10.10.61'

}

Add-PrinterPort @PPHT

1. Add the printer to PSRV:

$PRHT = @{

Name = 'SalesPrinter1'

DriverName = $M1 # Xerox Phaser 6510 PCL6

PortName = 'SalesPP'

}

Add-Printer @PRHT

1. Share the printer:

Set-Printer -Name SalesPrinter1 -Shared $True

1. Review the printer port, printer driver, and printer you just created:

Get-PrinterPort -Name SalesPP |

Format-Table -Autosize -Property Name, Description,

PrinterHostAddress, PortNumber

Get-PrinterDriver -Name xerox\* |

Format-Table -Property Name, Manufacturer,

DriverVersion, PrinterEnvironment

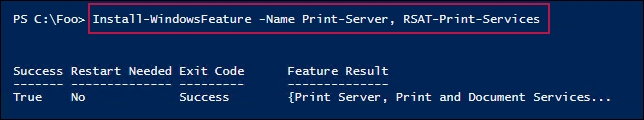
Get-Printer -ComputerName PSRV -Name SalesPrinter1 |

Format-Table -Property Name, ComputerName,

Type, PortName, Location, Shared

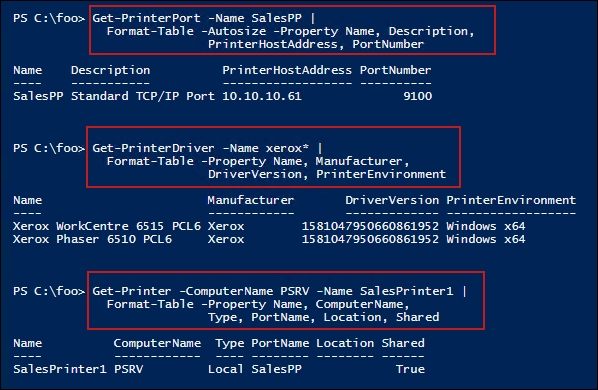
## How it works…

In step 1, you install the Print-Server feature on PSRV, along with the management tools, which looks like this:



In step 2, you ensure that there's a folder to hold the drivers. With step 3, you use the BITS service to download the drivers. In step 4, you extract the downloaded driver archive. In step 5, you install one print driver. In step 6, you add a printer port for a networked printer. In step 7, you create the networked printer based on the printer port and printer driver you just added. Finally, in step 8, you share the printer. These steps produce no output.

In step 9, you examine the printer port, printer driver, and the printer created in this recipe, which looks like this:



## There's more...

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. Get the printer to publish:

$Printer = Get-Printer -Name SalesPrinter1

1. Display the publication status:

$Printer | Format-Table -Property Name, Published

1. Set the printer details and publish the printer to AD:

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

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

1. View the updated publication status:

Get-Printer -Name SalesPrinter1 |

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

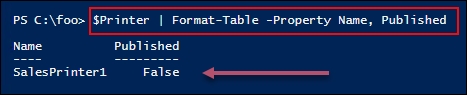
1. See the shared printer status:

Get-SmbShare -Name SalesPrinter1

## How it works…

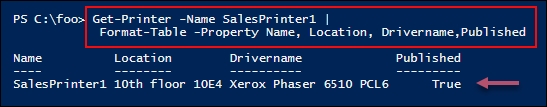
In step 1, you retrieved the printer details for the SalesPrinter1 printer that you set up in the Installing and sharing printers recipe. There's no output from this step.

In step 2, you display the printer details and can see that the printer is unpublished, which looks like this:



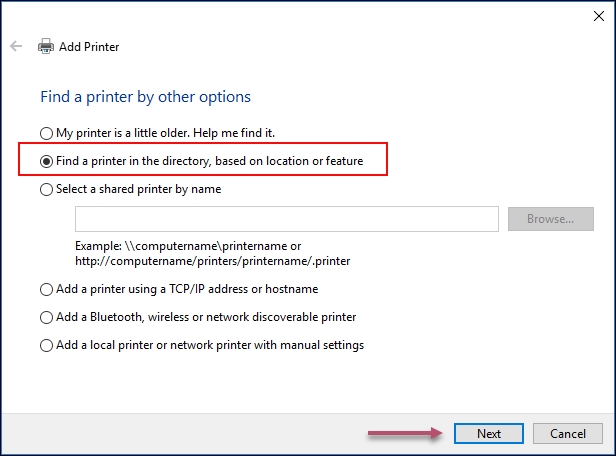
In step 3, you use the Set-Printer cmdlet to set the location details for the printer and publish the printer in AD. This step updates the printer details in the AD and produces no output.

In step 4, you view the details of the printer, which looks like this:

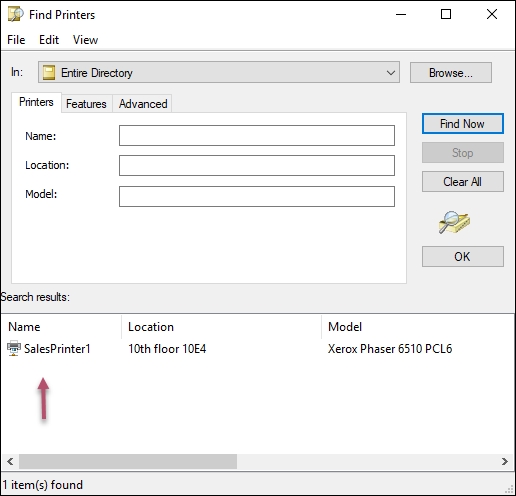


## There's more...

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 into the client computer, CL1, 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:



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



# Changing the spool directory

During the printing process, the Windows printer spooler in Windows uses an on-disk folder to hold the temporary files that have been created. If multiple users each print large documents to a single printer, the print queue 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. Load the System.Printing namespace and classes:

Add-Type -AssemblyName System.Printing

1. Define the required permissions—that is, the ability to administrate the server:

$Permissions =

[System.Printing.PrintSystemDesiredAccess]::

AdministrateServer

1. Create a PrintServer object (in memory) with the required permissions:

$NOHT = @{

TypeName = 'System.Printing.PrintServer'

ArgumentList = $Permissions

}

$PS = New-Object @NOHT

1. Create a new spool path:

$NIHT = @{

Path = 'C:\SpoolPath'

ItemType = 'Directory'

Force = $true

ErrorAction = 'SilentlyContinue'

}

New-Item @NIHT | Out-Null

1. Update the default spool folder path on the in-memory object:

$Newpath = 'C:\SpoolPath'

$PS.DefaultSpoolDirectory = $Newpath

1. Commit the change:

$Ps.Commit()

1. Restart the spooler, which now uses the new Spooler folder:

Restart-Service -Name Spooler

1. Once the spooler has restarted, view the results:

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:

1. Stop the Spooler service:

Stop-Service -Name Spooler

1. Create a new spool folder for the print server to use:

$SPL = 'C:\SpoolViaRegistry'

$NIHT2 = @{

Path = $SPL

Itemtype = 'Directory'

ErrorAction = 'SilentlyContinue'

}

New-Item @NIHT2 | Out-Null

1. Set the details in the registry:

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

'Print\Printers'

$Spooldir = 'C:\SpoolViaRegistry' # Folder should exist

$IP = @{

Path = $RPath

Name = 'DefaultSpoolDirectory'

Value = $SPL

}

Set-ItemProperty @IP

1. Restart the Spooler:

Start-Service -Name Spooler

1. View the results:

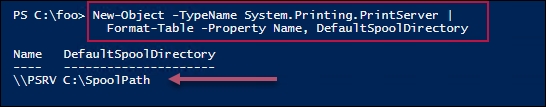
New-Object -TypeName System.Printing.PrintServer |

Format-Table -Property Name, DefaultSpoolDirectory

## How it works…

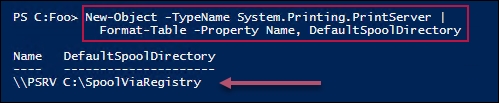
In step 1, you loaded the System.Printing assembly to provide additional .NET classes that relate to printing and print servers. In step 2, you created an object that defines the permissions needed to administrate the printer server. In step 3, you created an in-memory print server object using the permissions object you just instantiated. In step 4, you created a new folder to serve as the spooler folder. In step 5, you updated the default spooler path to the in-memory spooler object. In step 6, you committed the changes you just made to the spooler object, and then in step 7, you restarted the Spooler service. These first seven steps produced no output.

In step 8, after the Spooler service restarted, you viewed the properties of the print server, including the default printer spool directory you just updated. It looks like this:



Then, you updated the spooler folder by directly editing the registry. In step 9, you stopped the spooler service. In step 10, you created a second spooler folder, and in step 11, you set the relevant registry setting for the printer spooler folder. Then, in step 12, you restarted the spooler service. Step 9 through step 12 produced no output.

In step 13, you looked at the updated printer spooler properties, which looks like this:



## There's more...

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 actual 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 PSRV1 printer, set up as per the Installing and sharing printers recipe.

## How to do it...

1. Add the print driver for the new printing device:

$M2 = 'Xerox WorkCentre 6515 PCL6'

Add-PrinterDriver -Name $M2

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

$Printern = 'SalesPrinter1'

$Printer = Get-Printer -Name $Printern

1. Update the driver using the Set-Printer cmdlet:

$Printer | Set-Printer -DriverName $M2

1. Observe the result:

Get-Printer -Name $Printern |

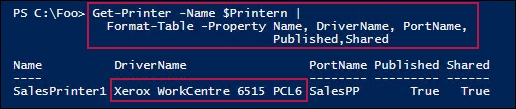
Format-Table -Property Name, DriverName, PortName,

Published, Shared

## How it works…

In step 1, you added a printer driver for the Xerox 6515 device. In step 2, you instantiated an in-memory printer object that represents the print server on PSRV. In step 3, you updated SalesPrinter1 with the details of the updated driver name. These steps produced no output.

In step 4, you reviewed the changes made, which look like this:



# 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. Get the printer objects from WMI:

$Printers = Get-CimInstance -ClassName Win32\_Printer

1. Display the number of printers defined on PSRV:

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

1. Get the sales group printer WMI object:

$Printer = $Printers |

Where-Object Name -eq "SalesPrinter1"

1. Display the printer's details:

$Printer | Format-Table -AutoSize

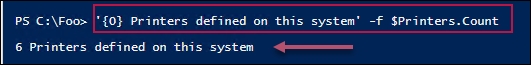
1. Print a test page:

Invoke-CimMethod -InputObject $Printer -MethodName PrintTestPage

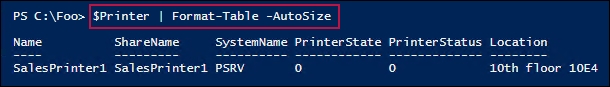
## How it works…

In step 1, you used Get-CimInstance to return all the printers defined on this system. There's no output from this step.

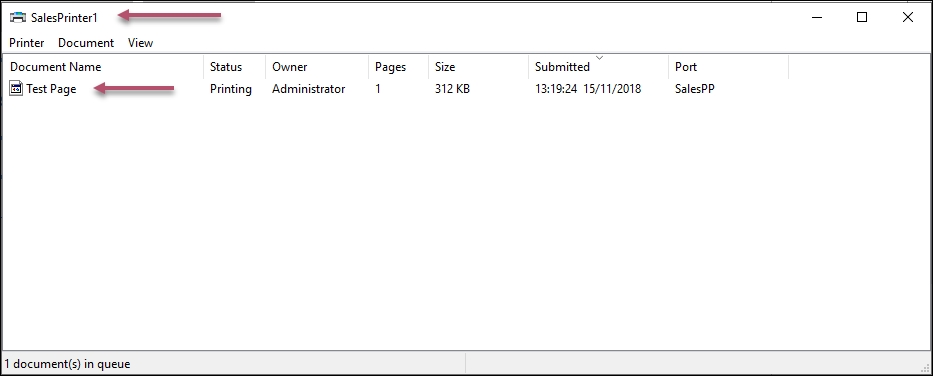
In step 2, you displayed the total printers defined, which looks like this:



In step 3, you got the printer object that corresponds to the sales group LaserJet printer, which generates no output. In step 4, you displayed the details of this printer, which looks like this:



In step 5, you invoked the PrintTestPage method on the sales group printer (SalesPrinter1) to generate a test page on the printer. Using the printer MMC snap-in, generating a test page looks like this:



# Managing 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; there are no cmdlets for managing ACLs on printers. You can always use .NET directly to manage the ACL, or you can use a third-party script that does the job for you. But the code for that is complex (and easy to mess up). For simplicity in this case, you're going to download and use a script, Set-PrintPermissions.ps1, available on TechNet.

The SetPrintPermissions.ps1 script enables you to grant or deny printer permissions to AD users or groups. In this recipe, you use this script to remove the permission for members of the everyone group, then enable print, document, and print management to members of a different group. The script uses permission names that are familiar to IT pros, making this script easy to adapt.

A downside to this script is that there's no feature in the script to display the permissions on a printer object. You can always use .NET security classes to delve into the permissions, or (as you do in this recipe) just use the GUI to verify the actual permissions.

Note

During the development of Windows Server 2019, printers that are created with the recipes in this chapter have some interesting ACEs set. One ACE is set with an SID that relates to the SID that belongs to 'defaultuser0'. To avoid confusion, the graphics in the How it works… section don't show these ACEs.

## Getting ready

This recipe uses AD accounts to set permissions based on the Create-SalesGroup.ps1 script . You need the PSRV print server and the DC1 domain controller. To test the permissions, you can use CL1.

## How to do it...

1. Download the Set-PrinterPermissions.ps1 script:

$URL = 'https://gallery.technet.microsoft.com/scriptcenter/' +

'Modify-Printer-Permissions-149ae172/file/116651/1/' +

'Set-PrinterPermissions.ps1'

$Target = 'C:\Foo\Set-PrinterPermissions.ps1'

Start-BitsTransfer -Source $URL -Destination $Target

1. Get help on the script:

Get-Help $Target

1. Use PrintUI.DLL to bring up the printer properties GUI:

rundll32.exe printui.dll,PrintUIEntry /p /nSalesprinter1

1. From the GUI, click on Security to view the initial ACL.
2. Remove the Everyone ACE from the ACL for SalesPrinter1:

$SPHT1 = @{

ServerName = 'PSRV'

Remove = $True

AccountName = 'EVERYONE'

SinglePrinterName = 'SalesPrinter1'

}

C:\foo\Set-PrinterPermissions.ps1 @SPHT1

1. Enable the members of the Sales group to print to this printer:

$SPHT2 = @{

ServerName = 'PSRV'

AccountName = 'Reskit\Sales'

AccessMask = 'Print'

SinglePrinterName = 'SalesPrinter1'

}

C:\foo\Set-PrinterPermissions.ps1 @SPHT2

1. Give SalesAdmins permission to manage the documents:

$SPHT3 = @{

ServerName = 'PSRV'

AccountName = 'Reskit\SalesAdmins'

AccessMask = 'ManageDocuments'

SinglePrinterName = 'SalesPrinter1'

}

C:\foo\Set-PrinterPermissions.ps1 @SPHT3

1. Bring up the Printer GUI:

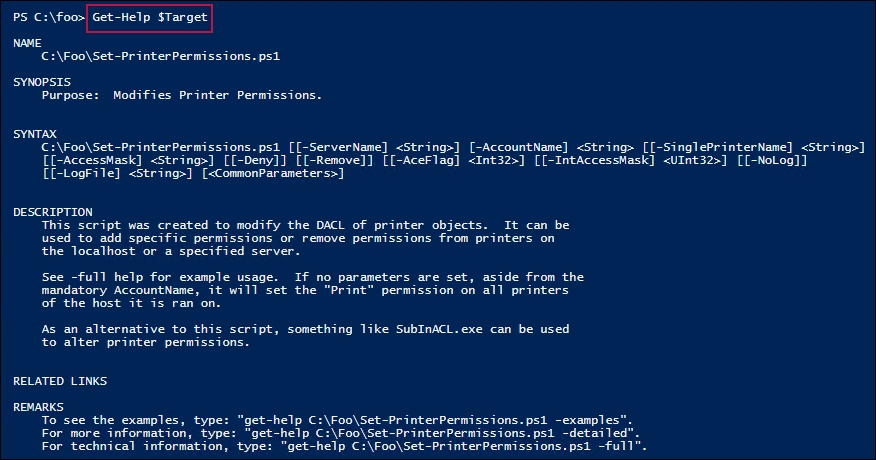
rundll32.exe printui.dll,PrintUIEntry /p /nSalesprinter1

1. Click the Security tab and view the updated ACL.

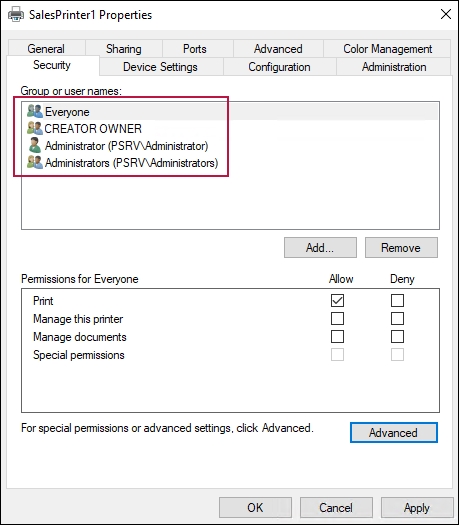
## How it works…

In step 1, you used the BITS service to download the Set-PrinterPermissions script. This step generates no output.

The Set-PrinterPermissions script has comment-based help, which provides usage assistance. In step 2, you used Get-Help to view the basic help information, which looks like this:

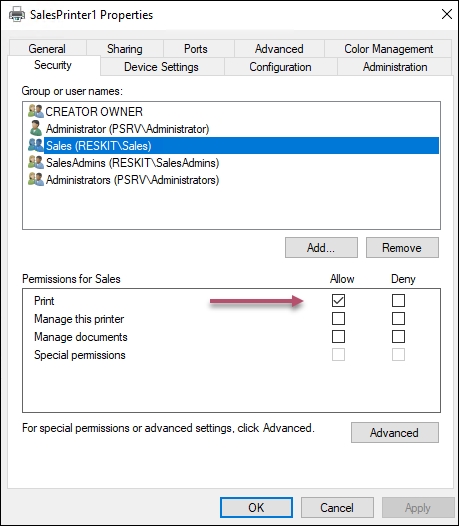


In step 3, you used PrintUI.DLL to bring up the properties of the Salesprinter1 printer. In step 4, you viewed the initial ACL for the printer, which looks like this:

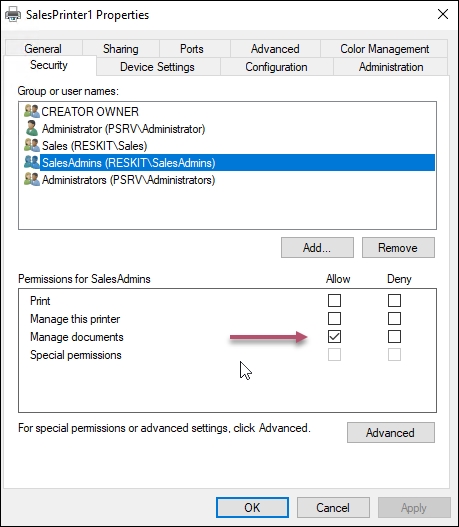


In step 5, you used the Set-Printerpermissions script to remove the default ACE, allowing everyone to print to the printer. In step 6, you enabled the members of the sales group to print to the printer, and in step 7, you enabled the members of the sales admins group to manage the printer. These steps produced no output to the console.

In step 8, you printed the printer UI to view the updated ACL, which looks like this:



If you click on Sales, you can see their permissions, like this:



## There's more...

In step 1, you used the BITS service to download the Set-PrinterPermissions script using the URL of the script. Alternatively, you could have used your search engine and downloaded it directly. The script isn't available on PowerShell Gallery.

In step 3 and step 8, you used PrintUI.DLL to display a print-management GUI from which you can view the updated ACL for this printer. As you can see, the members of the Sales group are able to print, and members of the SalesAdmins group can manage the printer.

# Creating a printer pool

Windows allows you to create a printer pool, which is a printer with two or more print devices (that is, printer ports) available. Windows sends a given print job to any of the printers in the pool. This feature is useful in environments where users do a lot of printing and need the speed that additional printers can provide, without having to ask the user to choose a 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 make use of PrintUI.DLL and RunDLL32 to deploy a printer pool. This is another example of utilizing older console applications to achieve your objective.

## Getting ready

Run this recipe on the PSRV print server. This recipe uses the printer and port you created in the Installing and sharing printers recipe.

## How to do it...

1. Add an additional port for the SalesPrinter1 printer:

$P = 'SalesPP2' # new port name

Add-PrinterPort -Name $P -PrinterHostAddress 10.10.10.62

1. Create the printer pool for SalesPrinter1:

$P1='SalesPP'

$P2='SalesPP2'

rundll32.exe printui.dll,PrintUIEntry /Xs /n $p Portname $P1,$P2

1. View the resulting details, which show both printer ports:

$P = 'SalesPrinter1'

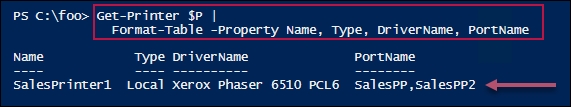
Get-Printer $P |

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

## How it works…

In step 1, you added a new printer port (SalesPP2) to serve as a second printer port for the SalesPrinter1 printer. In step 2, you set the two printer ports for SalesPrinter1, thus creating the printer pool. These steps produced no output.

In step 3, you can see the result of the first two steps. Notice that the SalesPrinter1 printer now has two available printer ports, as can be seen here:



## There's more...

In creating a printer pool, it's important to ensure that all print devices in a pool are the same, or at least can use the same driver. For example, you can use HP drivers for many printers. Having identical models and matching drivers is preferable.