7

Managing Networking in  
the Enterprise

In this chapter, we cover the following recipes:

* Configuring IP addressing
* Testing network connectivity
* Installing DHCP
* Configuring DHCP scopes and options
* Using DHCP
* Implementing DHCP failover and load balancing
* Deploying DNS in the Enterprise
* Configuring DNS forwarding
* Managing DNS zones and DNS resource records

# Introduction

Every organization's heart is its network – the infrastructure that enables your client and server systems to interoperate. Windows has included networking features since the early days of Windows for Workgroups 3.1 (and earlier with Microsoft LAN Manager).

One thing worth noting is that even in the cloud age, "the network" isn't going anywhere; its role of enabling a client to connect to a server is just changing, with some servers (and clients) now in the cloud. The cloud is really just resources in someone else's network, and you still need the network to communicate.

Every server or workstation in your environment needs to have a correct IP configuration. While IPv6 is gaining in popularity, most organizations rely on IPV4. In the Configuring IP addressing recipe, we look at setting a network interface's IPv4 configuration, including DNS settings.

Many organizations assign a static IPv4 address to most server systems. The servers used throughout this book, for example, make use of static IP addresses. For client hosts and some servers, an alternative to assigning a static IP address is to use the **Dynamic Host Control Protocol** (**DHCP**). DHCP is a network management protocol that enables a workstation to lease an IP address (and release it when the lease expires). You set up a DHCP server to issue IP address configuration to clients using the Installing DHCP recipe.

Once you have installed your DHCP server, you can use the Configuring DHCP scopes and options recipe to set up the details that your DHCP server is to hand out to clients. In the Configuring DHCP failover and load balancing recipe, we deploy a second DHCP server and configure it to act as a failover/load balancing DHCP service.

In this chapter's final recipe, Configuring DNS zones and resource records, you configure the DNS server on DC1 with zones and additional resource records. Before you can administer your Windows Server 2022 infrastructure, you need to create an environment to use PowerShell to carry out the administration.

# Configuring IP addressing

By default, Windows uses DHCP to configure all NICs that the Windows installation process discovers when installing Windows. Once you complete Windows installation, you can use the control panel applet (ncpa.cpl), the network shell console application (netsh.exe), or, of course, PowerShell to set IP configuration manually. In this recipe, you set the IP address details for SRV2 and ensure the host registers DNS names in the Reskit.Org DNS domain (on the DNS service running on DC1).

Setting up any host requires setting an IP address, a subnet mask, and a default gateway, which you do in the first part of this recipe. Then, you configure SRV2 (a workgroup host) to register with the DNS server on DC1.Reskit.Org. This approach raises some challenges. By default, when you create DC1.Reskit.Org as a DC, the domain promotion process sets the domain's DNS zone to require secure updates. That means a workgroup host cannot register. You can overcome this by setting the zone to allow all updates. But this could be dangerous as it allows ANY host to, potentially, register their address. A second challenge is that since SRV2 is not a domain member, remoting to DC1 fails. A solution to that issue is to set the WinRM service to trust all hosts. Configuring WinRM to disregard server authentication has security implications you should consider before using this approach in production.

## Getting ready

This recipe uses SRV2, a recently added workgroup host. As with all hosts used in this book, you install SRV2 using the Reskit installation scripts you can find on GitHub at <https://github.com/doctordns/ReskitBuildScripts>. You should also have loaded PowerShell 7 and VS Code, as you did in Chapter 1, Installing and Configuring PowerShell 7.1. To simplify this setup, you can run the two setup scripts in <https://github.com/doctordns/PACKT-PS7/tree/master/scripts/goodies>. Run the first script in the Windows PowerShell ISE and the second in an elevated VS Code instance.

Note that using Azure for these VMs is not supported.

By default, this host is a DHCP client.

## How to do it...

1. Discovering the network adapter, adapter interface, and adapter interface index

$IPType    = 'IPv4'

$Adapter   = Get-NetAdapter |  Where-Object Status -eq 'Up'

$Interface = $Adapter | Get-NetIPInterface -AddressFamily $IPType

$Index     = $Interface.IfIndex

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table -Property Interface\*, IPAddress, PrefixLength

1. Setting a new IP address for the NIC

$IPHT = @{

  InterfaceIndex = $Index

  PrefixLength   = 24

  IPAddress      = '10.10.10.51'

  DefaultGateway = '10.10.10.254'

  AddressFamily  = $IPType

}

New-NetIPAddress @IPHT

1. Verifying the new IP address

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table IPAddress, InterfaceIndex, PrefixLength

1. Setting DNS server IP address

$CAHT = @{

  InterfaceIndex  = $Index

  ServerAddresses = '10.10.10.10'

}

Set-DnsClientServerAddress @CAHT

1. Verifying the new IP configuration

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table

1. Testing that SRV2 can see the domain controller

Test-NetConnection -ComputerName DC1.Reskit.Org |

  Format-Table

1. Creating a credential for DC1

$U    = 'Reskit\Administrator'

$PPT  = 'Pa$$w0rd'

$PSC  = ConvertTo-SecureString -String $ppt -AsPlainText -Force

$Cred = [pscredential]::new($U,$PSC)

1. Setting WinRM on SRV2 to trust DC1

$TPPATH = 'WSMan:\localhost\Client\TrustedHosts'

Set-Item -Path $TPPATH -Value 'DC1' -Force

Restart-Service -Name WinRM -Force

1. Enabling non-secure updates to Reskit.Org DNS domain

$DNSSSB = {

  $SBHT = @{

    Name          = 'Reskit.Org'

    DynamicUpdate = 'NonsecureAndSecure'

}

  Set-DnsServerPrimaryZone @SBHT

}

Invoke-Command -ComputerName DC1 -ScriptBlock $DNSSSB -Credential $Cred

1. Ensuring SRV2 registers within the Reskit.Org DNS zone:

$DNSCHT = @{

  InterfaceIndex                 = $Index

  ConnectionSpecificSuffix       = 'Reskit.Org'

  RegisterThisConnectionsAddress = $true

  UseSuffixWhenRegistering       = $true

}

Set-DnsClient  @DNSCHT

1. Registering host IP address at DC1:

Register-DnsClient

1. Prestaging SRV2 in AD:

$SB = {New-ADComputer -Name SRV2}

Invoke-Command -ComputerName DC1 -ScriptBlock $SB

1. Testing the DNS server on DC1.Reskit.Org resolves SRV2:

Resolve-DnsName -Name SRV2.Reskit.Org -Type 'A' -Server DC1.Reskit.Org

## How it works...

In step 1, you use Get-NetAdapter and Get-NetIPAddress to determine the IP address of this server. Then, you display the resultant address, which looks like this:

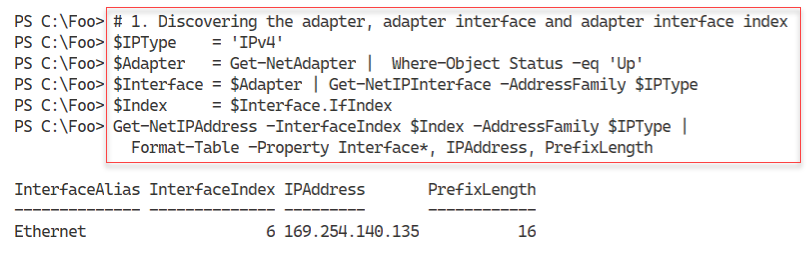


Figure 7.1: Discovering the adapter, adapter interface, and adapter interface index

1. Insert image B16762\_07\_01.png

In step 2, you use the New-NetIPAddress cmdlet to set a static IP address on SRV2. The output looks like this:

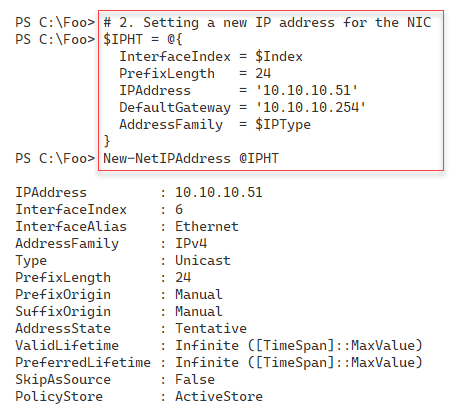


Figure 7.2: Setting a new IP address for the NIC

1. Insert image B16762\_07\_02.png

To double-check that you have configured SRV2 with the correct IP address configuration, you can use the Get-NetIPaddress cmdlet. The output looks like this:

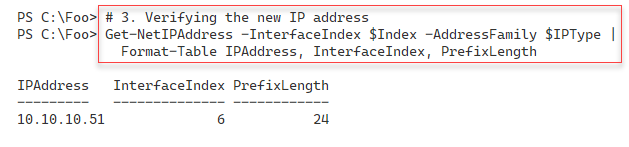


Figure 7.3: Verifying the new IP address

1. Insert image B16762\_07\_03.png

Besides setting an IP address, subnet mask, and default gateway, you also need to configure SRV2 with a DNS server address. In step 4, you use the Set‑DnsClientServerAddress cmdlet, which creates no output.

To check the updated IP configuration on SRV2, in step 5, you verify the configuration by (re)using the Get-Get-NetIPAddress cmdlet, with output like this:

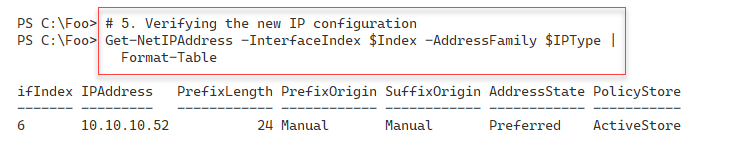


Figure 7.4: Verifying the new IP configuration

1. Insert image B16762\_07\_04.png

In step 6, you use Test-interconnection to ensure SRV2 can connect to DC1, the domain controller in the Reskit.Org domain, with this as the output:

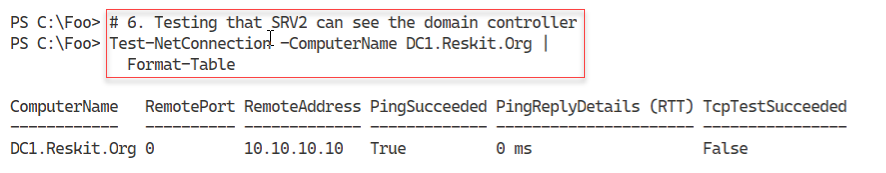


Figure 7.5: Testing that SRV2 can see the domain controller

1. Insert image B16762\_07\_05.png

To enable SRV2, a workgroup computer to run commands on DC1, you need the correct Windows credentials. In step 7, which creates no output, you create credentials for the Administrator user in Reskit.Org.

With step 8, you configure the WinRM service to trust DC1 explicitly. This step creates no output.

In step 9, you reconfigure the DNS service on DC1 to enable non-secure updates to the Reskit.Org domain. In step 10, you configure SRV2 to register itself within the Reskit.Org zone on DC1. And then, in step 11, you register SRV2's IP address within the DNS service on DC1. These three steps also produce no output.

In the next step, step 12, you pre-stage the SRV2 computer into the Reskit.Org AD. This step, which creates no output at the console, creates the SRV2 computer account within the AD. This means a low privilege user can now add SRV2 to the domain. Also, note that you run this command remotely on DC1 – this is because you have not yet added the AD tools to SRV2.

In the final step, step 13, you query the DNS service to resolve the domain name, SRV2.Reskit.Org. This step produces the following output:

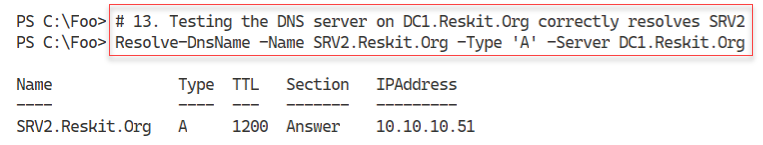


Figure 7.6: Testing the DNS server on DC1.Reskit.Org resolves SRV2

1. Insert image B16762\_07\_06.png

In this recipe, you configured a workgroup server to have a static IP address. You also configured the DNS service to enable SRV2 to register a DNS record within the Reskit.Org domain. In most production scenarios, you would join SRV2 to the domain, in which case DNS registration just works without needing step 7 through step 11.

## There's more...

In step 5, you verify SRV2's IP address. This test does not check SRV2's DNS configuration. To check the DNS address as well, you could use Get-NetIPConfiguration.

In step 7, you create a credential to enable you to run commands on DC1. In this step, you use the Enterprise/Domain Administrator account. In production, a better approach would be to create another user with a subset of the Enterprise Admin's group's permissions, then use that user to perform step 9.

In step 8, you configure WinRM to trust the DNS server, DC1. This configuration is necessary for a workgroup environment because, by default, workgroup computers do not trust other servers when using PowerShell remoting. PowerShell remoting, by default, performs mutual authentication. Kerberos provides mutual authentication in a domain environment, while in a workgroup environment, you could use SSL to connect to DC1. By configuring SRV2 to trust DC1, you are disabling the authentication of DC1 by SRV2. In a protected environment, like where you have your set of Reskit.Org servers, this is acceptable. In production and especially in larger environments, a better approach is to enable SSL for remoting to hosts in separate security realms.

# Testing network connectivity

In today's connected world, network connectivity is vital. When you add a new server to your infrastructure, it is useful to ensure that the server can connect to and use the network.

In this recipe, you perform necessary network connectivity tests on the newly installed SRV2 host. You should ensure that full connectivity exists before adding a server to the domain.

## Getting ready

This recipe uses SRV2, a workgroup host. You gave this host a static IP address in the Configuring IP addressing recipe.

## How to do it...

1. Verifying SRV2 itself is up and that Loopback is working:

Test-Connection -ComputerName SRV2 -Count 1 -IPv4

1. Testing connection to local host's WinRM port:

Test-NetConnection -ComputerName SRV2 -CommonTCPPort WinRM

1. Testing basic connectivity to DC1:

Test-Connection -ComputerName DC1.Reskit.Org -Count 1

1. Checking connectivity to SMB port on DC1:

Test-NetConnection -ComputerName DC1.Reskit.Org -CommonTCPPort SMB

1. Checking connectivity to the LDAP port on DC1:

Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389

1. Examining the path to a remote server on the Internet:

$NCHT = @{

  ComputerName     = 'WWW.Packt.Com'

  TraceRoute       = $true

  InformationLevel = 'Detailed'

}

Test-NetConnection @NCHT    # Check our wonderful publisher

## How it works...

In step 1, you verify that SRV2's Loopback adapter works and that the basic TCP/IP stack is up and working. The output looks like this:

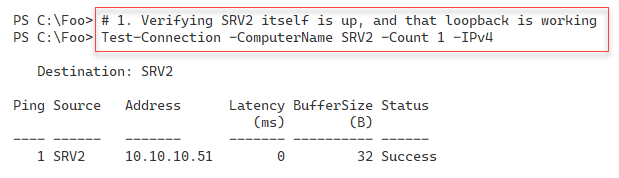


Figure 7.7: Verifying SRV2 itself is up and that loopback is working

1. Insert image B16762\_07\_07.png

In step 2, you check that the WinRM port is open and working, with output like this:

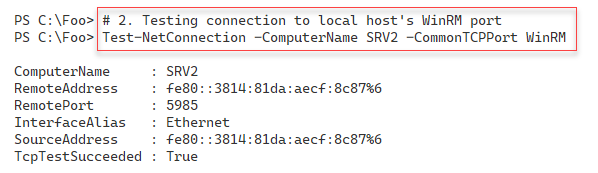


Figure 7.8: Testing connection to the local host's WinRM port

1. Insert image B16762\_07\_08.png

In the Reskit.Org network, DC1 is a domain controller and a DNS server. In step 3, you test the connectivity to this critical enterprise server, with output like this:

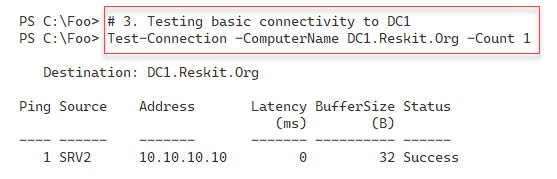


Figure 7.9: Testing basic connectivity to DC1

1. Insert image B16762\_07\_09.png

In any domain environment, hosts need to access the SYSVOL share on a DC to download group policy .POL files. In step 4, you test that SRV2 can access the SMB port, port 445, on the DC, with output like this:

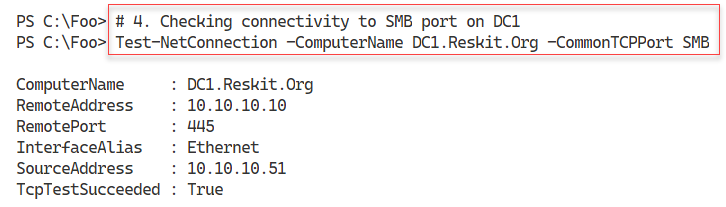


Figure 7.10: Checking connectivity to SMB port on DC1

1. Insert image B16762\_07\_10.png

In step 5, you test that SRV2 can access DC1 on the LDAP port, port 389, with the following output:

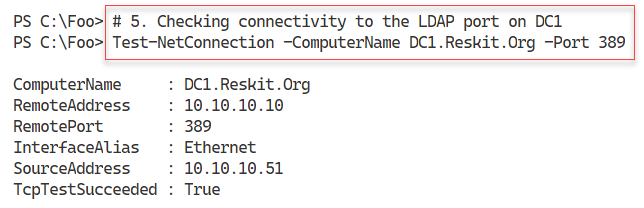


Figure 7.11: Checking connectivity to the LDAP port on DC1

1. Insert image B1672\_01\_11.png

In step 6, you check connectivity to the Internet and test the network path to the publisher's website at www.packt.com. The output is:

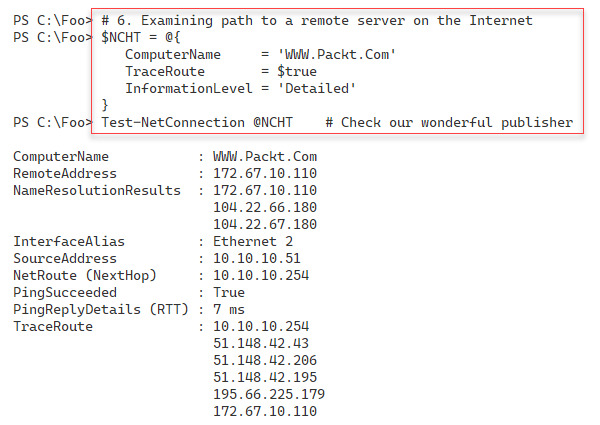


Figure 7.12: Examining the path to a remote server on the Internet

1. Insert image B1672\_01\_12.png

## There's more...

This recipe's steps confirm that the host can accept connections over WinRM and can contact the DC for core activities. You could add several additional tests, such as testing you can access the DNS server and resolve DNS queries.

In step 6, you test the Internet connectivity to our publisher's website (www.packt.com). Since we are just testing the connectivity to this site, you only need to specify the actual computer name and do not need the HTTP or HTTPS prefix.

# Installing DHCP

In previous recipes, you configured SRV2 with a static IP address and tested its connectivity. Each server needs a unique IP address and other configuration options to configure on a server-by-server basis. You can also configure client computers running Windows 10 or other OSes manually, although this can be a huge and challenging task in large organizations.

**Dynamic Host Configuration Protocol** (**DHCP**) enables a DHCP client to get its IP configuration and other networking details automagically from a DHCP server. DHCP automates IP configuration and avoids the work and the inevitable issues involved with manual IP configuration.

Windows and most other client operating systems, including Linux and Apple Macs, have a built-in DHCP client. Windows Server also includes a DHCP server service you can install to provide DHCP services to the clients. You can install DHCP using Server Manager and configure the service using the DHCP GUI application. Alternatively, you can automate the installation of DHCPm as you can see in this recipe. In the next recipe, Configuring DHCP scopes and options, you configure the DHCP service to issue IP addresses in a specific range. You also configure DHCP to provide DHCP clients with other IP address configuration options, such as the subnet mask, default gateway, and the DNS server IP address or addresses.

## Getting ready

This recipe uses DC1, a domain controller in the Reskit.Org domain. You should have installed AD on this host and configured it as per earlier recipes in Chapter 5, Exploring .NET and Chapter 6, Managing Active Directory.

## How to do it...

1. Installing the DHCP feature on DC1 and adding the management tools:

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name DHCP -IncludeManagementTools

1. Adding DC1 to trusted DHCP servers and add the DHCP security group:

Import-Module -Name DHCPServer -WarningAction SilentlyContinue

Add-DhcpServerInDC

Add-DHCPServerSecurityGroup

1. Letting DHCP know it's all configured:

$DHCPHT = @{

Path = 'HKLM:\SOFTWARE\Microsoft\ServerManager\Roles\12'

Name = 'ConfigurationState'

Value = 2

}

Set-ItemProperty @DHCPHT

1. Restarting DHCP server:

Restart-Service -Name DHCPServer -Force

1. Testing service availability:

Get-Service -Name DHCPServer |

Format-List -Property \*

## How it works...

In step 1, you import the ServerManager module and use Install-WindowsFeature to add the DHCP server service to DC1. The output from this step looks like this:

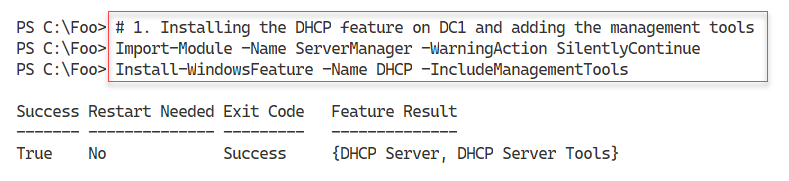


Figure 7.13: Installing the DHCP feature on DC1 and adding the management tools

1. Insert image B16762\_07\_13.png

In step 2, you add DC1 to the set of authorized DHCP servers in the domain and add the DHCP security groups to the DHCP server, which produces no output to the console. The groups that this command adds are the DHCP Users and DHCP Administrators security groups. For more details on these groups, see https://secureidentity.se/delegate-dhcp-admins-in-the-domain/.

In step 3, you set a registry entry to tell Windows that all post-deployment DHCP configuration activities are complete. The GUI installation process takes you through this automatically. When installing via PowerShell, you need to set the registry entry to complete the configuration.

When you have completed the configuration activities, you restart the DHCP service. Once restarted, the DHCP service can issue IP configuration to DHCP clients. For this to happen, you must also have specified the configuration information you specify in the Configuring DHCP scopes and options recipe. Step 2, step 3, and step 4 produce no output.

In step 5, you complete this recipe by ensuring that the DHCP service has started. The output of this step looks like this:

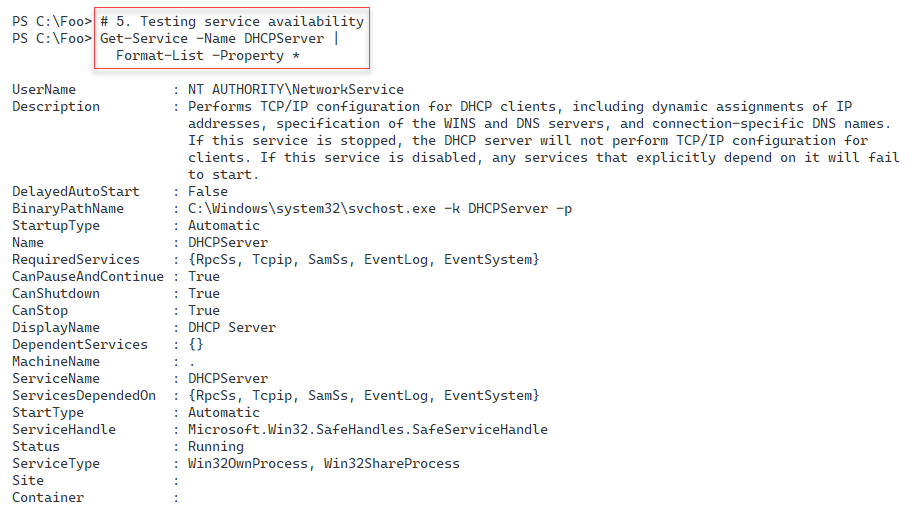


Figure 7.14: Testing service availability

1. Insert image B16762\_07\_14.png

## There's more...

When the Windows DHCP service starts, it checks to ensure the server is on the DHCP server list authorized in the domain. The DHCP service does not start on any non-authorized DHCP server. Adding DC1 to the list of authorized servers can help to guard against rogue DHCP servers.

In step 5, you check the DHCP service. The output from Get-Service includes a description of the service and the path name to the service executable. The DHCP service does not run in its own process. Instead, it runs inside svchost.exe. It is for this reason that you do not see the service explicitly when you use Get-Process.

# Configure DHCP scopes and options

Installing DHCP is simple, as you see in the Installing DHCP recipe. You add the Windows feature and then carry out two small configuration steps. In most cases, you probably do not need to take these extra steps. The extra steps enable you to use the relevant security groups and avoid the Server Manager GUI message stating that there are configuration steps that have not been performed yet.

Before your DHCP server can provide IP address configuration information to DHCP clients, you need to create a DHCP scope and DHCP options. A DHCP scope is a range of DHCP addresses that your DHCP server can give out for a given IP subnet. DHCP options are specific configuration options your DHCP server provides, such as the DNS server's IP address and the IPv4 default gateway.

You can set DHCP options at a scope level or a server level, depending on your organization's needs. For example, you would most likely specify a default gateway in the scope options, with DNS server address(es) set at the server level.

In this recipe, you create a new scope for the 10.10.10.0/24 subnet and specify both scope- and server-level options.

## Getting ready

You run this recipe on DC1, a domain controller in the Reskit.Org domain, after installing the DHCP server service. You must have installed PowerShell 7 and VS Code on this host.

## How to do it...

1. Importing the DHCP server module:

Import-Module DHCPServer -WarningAction SilentlyContinue

1. Creating an IPv4 scope:

$SCOPEHT = @{

Name = 'ReskitOrg'

StartRange = '10.10.10.150'

EndRange = '10.10.10.199'

SubnetMask = '255.255.255.0'

ComputerName = 'DC1.Reskit.Org'

}

Add-DhcpServerV4Scope @SCOPEHT

1. Getting IPV4 scopes from the server:

Get-DhcpServerv4Scope -ComputerName DC1.Reskit.Org

1. Setting server-wide option values:

$OPTION1HT = @{

ComputerName = 'DC1.Reskit.Org' # DHCP Server to Configure

DnsDomain = 'Reskit.Org' # Client DNS Domain

DnsServer = '10.10.10.10' # Client DNS Server

}

Set-DhcpServerV4OptionValue @OPTION1HT

1. Setting a scope-specific option:

$OPTION2HT = @{

ComputerName = 'DC1.Reskit.Org' # DHCP Server to Configure

Router = '10.10.10.254'

ScopeID = '10.10.10.0'

}

Set-DhcpServerV4OptionValue @OPTION2HT

1. Viewing DHCP server options:

Get-DhcpServerv4OptionValue | Format-Table -AutoSize

1. Viewing scope-specific options:

Get-DhcpServerv4OptionValue -ScopeId '10.10.10.10' |

Format-Table -AutoSize

1. Viewing DHCPv4 option definitions:

Get-DhcpServerv4OptionDefinition | Format-Table -AutoSize

## How it works...

In step 1, you import the DHCPServer module. When you installed DHCP (in the Installing DHCP recipe), you added the management tools, including this module. However, the DHCP team has not yet made this module compatible with PowerShell 7. This step, which produces no output, loads the module using the Windows PowerShell compatibility solution. You read about the Windows PowerShell compatibility solution in Chapter 3.

In step 2, you create a new DHCP scope for IPV4 addresses. The scope enables the DHCP server to issue IP addresses in the 10.10.10.150 - 10.10.10.199 range. This step produces no output.

In step 3, you use Get-DHCPServerIPV4Scope to retrieve details of all the DHCP scopes you have defined on DC1. The output of this step looks like this:

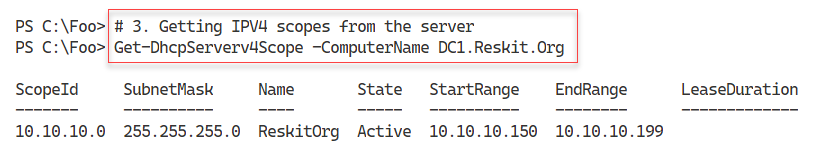


Figure 7.15: Getting IPV4 scopes from the server

1. Insert image B16762\_07\_15.png

In step 4, you set two server-wide DHCP options, creating no output. These are options and values offered to all clients of any DHCP scope defined on this server. In step 5, you specify a scope option. These two steps produce no output.

In step 6, you view the DHCP server-wide options, with output that looks like this:

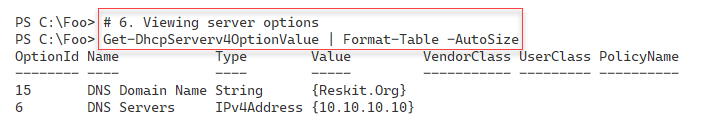


Figure 7.16: Viewing server options

1. Insert image B16762\_07\_16.png

With step 7, you view the options you have set on the 10.10.10.10 scope, which looks like this:

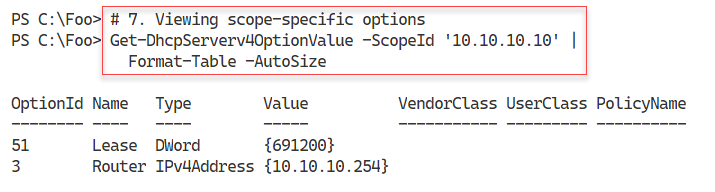


Figure 7.17: Viewing scope-specific options

1. Insert image B16762\_07\_17.png

There are 66 DHCP options you can use to provide option values to DHCP clients. Most of these options are of little use in most cases but provide support for niche and uncommon scenarios. You view the set of options defined by default in step 8, which looks like this:

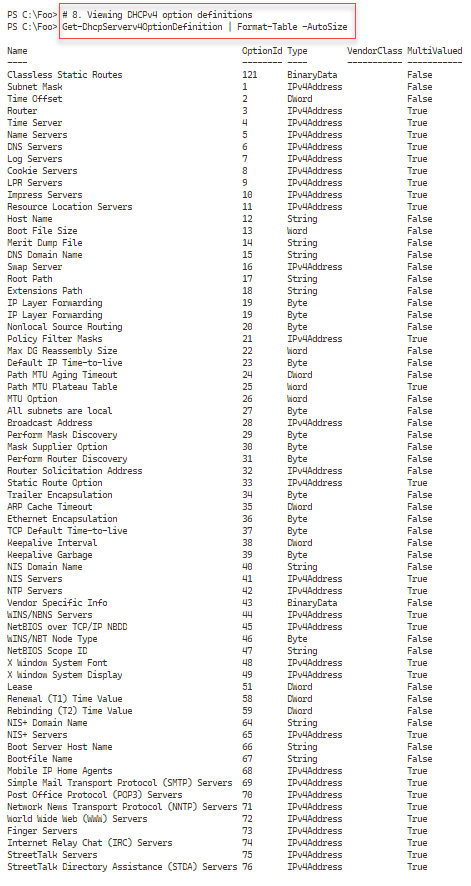


Figure 7.18: Viewing DHCPv4 option definitions

1. Insert image B16762\_07\_18.png

## There's more...

In step 2, you create a new scope and give it a scope name. However, as you can see in step 5 and elsewhere, the DHCP cmdlets do not provide a -DHCPScopeName parameter. Instead, you specify a "ScopeID." In general, this is the subnet for the IP addresses in the scope, 10.10.10.0/24. But even then, as you can see in step 7, the cmdlet accepts any IP address in the 10.10.10.0/24 subnet as the subnet ID, including 10.10.10.10 as shown.

# Using DHCP

After installing the DHCP service and configuring the scope(s) and option values, your DHCP services can issue IP configuration data to any client. Since the DHCP protocol acts at the IP level, the protocol performs no authentication when any DHCP client uses the protocol to request IP configuration details. That means that any client you attach to the physical subnet can ask for and receive IP confirmation details.

In the Configuring IP addressing recipe, you set a static IP address for SRV2. In this recipe, you reconfigure this server to obtain a DHCP-based IP address (and the options you set in the Configuring DHCP scopes and options recipe).

## Getting ready

You run this recipe on SRV2, which you've reconfigured to get its address via DHCP. You also need DC1, a domain controller for the Reskit.Org domain, and a DHCP server that you set up and configured in earlier recipes in this chapter.

## How to do it...

1. Adding DHCP RSAT tools

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name RSAT-DHCP

1. Importing the DHCP module

Import-Module -Name DHCPServer -WarningAction SilentlyContinue

1. Viewing the scopes on DC1

Get-DhcpServerv4Scope -ComputerName DC1

1. Getting V4 scope statistics from DC1

Get-DhcpServerv4ScopeStatistics -ComputerName DC1

1. Discovering a free IP address

Get-DhcpServerv4FreeIPAddress -ComputerName DC1 -ScopeId 10.10.10.42

1. Getting SRV2 NIC configuration

$NIC = Get-NetIPConfiguration -InterfaceIndex 6

1. Getting IP interface details

$NIC |

  Get-NetIPInterface  |

    Where-Object AddressFamily -eq 'IPv4'

1. Enabling DHCP on the NIC

$NIC |

  Get-NetIPInterface  |

    Where-Object AddressFamily -eq 'IPv4' |

      Set-NetIPInterface -Dhcp Enabled

1. Checking IP address assigned

Get-NetIPAddress -InterfaceAlias "Ethernet"   |

  Where-Object AddressFamily -eq 'IPv4'

1. Getting updated V4 scope statistics from DC1

Get-DhcpServerv4ScopeStatistics -ComputerName DC1

1. Discovering the next free IP address:

Get-DhcpServerv4FreeIPAddress -ComputerName DC1 -ScopeId 10.10.10.42

1. Checking IPv4 DNS name resolution:

Resolve-DnsName -Name SRV2.Reskit.Org -Type A

## How it works...

In step 1, you install the RSAT-DHCP feature to add the DHCP server's PowerShell module on SRV1, with output like this:

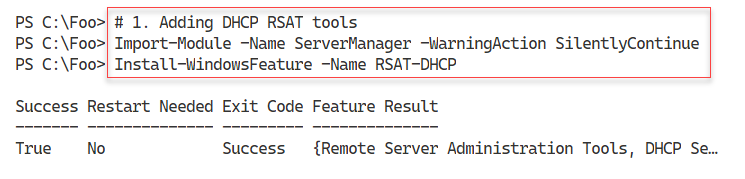


Figure 7.19: Adding DHCP RSAT tools

1. Insert image B16762\_07\_19.png

The DHCP server module is not .NET Core compatible. In step 2, you explicitly load this module using Import-Module, which creates no output.

In step 3, you look at the scopes available on DC1 (the DHCP server you installed in the Installing DHCP recipe). The output looks like this:

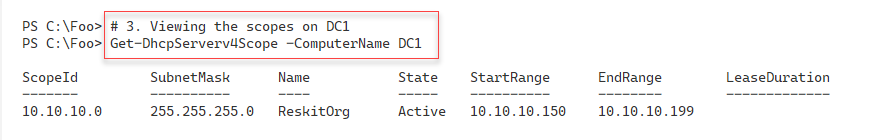


Figure 7.20: Viewing the scopes on DC1

1. Insert image B16762\_07\_20.png

In step 4, you examine the scope statistics for the DHCP scope you created in the Configuring DHCP scopes and options recipe, which produces output like this:

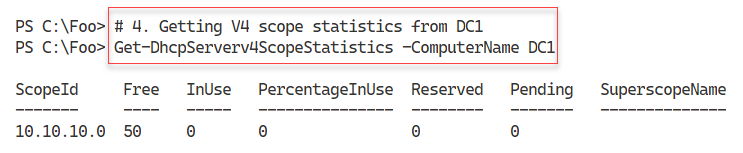


Figure 7.21: Getting V2 scope statistics from DC1

1. Insert image B16762\_07\_21.png

In step 5, you use the Get-DhcpServerv4FreeIPAddress cmdlet to find the first available free IP address in the scope. The output resembles this:

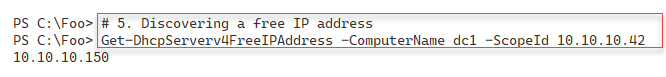


Figure 7.22: Discovering a free IP address in the scope

1. Insert image B16762\_07\_22.png

In step 6, you get the NIC details and store them in the $NIC variable, producing no output. Note that you specify an InterfaceIndex of 6, which should be your VM's NIC.

In step 7, you use that variable to get the details of the NIC, with output like this:

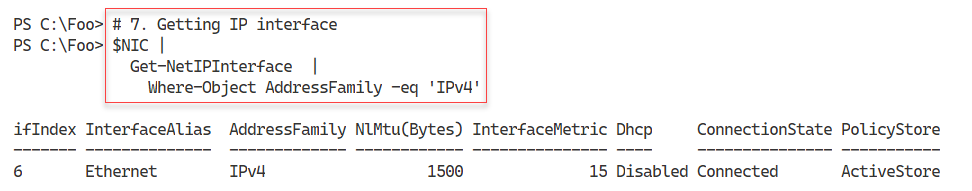


Figure 7.23: Getting the IP interface details

1. Insert image B16762\_07\_23.png

In step 8, you change the NIC to get configuration details from the DHCP server. This step creates no output. In step 9, you view the NIC's IPV4 address; this time, one assigned by DHCP. The output looks like this:

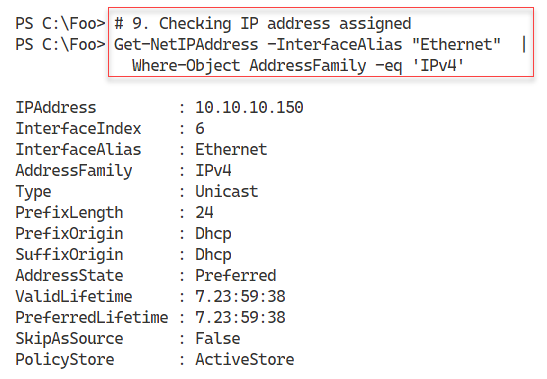


Figure 7.24: Checking IP address assigned

1. Insert image B16762\_07\_24.png

In step 10, you re-examine the scope statistics, with output like this:

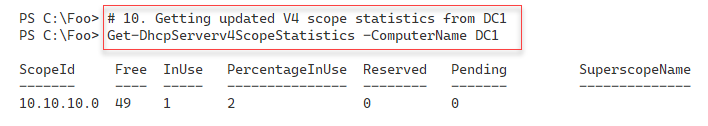


Figure 7.25: Getting updated V4 scope statistics from DC1

1. Insert image B16762\_07\_25.png

With step 11, you recheck and discover the next free IP address in the DHCP scope, with output like this:

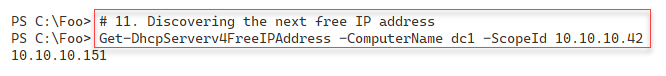


Figure 7.26: Discovering the next free IP address

1. Insert image B16762\_07\_26.png

In the final step, you check that SRV2 has registered its new IP address in the DNS server on DC1. The output looks like:

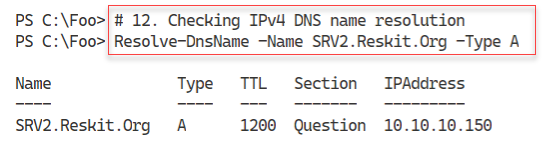


Figure 7.27: Checking IPv4 DNS name resolution

1. Insert image B16762\_07\_27.png

## There's more...

In this recipe, you use the DHCP server cmdlets to get information from the DHCP server on DC1. These cmdlets show how you can obtain information from the DHCP server, including the next free IP address and statistics on the scope (free/used addresses and more).

In step 7, you get the IP interface details to allow you, in step 8, to convert the NIC from a static IP address into a dynamic one, based on DHCP.

In step 9, you view the DHCP supplied IP address information for SRV2. If you perform step 8 and immediately run step 9, you may find that the NIC shows an **Automatic Private IP Addressing** (**APIPA**) IP address in the 169.254.0.0/24 subnet. This address is transient. When you change to DHCP (as you did in step 8), Windows removes the static address and creates an APIPA address. Once SRV2 contacts the DHCP server and negotiates an address, you see the output shown for step 9, as in *Figure 7.24*. Obtaining a lease can take a few seconds, so be patient.

# Implementing DHCP failover and load balancing

As shown in the two previous recipes, the installation and configuration of a single on-premises DHCP server is straightforward. However, a single DHCP server represents a single point of failure, which is never a good thing. The solution is always to have a second DHCP server. In earlier versions of Windows, you could do this with two DHCP servers, each with a scope. Typically, you used to split the full set of IP addresses and allow each server to have part of that set. The traditional “wisdom” was to do an 80/20 split (have 80% of the scope supplied by your primary DHCP server and 20% on the backup server).

Independent DHCP servers are an error-prone approach and were never ideal since these independent servers did not coordinate scope details. That 80/20 "rule" was a recommendation for one specific customer scenario (a large firm in the Pacific Northwest) and possibly was not meant to become a best practice.

In Windows Server 2012, Microsoft added a DHCP failover and load balancing mechanism that simplified deploying DHCP in an organization. You can now set up two DHCP servers, define a DHCP scope, and allow both servers to work in tandem.

In this recipe, you install DHCP on a second server, DC2, and then configure and use the failover and load balancing capabilities of Windows Server.

## Getting ready

This recipe uses the two DCs you have installed, DC1 and DC2. You should also have installed DHCP on DNS (Installing DCHP) and configured a DNS zone (Configuring DHCP scopes and options). You run this recipe on DC2.

## How to do it...

1. Installing the DHCP server feature on DC2

Import-Module -Name ServerManager -WarningAction SilentlyContinue

$FEATUREHT = @{

  Name                   = 'DHCP'

  IncludeManagementTools = $True

}

Install-WindowsFeature @FEATUREHT

1. Letting DHCP know it is fully configured

$IPHT = @{

  Path  = 'HKLM:\SOFTWARE\Microsoft\ServerManager\Roles\12'

  Name  = 'ConfigurationState'

  Value = 2

}

Set-ItemProperty @IPHT

1. Authorizing the DHCP server in AD

Import-Module -Name DHCPServer -WarningAction 'SilentlyContinue'

Add-DhcpServerInDC -DnsName DC2.Reskit.Org

1. Viewing authorized DHCP servers in the Reskit domain

Get-DhcpServerInDC

1. Configuring failover and load balancing

$FAILOVERHT = @{

  ComputerName       = 'DC1.Reskit.Org'

  PartnerServer      = 'DC2.Reskit.Org'

  Name               = 'DC1-DC2'

  ScopeID            = '10.10.10.0'

  LoadBalancePercent = 60

  SharedSecret       = 'j3RryIsTheB3est!'

  Force              = $true

  Verbose            = $True

}

Invoke-Command -ComputerName DC1.Reskit.org -ScriptBlock {

  Add-DhcpServerv4Failover @Using:FAILOVERHT

}

1. Getting active leases in the scope (from both servers!)

$DHCPServers = 'DC1.Reskit.Org', 'DC2.Reskit.Org'

$DHCPServers |

  ForEach-Object {

    "Server $\_" | Format-Table

    Get-DhcpServerv4Scope -ComputerName $\_ | Format-Table

  }

1. Viewing DHCP server statistics from both DHCP servers

$DHCPServers |

  ForEach-Object {

    "Server $\_" | Format-Table

    Get-DhcpServerv4ScopeStatistics -ComputerName $\_  | Format-Table

  }

## How it works...

Using the first three steps in this recipe, you install the DHCP server feature on DC2. These steps duplicate the approach you used in the Installing DHCP recipe to install DHCP on DC1.

In step 1, you import the Server Manager module and then install the DHCP feature, with output like this:

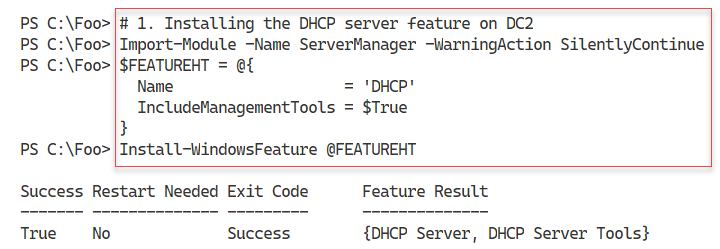


Figure 7.28: Installing the DHCP server feature on DC2

1. Insert image B1672\_01\_28.png

In step 2, you set a registry key to tell Windows that you have completed the DHCP feature installation. Without this step, any time you use the Server Manager GUI on DC2, you will see a message indicating that additional configuration is required. This step produces no output.

In step 3, you import the DHCP server PowerShell module and add DC2 to the list of authorized DHCP servers in the Reskit domain. This step produces no output.

In step 4, you examine the set of authorized DHCP servers, which produces the following output:



Figure 7.29: Viewing authorized DHCP servers in the Reskit domain

1. Insert image B1672\_01\_29.png

In step 5, you configure DHCP load balancing and failover, with output like this:

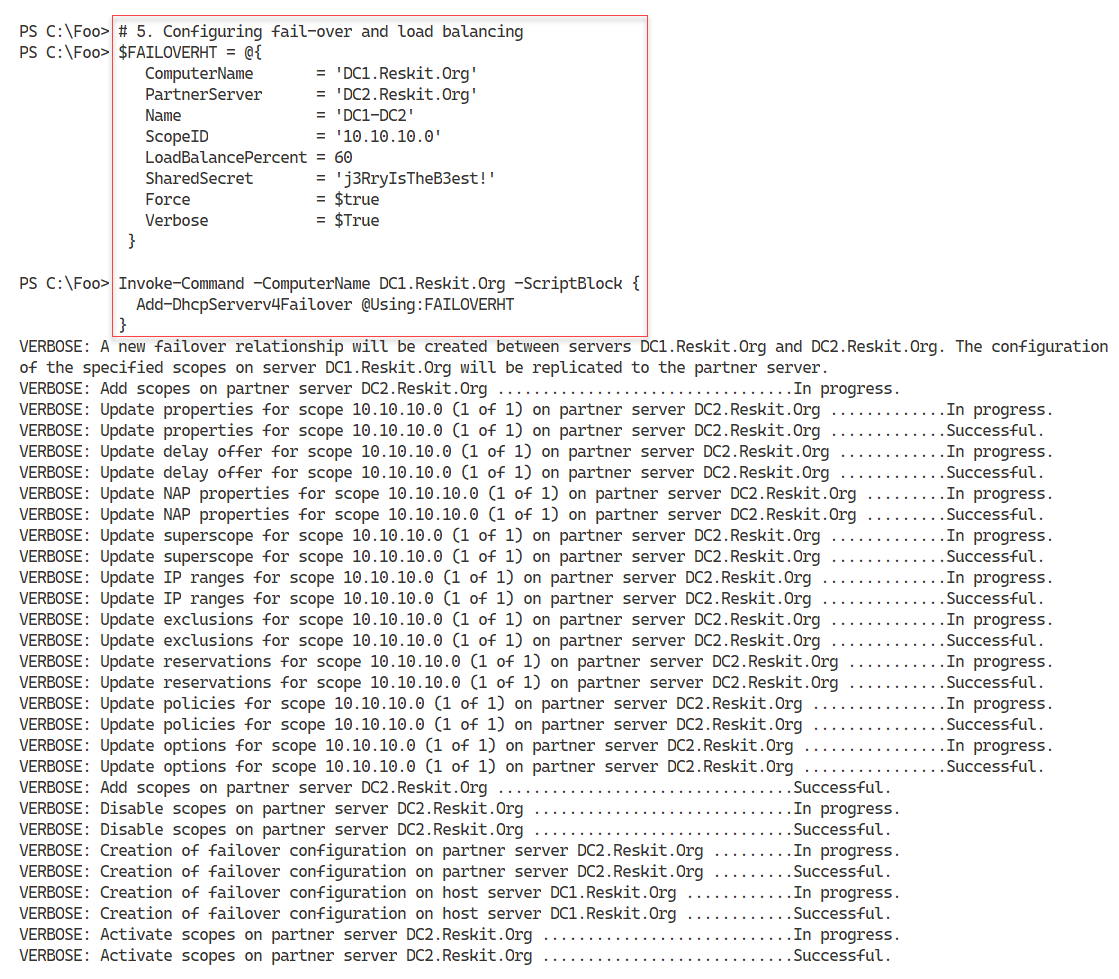


Figure 7.30: Configuring failover and load balancing

1. Insert image B16762\_07\_30.png

With step 6, you examine the scopes on both DHCP servers, with the following output:

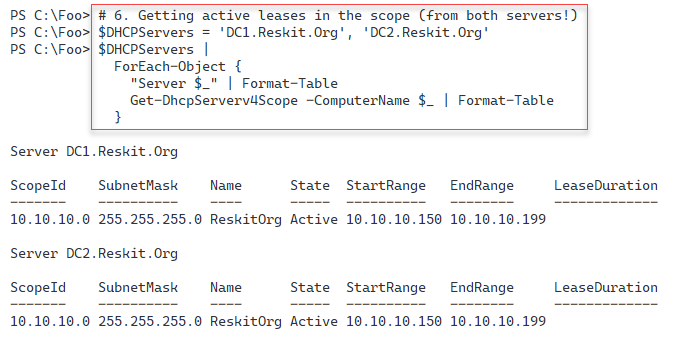


Figure 7.31: Getting active leases in the scope from both servers

1. Insert image B16762\_07\_31.png

In step 7, you view the DHCP server statistics from your two DHCP servers, with output like this:

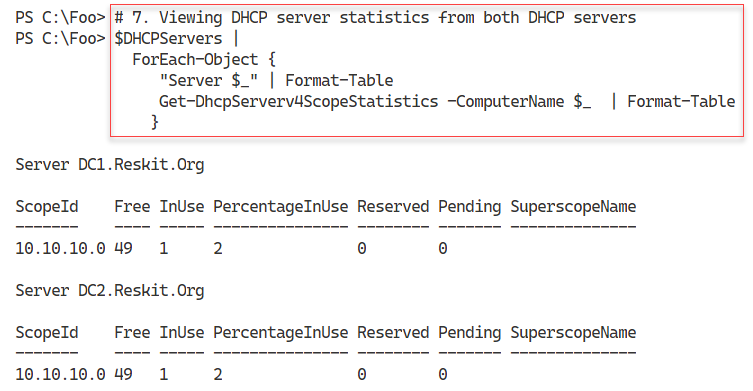


Figure 7.32: Viewing DHCP statistics from both servers

1. Insert image B16762\_07\_32.png

## There's more...

Unlike in the Installing DHCP recipe, in this recipe, you do not add the DHCP-related local security groups on DC2. If you plan to delegate administrative privileges, you may wish to add those groups, as you did in the earlier recipe.

In step 5, you establish a load balancing and failover relationship between the two DHCP servers. By using the -Verbose switch for the Add-DhcpServerV4Failover cmdlet, you can see precisely what the command is doing, step by step. As you can see in the output in *Figure 7.30*, this command copies full details of all the scopes on DC1 to DC2. You should note that the relationship is on a scope-by-scope basis.

Depending on your needs, you can configure different sorts of relationships between DHCP servers using the -ServerRole parameter. For more details on this parameter and others that you can use to fine-tune the relationship between the two partners, see https://docs.microsoft.com/powershell/module/dhcpserver/add-dhcpserverv4failover.

The DHCP failover feature provides several additional settings to control precisely when to failover and for how long (and when to failback). These settings allow you to control, for example, what might happen during a planned reboot of your DHCP server.

# Deploying DNS in the Enterprise

When you installed Active Directory in Chapter 6, you added a single DNS server on DC1. When you added a replica DC, DC2, and when you added the child domain with UKDC1, you did not set up any additional DNS servers in your forest. In an enterprise organization, this is not best practice. You always want to configure your clients and servers so that they use at least two DNS servers. For servers with a static DNS setting, you should also update the DHCP DNS server option settings to ensure that your DHCP servers also provide two DNS server entries to DHCP clients.

In most organizations, there are several DNS service configuration options you may wish to set. These include whether to allow DNS server recursion on the server, the maximum size of the DNS cache, and whether to use **Extended DNS** (**EDNS**).

EDNS (also referred to as EDNS0 or, more recently, EDNS(0)) is an extension mechanism that enables more recent DNS servers to interact with older servers that may not be capable of specific actions. For more details on EDNS(0), see https://en.wikipedia.org/wiki/Extension\_mechanisms\_for\_DNS#:~:text=Extension%20mechanisms%20for%20DNS%20(EDNS,increasing%20functionality%20of%20the%20protocol.

This recipe is a starting point. There are other DNS server options you may wish to consider updating, such as DNS security. Additionally, you may need to reconfigure other servers and update the static IP address settings (to point to both DNS servers). And finally, in the Reskit.Org forest, you should also be updating the child domain UK.Reskit.Org.

In this recipe, you update the domain-wide DNS settings in the Reskit.Org domain. These settings include ensuring that you configure DC2's DNS service in the same way as DC1, updating DNS server configuration settings, configuring both DC1 and DC2 to point to the correct DNS servers, and updating DHCP to issue both DNS server addresses. You then examine the DNS settings.

In the Configuring IP addressing recipe, you configured SRV2, a workgroup host, to update DNS. You configured the Reskit.Org zone on DC1 to allow non-domain members to update their DNS registrations using dynamic DNS. This configuration is not a best practice configuration. In the final steps of this recipe, you update the DNS zones for Reskit.Org to allow only secure updates and join SRV2 to the Reskit.Org domain with a static IP address.

## Getting ready

You run this recipe on DC2, with DC1 and SRV2 also operational. You run this recipe after setting up DNS on both DC1 and DC2 and implementing DHCP.

## How to do it...

1. Installing the DNS feature on DC2

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name DNS -IncludeManagementTools

1. Creating a script block to set DNS server options

$SB1 = {

  # Enable recursion on this server

  Set-DnsServerRecursion -Enable $true

  # Configure DNS Server cache maximum size

  Set-DnsServerCache  -MaxKBSize 20480  # 28 MB

  # Enable EDNS

  $EDNSHT = @{

    EnableProbes    = $true

    EnableReception = $true

  }

  Set-DnsServerEDns @EDNSHT

  # Enable Global Name Zone

  Set-DnsServerGlobalNameZone -Enable $true

}

1. Reconfiguring DNS on DC2 and DC1

Invoke-Command -ScriptBlock $SB1

Invoke-Command -ScriptBlock $SB1 -ComputerName DC1

1. Creating ascript block to configure DC2 to have two DNS servers

$SB2 = {

  $NIC =

     Get-NetIPInterface -InterfaceAlias "Ethernet" -AddressFamily IPv4

  $DNSSERVERS = ('127.0.0.1','10.10.10.10')

  $DNSHT = @{

    InterfaceIndex  = $NIC.InterfaceIndex

    ServerAddresses = $DNSSERVERS

  }

  Set-DnsClientServerAddress @DNSHT

  Start-Service -Name DNS

}

1. Configuring DC2 to have two DNS servers

Invoke-Command -ScriptBlock $SB2

1. Creating a script block to configure DC1 to have two DNS servers

$SB3 = {

  $NIC =

    Get-NetIPInterface -InterfaceAlias "Ethernet" -AddressFamily IPv4

  $DNSSERVERS = ('127.0.0.1','10.10.10.11')

  $DNSHT = @{

    InterfaceIndex  = $NIC.InterfaceIndex

    ServerAddresses = $DNSSERVERS

  }

  Set-DnsClientServerAddress @DNSHT

  Start-Service -Name DNS

}

1. Configuring DC1 to have two DNS servers

Invoke-Command -ScriptBlock $SB3 -ComputerName DC1

1. Updating DHCP scope to add two DNS entries

$DNSOPTIONHT = @{

  DnsServer    = '10.10.10.11',

                 '10.10.10.10'    # Client DNS Servers

  DnsDomain    = 'Reskit.Org'

  Force        = $true

}

Set-DhcpServerV4OptionValue @DNSOPTIONHT -ComputerName DC1

Set-DhcpServerV4OptionValue @DNSOPTIONHT -ComputerName DC2

1. Getting DNS service details

$DNSRV = Get-DNSServer -ComputerName DC2.Reskit.Org

1. Viewing recursion settings

$DNSRV |

  Select-Object -ExpandProperty ServerRecursion

1. Viewing server cache settings

$DNSRV |

  Select-Object -ExpandProperty ServerCache

1. Viewing EDNS settings

$DNSRV |

  Select-Object -ExpandProperty ServerEdns

1. Setting Reskit.Org zone to be secure only

$DNSSSB = {

  $SBHT = @{

    Name          = 'Reskit.Org'

    DynamicUpdate = 'Secure'

  }

  Set-DnsServerPrimaryZone @SBHT

}

Invoke-Command -ComputerName DC1 -ScriptBlock $DNSSSB

Invoke-Command -ComputerName DC2 -ScriptBlock $DNSSSB

**Note:** Run the next step on SRV2.

1. Adding SRV2 to domain

$User = 'Reskit\Administrator'

$Pass = 'Pa$$w0rd'

$PSS = $Pass | ConvertTo-SecureString -Force -AsPlainText

$CRED = [PSCredential]::new($User,$PSS)

$Sess = New-PSSession -UseWindowsPowerShell

Invoke-Command -Session $Sess -Scriptblock {

$ACHT = @{

Credential = $using:Cred

Domain = 'Reskit.org'

Force = $True

}

Add-Computer @ACHT

Restart-Computer

} | Out-Null

## How it works...

In step 1, you install the DNS feature on DC1, with output like this:

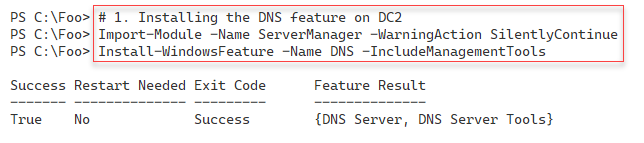


Figure 7.33: Installing the DNS feature on DC2

1. Insert image B16762\_07\_33.png

In step 2, you create a script block that contains the DNS server options. In step 3, you run this script block on both DC1 and DC2. In step 4, you create a script block to use to set the IP configuration on DC2. In step 5, you run this script block to reconfigure DC2. In step 6 and step 7, you reconfigure the static IP setting on DC1. In step 8, you update the DHCP server's DHCP options to ensure the DHCP server on both hosts supplies two DNS server IP addresses to clients of either DHCP server. These seven steps produce no output.

In step 9, you get the DNS server details from DC2, producing no output. In step 10, you examine the DNS server recursion settings, with output like this:

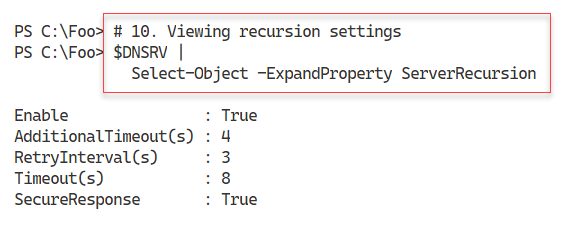


Figure 7.34: Viewing DNS server recursion settings

1. Insert image B1672\_01\_34.png

In step 11, you examine the DNS server cache settings on DC2, with output like this:

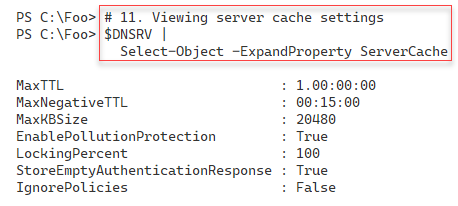


Figure 7.35: Viewing server cache settings

1. Insert image B1672\_01\_35.png

In step 12, you view the EDNS settings on DC2, with output like this:

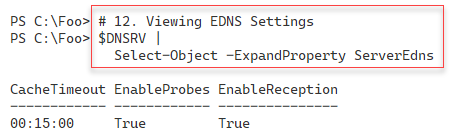


Figure 7.36: Viewing EDNS settings on DC2

1. Insert image B1672\_01\_36.png

In step 13, you change the DNS domain for Reskit.Org so that it only accepts secure dynamic updates on both DNS servers in the domain.

In the final step in thie recipe, in step 14, you add SRV2 to the Reskit.Org domain. This step produces no output.

## There's more...

In this recipe, you install and configure DNS on DC2. You also set DNS server options on both DNS servers (DC1 and DC2). In step 5 and step 7, you configure your domain controllers so that they point to themselves first and to the other DNS servers second. Configuring DNS in this manner is a best practice.

In an earlier recipe, you configured your DHCP servers so that they provided a single DNS server IP address to DHCP clients. In step 8, you update DHCP to provide two DNS server IP addresses.

You then obtain and view the DNS server recursion, server cache, and EDNS settings on DC2. You could extend these steps by examining the settings on DC2 and comparing them to ensure consistent configuration.

In the Configuring IP addressing recipe, you used a workgroup computer, SRV2, and enabled it to register with DNS by setting the domain to accept insecure updates. Using non-secure DNS updates is not a best practice as it could allow a rogue computer to "steal" a real DNS name. In step 13, you ensure that the Reskit.Org DNS domain only allows secure updates. Then, in step 14 and step 15, you add SRV2 to the domain. For the remainder of this book, SRV2 is a domain-joined computer and not a workgroup system. These last three steps produce no output.

In this recipe, you set up DNS for use in the Reskit.Org domain. You made no changes to DNS with respect to the UK.Reskit.Org sub-domain. In production, you would want at least a second DC and would need to update your DNS server address settings for servers in the UK domain.

Another production aspect regards replication of domains. In this chapter, you created a DNS zone for Reskit.Org. The recipes you use set that domain to be AD integrated and to be replicated to every DC in the Reskit.Org forest. This means that all DNS server changes to the Reskit.Org domain are replicated automatically to UKDC1. Depending on your usage scenario, you may wish to adjust the zone replication scope for AD-integrated zones.

# Configuring DNS forwarding

When a DNS server gets a query for a **resource record** (**RR**) not held by the server, it can use recursion to discover a DNS server that can resolve the RR. If, for example, you use Resolve‑DNSName to resolve www.packt.com, the configured DNS server may not hold a zone that would help. Your DNS service then looks to the DNS root servers to discover a DNS server that can via the recursion process. Eventually, the process finds a DNS server that can resolve the RR. Your DNS server then caches these details locally in the DNS server cache.

If you are resolving publicly available names, this process works great. But you might have internally supplied DNS names that the DNS can't resolve via the mechanism. An example might be when two companies merge. There may be internal hostnames (for example, intranet.kapoho.com and intranet.reskit.org) that your organization's internal DNS servers can resolve but are not available from publicly facing DNS servers. In that scenario, you can set up conditional forwarding. Conditional forwarding enables one DNS server to forward a query to another DNS server or set of servers and not use recursion. You can learn a bit more about conditional forwarding here: https://medium.com/tech-jobs-academy/dns-forwarding-and-conditional-forwarding-f3118bc93984.

An alternative to using conditional forwarding is to use stub zones. You can learn more about the differences between conditional forwarding and stub zones here: https://blogs.msmvps.com/acefekay/2018/03/20/what-should-i-use-a-stub-conditional-forwader-forwarder-or-secondary-zone/.

## Getting ready

In this recipe, you use DC1, a domain controller and DNS server for Reskit.Org. You have previously promoted DC1 to be a DC and installed/configured DNS by following the recipes earlier in this chapter and book.

## How to do it...

* + - 1. Obtaining the IP addresses of DNS servers for packt.com

$NS = Resolve-DnsName -Name packt.com -Type NS |

  Where-Object Name -eq 'packt.com'

$NS

* + - 1. Obtaining the IPV4 addresses for these hosts

$NSIPS = foreach ($Server in $NS) {

  (Resolve-DnsName -Name $Server.NameHost -Type A).IPAddress

}

$NSIPS

* + - 1. Adding conditional forwarder on DC1

$CFHT = @{

  Name          = 'Packt.Com'

  MasterServers = $NSIPS

}

Add-DnsServerConditionalForwarderZone @CFHT

* + - 1. Checking zone on DC1

Get-DnsServerZone -Name Packt.Com

* + - 1. Testing conditional forwarding

Resolve-DNSName -Name WWW.Packt.Com -Server DC1 |

 Format-Table

## How it works...

In step 1, you resolve the name servers serving the packt.com domain on the Internet, and then you display the results, like this:

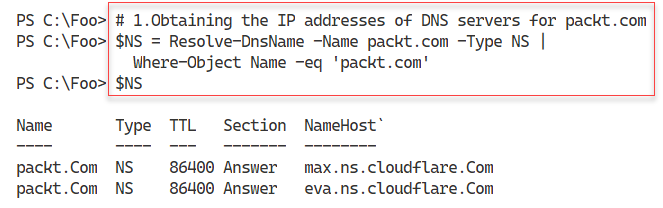


Figure 7.37: Obtaining the IP addresses of the DNS servers for packt.com

1. Insert image B16762\_07\_37.png

In step 2, you use the DNS servers you just retrieved and resolve their IPv4 addresses from the hostnames (which you got in step 1). This step produces the following output:

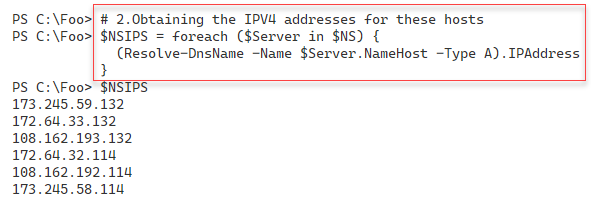


Figure 7.38: Obtaining the IPv4 addresses for hosts

1. Insert image B16762\_07\_38.png

In step 3, which generates no output, you create a DNS forwarding zone for packt.com, which you populate with the IP addresses returned in step 2.

In step 4, you view the conditional forwarder domain defined on DC1, with output like this:

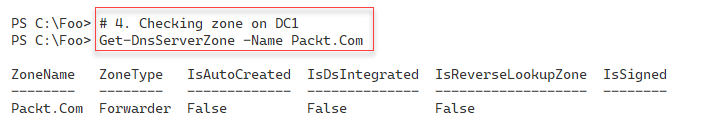


Figure 7.39: Checking the zone on DC1

1. Insert image B16762\_07\_39.png

With the final step, step 5, you resolve www.packt.com, which returns both IPv4 and IPv6 addresses, like this:

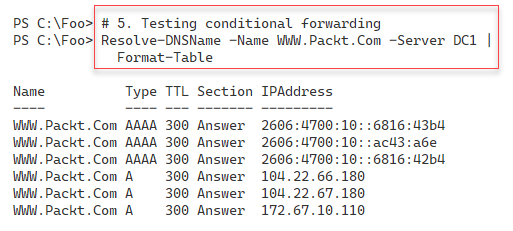


Figure 7.40: Testing conditional forwarding

1. Insert image B1672\_01\_40.png

## There's more...

In step 1, you discover the name server names for the DNS servers that serve packt.com. In this case, these servers are part of Cloudflare's distributed DNS service. For more information on how this service works, see https://www.cloudflare.com/dns/.

In step 2, you resolve those DNS server names into IP addresses. In step 3, you create a conditional forwarding domain that forwards queries for packt.com (such as www.packt.com) to one of the six IP addresses you saw in step 2.

In step 4, you view the conditional forwarding zone on DC1. Since the zone is not DS integrated, DNS does not replicate it to DC2. In production, you should repeat step 3 on DC2.

In step 5, you resolve www.packt.com via conditional forwarding. Since the name servers you discovered support IPv6 AAAA address records, this step returns both IPv4 and IPv6 addresses. If you are in an environment that supports IPv6, you should consider modifying step 2 so that it returns the IPv4 and IPv6 addresses and then use them in step 3.

# Managing DNS zones and resource records

The DNS service enables you to resolve names to other information. Most DNS usage resolves a hostname to its IP (IPv4 or IPv6) addresses. But there are other resolutions, such as determining email servers or for anti-spam, that also rely on DNS.

DNS servers hold zones. A DNS zone is a container for a set of **Resource Records** (**RRs**) related to a specific DNS domain. When you enter www.packt.com, your browser uses DNS to resolve that website name into an IP address and contacts the server at that IP address. If you use an email client to send mail, for example, to DoctorDNS@Gmail.Com, the email client uses DNS to discover an email server to which to send the mail.

Before you can use DNS to hold a RR, you must first create a DNS forward lookup zone. A zone is one part of the global (or internal) DNS namespace. You can configure different zones to hold different parts of your namespace. In Chapter 6, Managing Active Directory, you added a child domain to the Reskit.Org forest, UK.Reskit.Org. You could, for example, have one zone holding RRs for Reskit.Org on DC1 and DC2, while delegating to a new zone, Uk.Reskit.Org, on UKDC1.

A reverse lookup zone is one that does the reverse – you use it to obtain a hostname from its IP address. You may find reverse lookup zones useful, but you do not need them for most domain usage. The old nslookup.exe command is one tool that uses the reverse lookup zone, for example. For more details on DNS in Windows, see https://docs.microsoft.com/windows-server/networking/dns/dns-top.

Traditionally, DNS stores and obtains zone details from files on the DNS server. When the DNS service starts up, it reads these zone files and updates the files as needed when it receives updates. If you have multiple DNS servers, you need to configure your service to perform replication to ensure all DNS servers are in sync.

With Windows 2000, Microsoft added AD-integrated zones. DNS stores the DNS data for these zone types within AD. These zones replicate their zone data via AD replication, which simplifies the setup. This also means that when you created the AD service on DC1, DNS created a zone in the DNS server on DC1 and replicated the zone data to all DCs in a forest. For more information on AD-integrated zones, see https://docs.microsoft.com/windows-server/identity/ad-ds/plan/active-directory-integrated-dns-zones.

In this recipe, you create a DNS forward and reverse look zone and then create RRs in those zones. Once added, you test DNS resolution.

## Getting ready

You run this recipe on DC1, but you need both DC1 and DC2 online. You have installed AD and DNS on both servers, and by doing so, you created a single forward lookup zone for the Reskit.Org forest.

## How to do it...

1. Creating a new primary forward DNS zone for Cookham.Net

$ZHT1 = @{

  Name              = 'Cookham.Net'

  ResponsiblePerson = 'dnsadmin.cookham.net.'

  ReplicationScope  = 'Forest'

  ComputerName      = 'DC1.Reskit.Org'

}

Add-DnsServerPrimaryZone @ZHT1

1. Creating a reverse lookup zone

$ZHT2 = @{

  NetworkID         = '10.10.10.0/24'

  ResponsiblePerson = 'dnsadmin.reskit.org.'

  ReplicationScope  = 'Forest'

  ComputerName      = 'DC1.Reskit.Org'

}

Add-DnsServerPrimaryZone @ZHT2

1. Registering DNS for DC1, DC2

Register-DnsClient

Invoke-Command -ComputerName DC2 -ScriptBlock {Register-DnsClient}

1. Checking the DNS zones on DC1

Get-DNSServerZone -ComputerName DC1

1. Adding resource records to Cookham.Net zone

# Adding an A record

$RRHT1 = @{

  ZoneName      =  'Cookham.Net'

  A              =  $true

  Name           = 'Home'

  AllowUpdateAny =  $true

  IPv4Address    = '10.42.42.42'

}

Add-DnsServerResourceRecord @RRHT1

# Adding a Cname record

$RRHT2 = @{

  ZoneName      = 'Cookham.Net'

  Name          = 'MAIL'

  HostNameAlias = 'Home.Cookham.Net'

}

Add-DnsServerResourceRecordCName @RRHT2

# Adding an MX record

$MXHT = @{

  Preference     = 10

  Name           = '.'

  TimeToLive     = '4:00:00'

  MailExchange   = 'Mail.Cookham.Net'

  ZoneName       = 'Cookham.Net'

}

Add-DnsServerResourceRecordMX @MXHT

1. Restarting DNS service to ensure replication

Restart-Service -Name DNS

$SB = {Restart-Service -Name DNS}

Invoke-Command -ComputerName DC2 -ScriptBlock $SB

1. Checking results of RRs in Cookham.Net zone

Get-DnsServerResourceRecord -ZoneName 'Cookham.Net'

1. Testing DNS resolution on DC2, DC1

# Testing The CNAME from DC1

Resolve-DnsName -Server DC1.Reskit.Org -Name 'Mail.Cookham.Net'

# Testing the MX on DC2

Resolve-DnsName -Server DC2.Reskit.Org -Name 'Cookham.Net'

1. Testing the reverse lookup zone

Resolve-DnsName -Name '10.10.10.10'

## How it works...

In step 1, which you run on DC1, you create a new primary forward DNS zone for the DNS domain Cookham.Net. In step 2, you create a primary reverse lookup zone for 10.10.10.0/24. In step 3, you run Register-DNSClient on both DC1 and DC2. This ensures that both DCs have updated their DNS details. These three steps produce no console output.

In step 4, you check the DNS zones held by the DNS service on DC1. The output looks like this:

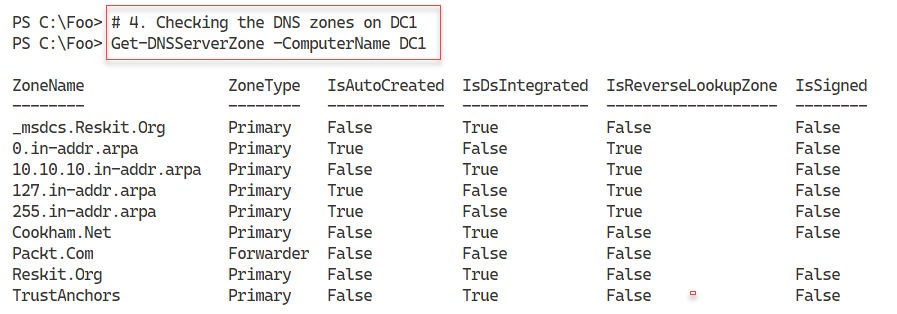


Figure 7.41: Checking the DNS zones on DC1

1. Insert image B1672\_07\_41.png

In step 5, you add three RRs to Cookham.Net: an A record, a CNAME record, and an MX record. In step 6, you restart the DNS service on both DC1 and DC2 to ensure replication has ensured both DNS servers are up to date. These two steps generate no console output.

In step 7, you get all the DNS resource records for Cookham.Net, which looks like this:

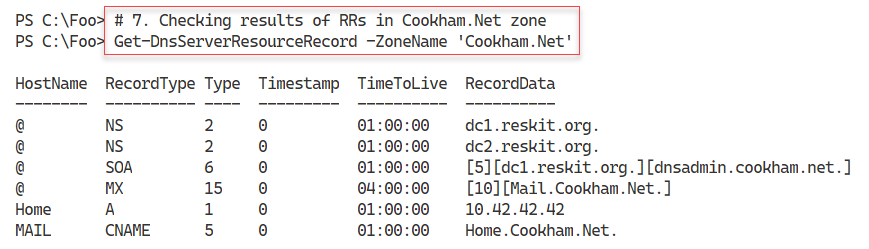


Figure 7.42: Checking the DNS RRs for Cookham.Net

1. Insert image B1672\_07\_42.png

In step 8, you test the DNS name resolution from DC1 and DC2. You first resolve the Mail.Cookham.Net CNAME RR from DC1, then check the MX record from DC2. The output from these two commands is as follows:

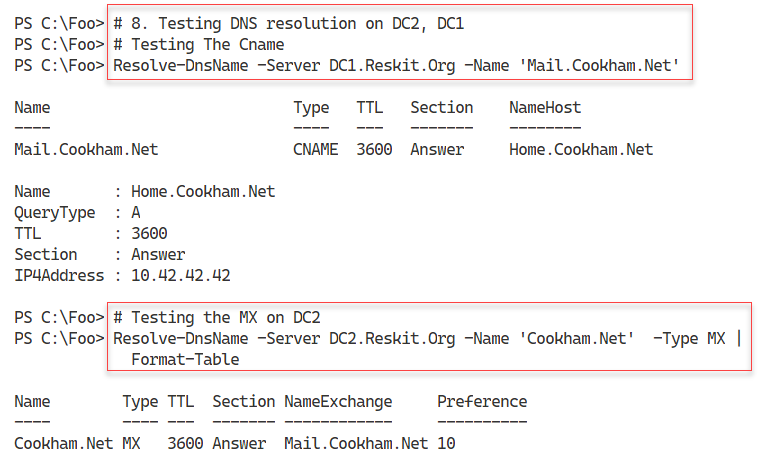


Figure 7.43: Checking the DNS name resolution from DC1 and DC2

1. Insert image B1672\_07\_43.png

In step 9, you test the reverse lookup zone and resolve 10.10.10.10 into a domain name, like this:

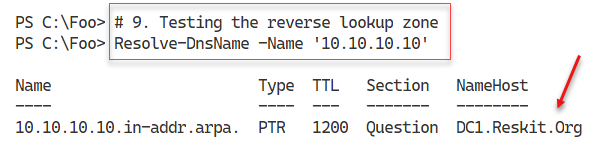


Figure 7.44: Testing the reverse lookup zone

Insert image B1672\_07\_44.png

## There's more...

In step 5, you create some new RRs for the Cookham.Net zone, which you test in later steps. To ensure that AD and DNS replicate the new RRs from, in this case, DC1 to DC2, in step 6, you restart the DNS service on both DCs. Restarting the DNS service ensures replication between the DNS services.